2014

Adaptive Heterogeneous Multi-Population Cultural Algorithm

Mohammadrasool R. RAEESI NAFCHI
University of Windsor

Adaptive Heterogeneous Multi-Population Cultural Algorithm

Mohammadrasool Raeesi Nafchi

A Dissertation
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

May 14, 2014

Adaptive Heterogeneous Multi-Population Cultural Algorithm

by

Mohammadrasool Raeesi Nafchi

APPROVED BY

M. V. dos Santos, External Examiner

Ryerson University

K. Tepe

Electrical and Computer Engineering

S. Goodwin

School of Computer Science

D. Wu

School of Computer Science

Z. Kobti, Advisor

School of Computer Science

May 14, 2014

Declaration of Co-Authorship / Previous Publication

I. Co-Authorship Declaration

I hereby declare that this dissertation incorporates material that is result of joint research. In all cases, the key ideas, primary contributions, experimental designs, data analysis and interpretation, were performed by the author, and the contribution of co-authors was primarily through the proof reading of the published manuscripts.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my dissertation, and have obtained written permission from each of the co-author(s) to include the above material(s) in my dissertation.

I certify that, with the above qualification, this dissertation, and the research to which it refers, is the product of my own work.

II. Declaration of Previous Publication

This dissertation includes 10 original papers that have been previously published/submitted for publication in peer reviewed journals and conferences, as follows:

Section	Full Citation	Publication Status
3.2	M. R. Raeesi N. and Z. Kobti, "A machine operation lists based memetic algorithm for job shop scheduling," in IEEE Congress on Evolutionary Computation (CEC), New Orleans, LA, USA, June 5-8 2011, pp. 2436-2443. [166]	Published

Table 1 – Continued on next page

Table 1: **(continued from previous page)**

Section	Full Citation	Publication Status
3.3	M. R. Raeesi N. and Z. Kobti, "A memetic algorithm for job shop scheduling using a critical-path-based local search heuristic," Memetic Computing, vol. 4 (3), pp. 231-245, 2012. [169]	Published
3.4	M. R. Raeesi N. and Z. Kobti, "Incorporating a genetic algorithm to improve the performance of variable neighborhood search," in The Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC), Mexico City, Mexico, November 5-9 2012, pp. 144-149. [167]	Published
3.5	M. R. Raeesi N. and Z. Kobti, "Incorporating highly explorative methods to improve the performance of variable neighborhood search," Transactions on Computational Science XXI, vol. 8160, pp. 315-338, 2013. [172]	Published
5.2	M. R. Raeesi N. and Z. Kobti, "A knowledge-migration-based multi-population cultural algorithm to solve job shop scheduling," in The 25th Florida Artificial Intelligence Research Society Conference (FLAIRS-25), Marco Island, FL, USA, May 23-25 2012, pp. 68-73. [168]	Published

Table 1 – Continued on next page

Table 1: **(continued from previous page)**

Section	Full Citation	Publication Status
5.3	M. R. Raeesi N. and Z. Kobti, "A multiagent system to solve jssp using a multi-population cultural algorithm," in The 25th Canadian Conference on Artificial Intelligence (Canadian AI), vol. 7310, Toronto, ON, Canada, May 28-30 2012, pp. 362-367. [170]	Published
6.2	M. R. Raeesi N. and Z. Kobti, "Heterogeneous multi-population cultural algorithm," in IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, June 20-23 2013, pp. 292-299. [171]	Published
6.3	M. R. Raeesi N., J. Chittle, and Z. Kobti, "A new dimension division scheme for heterogenous multi-population cultural algorithm," in The 27th Florida Artificial Intelligence Research Society Conference (FLAIRS-27), Pensacola Beach, FL, USA, May 21-23 2014, pp. 75-80. [165]	Published
6.4	M. R. Raeesi N. and Z. Kobti, "Heterogeneous multi-population cultural algorithm with a dynamic dimension decomposition strategy," in The 27th Canadian Conference on Artificial Intelligence (Canadian AI), vol. 8436, Montreal, QC, Canada, May 6-9 2014, pp. 345-350. [174]	Published
6.5	M. R. Raeesi N. and Z. Kobti, "Adaptive heterogenous multi-population cultural algorithm for large scale global optimization," 2014. [173]	Under Review

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my dissertation. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my dissertation does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my dissertation, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my dissertation.

I declare that this is a true copy of my dissertation, including any final revisions, as approved by my dissertation committee and the Graduate Studies office, and that this dissertation has not been submitted for a higher degree to any other University or Institution.

Abstract

Optimization problems is a class of problems where the goal is to make a system as effective as possible. The goal of this research area is to design an algorithm to solve optimization problems effectively and efficiently. Being effective means that the algorithm should be able to find the optimal solution (or near optimal solutions), while efficiency refers to the computational effort required by the algorithm to find an optimal solution. In other words, an optimization algorithm should be able to find the optimal solution in an acceptable time. Therefore, the aim of this dissertation is to come up with a new algorithm which presents an effective as well as efficient performance.

There are various kinds of algorithms proposed to deal with optimization problems. Evolutionary Algorithms (EAs) is a subset of population-based methods which are successfully applied to solve optimization problems. In this dissertation the area of evolutionary methods and specially Cultural Algorithms (CAs) are investigated. The results of this investigation reveal that there are some room for improving the existing EAs. Consequently, a number of EAs are proposed to deal with different optimization problems. The proposed EAs offer better performance compared to the state-of-the-art methods.

The main contribution of this dissertation is to introduce a new architecture for optimization algorithms which is called Heterogeneous Multi-Population Cultural Algorithm (HMP-CA). The new architecture first incorporates a decomposition technique to divide the given problem into a number of sub-problems, and then it assigns the sub-problems to different local CAs to be optimized separately in parallel. In order to evaluate the proposed architecture, it is applied on numerical optimization problems. The evaluation results reveal that HMP-CA is fully effective such that it can find the optimal solution for every single run. Furthermore, HMP-CA outperforms the state-of-the-art methods by offering a more efficient performance.

The proposed HMP-CA is further improved by incorporating an adaptive decomposition technique. The improved version which is called Adaptive HMP-CA (A-HMP-CA) is evaluated over large scale global optimization problems. The results of this evaluation show that HMP-CA significantly outperforms the state-of-the-art methods in terms of both effectiveness and efficiency.

Dedication

I am deeply thankful to my parents, Fereshteh and Parviz Raeisi, who raised me with a love of science and supported me in all my pursuits. I have to appreciate my parents, brother and sisters for their love, support, and sacrifices, while I was far away from home during my graduate studies.

I have saved the last word of appreciation for my loving, supportive, encouraging, and patient wife, Mina Ahmadi, who has been with me all these years and has made them the best years of my life. Without her, this dissertation would never have been written.

This dissertation is lovingly dedicated to my wife and my parents for their constant support, encouragement and unconditional love.

I love you all dearly.

Acknowledgements

There is not enough space, nor time, to acknowledge everyone to the extent they deserve. I hope people understand the brevity of this part.

First and foremost, I want to thank my doctorate supervisor Dr. Ziad Kobti. Under his guidance, I was given the freedom to pursue my research in my own way and I greatly appreciated that freedom. It was a great pleasure to work and discuss with him. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating. The joy and enthusiasm he has for his research was contagious and motivational for me, even during tough times in the Ph.D. pursuit.

Secondly, I would like to thank the theory faculty at the University of Windsor, especially Dr. Richard Frost, for the substantial influence that their courses have had on my research. I gratefully acknowledge the members of my Ph.D. committee, Dr. Scott Goodwin, Dr. Dan Wu and Dr. Kemal Tepe, for their time, interest, and helpful comments and I would also like to thank the external examiner of my Ph.D. dissertation, Dr. Marcus dos Santos, for his valuable feedback on a preliminary version of this dissertation.

I would like to thank my friends in the School of Computer Science at University of Windsor, especially Morteza Mashayekhi and Abbad Golestani, for all the great times that we have shared. I am particularly thankful to my roommates and friends during these years, especially Esmaeil Navaei, Amir Hassannejadasl, Javad Samei, Saber Fallahpour, Armin Sajadi, Mohammad Ghorbani, Abbas Ghaei and Hadi Razavipour, for their infinite support and encouragement.

Contents

List of Tables

List of Figures

Preface

This dissertation presents research outcomes from my PhD studies at the School of Computer Science, University of Windsor, between September 2010 and May 2014. I started my PhD program by reviewing Evolutionary Algorithms (EAs) and their problem domains specially focusing on optimization problems.

I found out that EAs work better if they are combined with local search heuristics. This combination which is called Memetic Algorithm (MA) was introduced before by other researchers. Afterwards, I came up with the idea of incorporating knowledge to improve the search mechanism in EAs. Knowledge extraction within EAs was also previously introduced in the literature as Cultural Algorithm (CA). In order to overcome the premature convergence issue of the CAs, I decided to use the concept of multiple populations which was also presented as Multi-Population Cultural Algorithm (MP-CA) in the literature by that time. Although MAs and MP-CAs were introduced in the literature by other researchers, I proposed a number of ideas to improve the traditional versions. These improvement are described in details in the following chapters.

After reaching the research area of MP-CAs, I provided a detailed survey on the existing MP-CAs in order to define the open problems and the rooms for improvement. Based on the results of this survey I came up with the idea of incorporating problem decomposition within MP-CAs and I called the new algorithm as Heterogeneous Multi-Population Cultural Algorithm (HMP-CA). By conducting extensive

experiments, I presented that my new algorithm offers the competitive performance compared to the state-of-the-art methods.

I improved my proposed HMP-CA by incorporating dynamic decomposition techniques. The new version which is called Dynamic Heterogeneous Multi-Population Cultural Algorithm (D-HMP-CA) presents a better performance compared to HMP-CA such that it offers a fully effective performance to deal with numerical optimization.

In order to improve D-HMP-CA, I proposed a new module to detect the additively interdependency of the problem dimensions. Then I merged D-HMP-CA with the new module which is called Variable Additively Interdependence Learning (VAIL) to a new algorithm and I renamed it Adaptive Heterogeneous Multi-Population Cultural Algorithm (A-HMP-CA). Since it has been shown that D-HMP-CA is fully effective to deal with optimization problems, I moved forward and evaluated A-HMP-CA on large scale global optimization problems. The experimental results represent that my proposed A-HMP-CA significantly outperforms the state-of-the-art methods.

Therefore, I can say that the main contribution of my PhD studies is the new optimization algorithm which is called Adaptive Heterogeneous Multi-Population Cultural Algorithm.

Chapter 1

Introduction

1.1 Problem Statement

Optimization problem is a class of problems where the goal is to make a system as effective as possible. Optimization problems are very well-known due to their applicability in a wide range of research areas. Energy utilization in manufacturing system, minimizing the whole cost in a supply chain management, and finding the minimum value of a mathematical function are three samples of optimization problems.

Each optimization problem has its own input parameters. The input parameters for a job scheduling system, for instance, is the order of jobs to be processed. An optimization algorithm optimizes a problem by adjusting its input parameters. A set of values with respect to all input parameters is called a solution and the set of all feasible solutions is called solution space. For the job scheduling example, any feasible order of jobs is a solution and the order of jobs which results in the minimum cost is the optimal solution.

The optimization problems can be differentiated based on their goals which could be either minimization or maximization. In the former one, the evaluation function is a cost function and the goal is to find out the solution with the minimum cost value,

while the latter one looks for the solution with the maximum fitness value using its evaluation function which is called fitness function. Since the strategies applied to solve both kinds of optimization problems are the same, in the rest of this article only minimization optimization problems are considered.

Definition 1.1.1. An optimization problem is mathematically defined as follows:

$$\begin{aligned} Minimize \quad & f(X) \\ & X = \{x_1, x_2, ..., x_D\} \\ & f : \mathbf{R}^D \rightarrow \mathbf{R} \\ Subject\ to \quad & g_i(X) = 0, i = 1, ..., m \\ & h_j(X) \leq 0, j = 1, ..., n \end{aligned} \tag{1.1}$$

where $f(X)$ denotes to the optimization function where its input is a D-dimensional vector of continuous parameters denoted by X including D parameters denoted by x_i where i is the dimension index. Each optimization function is accompanied with a number of equality and inequality constraints denoted by $g_i(X)$ and $h_j(X)$, respectively.

Based on this definition, a feasible solution would be

$$s = \{v_1, v_2, ..., v_D\} \qquad s \in S$$

where v_i is a value with respect to dimension i and should satisfy $v_i \in D_i$ and all other constraints. S denotes the solution space.

The solution for an optimization problem is to find out a candidate solution with minimum objective value.

$$s^* \in S \qquad and \qquad f(s^*) \leq f(s), \forall s \in S$$

By this definition, s^* is call the global optimal solution.

Generally optimization problems are categorized into two class based on the type of their input parameters, which are as follows:

- *Global Optimization Problems:* Problems with continuous input parameters such as real numbers. In these problems, there is no limit on the number of solutions. Consequently, the solution space is an infinite set. The global optimization problems with a large number of dimensions are called large scale global optimizations.

- *Combinatorial Optimization Problems:* Problems with discrete input parameters such as integers, permutations and graphs. In these problems, although the solution space is a finite set, it is a very large set of solutions.

1.2 Main Objectives

In the research area of optimization problems, the main goal is to design an algorithm to be able to find an optimal solution or near optimal solution within an acceptable time. In other words, an optimization algorithm should be:

- *Effective* in terms of finding an optimal or near optimal solution.

- *Efficient* in terms of the computational complexity it requires to do so.

Therefore, the main goal of my research was to come with a new algorithm to provide a better performance in terms of both effectiveness and efficiency compared to the state-of-the-art methods.

1.3 Approaches

Various kinds of algorithms are proposed to solve optimization problems. For some specific optimization problems, a number of heuristics are designed to work on the corresponding problem domain. Although these heuristics perform very well in these domains, they cannot offer the same performance for other problem domains even similar ones. Local search, for instance, is a heuristic which shows good performance in simple optimization problems specially the ones which have only one local and global optimal solution. Local search starts at an initial random solution and looks for a better solution in the neighborhood of the initial solution. It selects the best solution in that neighborhood and continues the same routine from there until it converges to a solution which is the best one in its neighborhood.

Due to the fact that heuristics are not successful in general, metaheuristics are introduced in the literature. A metaheuristic is formally defined as an iterative master process to guide the operations of subordinate heuristics to efficiently produce high-quality solutions. Metaheuristics which are so-called complex heuristics are more general method with a wider area of applicability. Metaheuristics are approximate algorithms and mostly they are non-deterministic.

Repeated Local Search (RLS), for instance, is an extended version of the simple local search. Since a local search might be trapped into local optimal regions, its execution for a number of times increases the chance of finding an acceptable solution. Iterated Local Search (ILS) is also an extended version of RLS such that it incorporates the information of the best solutions found in the previous executions. In fact, instead of starting from another random solution, ILS slightly perturbs the best solution recently found and continues form there.

Generally, the search mechanism in metaheuristics consists of two phases as follows:

- Exploration: In this phase the metaheuristics look for promising regions by searching unexplored regions. The goal of this phase is to decrease the chance of being trapped into local optimal solutions.

- Exploitation: In this phase the method highly investigates the promising regions to find the local optimal solutions there. The goal of this phase is to speed up the convergence.

It should be noted here that although exploration decreases the chance of immature convergence, too much exploration results in random search which is not going to be converged. Similarly, although exploitation improves the convergence rate, too much exploitation is equivalent to local search only which results in immature convergence. Therefore, the main goal of a metaheuristic is to find a good balance between exploration and exploitation which are also known as diversification and intensification, respectively.

Metaheuristics can be divided into two groups including single point and population-based search methods. The former approaches incorporates only one solution to find a better solution in each iteration, while the latter ones incorporate a population of solutions.

One of the most popular single point search methods is Variable Neighborhood Search (VNS). VNS which is proposed by Mladenovic and Hansen [136] is one of the most recently introduced metaheuristics. VNS can also be considered as an extended version of ILS. The idea of VNS is to incorporate a number of neighborhood structures and switch among them within the local search execution. VNS starts searching with respect to the first neighborhood structure until it converges. At this point, it switch to the next neighborhood structure and looks for a better solution. It continues this routine until it finds a solution which is the best one in its neighborhood with respect to all neighborhood structures.

This strategy helps VNS to decrease the chance of immature convergence dramatically such that it has been successfully applied in various areas including the Traveling Salesman Problem [55], the Open Vehicle Routing problem [57], the p-Median problem [56], and the Graph problem [24].

Evolutionary Algorithms (EAs) is a subset of population-based methods which are successfully applied to deal with optimization problems. Although there are different types of EAs introduced in the literature, they incorporate the same strategy which is called evolution. The key mechanism of evolution is inspired by the natural selection stating that the individuals which are fitter to the environment are more likely the ones surviving for the next generations.

The framework of EAs starts with an initial population of solutions which are usually randomly generated solutions. The solutions are getting evolved for a number of generations by incorporating evolutionary operators including:

- Recombination or crossover.
- Modification or mutation.
- Natural Selection.

The most popular EA is Genetic Algorithm (GA) which is proposed by Holland [93] and improved by various researchers consequently. The successful applications of GAs are also reported in the literature such as the GAs proposed by Lawrence [112], Chen *et al.* [30], Mattfeld *et al.* [130], Pezzella *et al.* [158] and Zhang *et al.* [221] to solve the job shop scheduling problem.

The most recently introduced evolutionary approach in the literature is Differential Evolution (DE). Storn and Price [191] designed DE to deal with global optimization problems. DE has an efficient search space exploration which makes it so popular in this area. Although DE is designed to work in continuous domains, researchers introduced a number of approaches to benefit DE in discrete domains.

Since the population-based search methods are more explorative and the single point search methods are more exploitative, their combination is expected to be a powerful method. Incorporating a local search heuristic within a population-based approach not only improves the quality of the final results, but also helps it to convergence faster. Moscato [139] defined Memetic Algorithm (MA) as a combination of a population-based global search and a local search heuristic to deal with an optimization problem. Although it is not possible to exactly distinguish the exploration and exploitation with in a MA, the population-based global search is highly responsible for search space exploration, while the local search is more likely a exploitative tool. In other words, the global search finds the promising regions and the local search finds the local optimal solutions in those regions.

MAs are described in Chapter 3 with more details. Additionally, this chapter represents my proposed MAs which are published as research papers including:

- A combination of a GA with a local search heuristic to solve a class of combinatorial optimization problem (represented in Section 3.2).
- A GA joint with a local search heuristic to deal with a class of combinatorial optimization problem (described in Section 3.3).
- A GA incorporating VNS to deal with a class of combinatorial optimization problem (illustrated in Section 3.4).
- A combination of a DE and VNS to solve a class of combinatorial optimization problem (characterized in Section 3.5).

Another subclass of EAs is Cultural Algorithms (CAs) which incorporates knowledge to guide its search mechanism. CA which is developed by Reynolds [175] is designed to extract knowledge during the evolutionary process and to incorporate the extracted knowledge to guide the search direction. The architecture of CA is

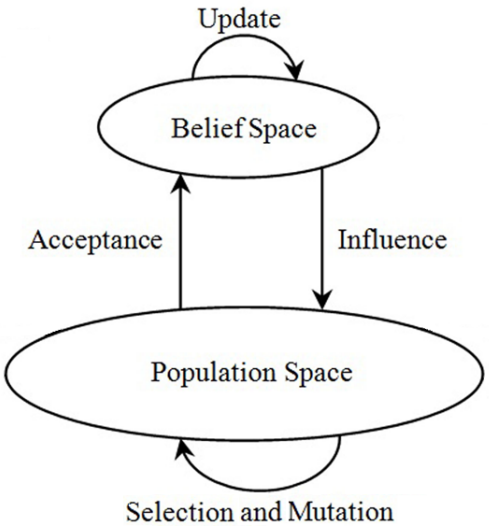


Figure 1.1: CA Architecture

presented in Figure 1.1. As illustrated in this figure, CA uses two different spaces, namely population space and belief space. The population space includes the solutions getting evolved over generations, while the belief space is designed to extract, update and record knowledge. In other words, in addition to the population of solutions which is common in evolutionary methods, CA has another space which is designed to manage the extracted knowledge over generations.

In addition to the spaces, CA incorporates two communication links between them which include:

- Acceptance Function: This link is incorporated to send the best found solutions in the population space to the belief space. The belief space extracts knowledge from the transferred solutions.

- Influence Function: This link transfers the updated knowledge of belief space to the population space which will be used to influence the search direction.

The circulation of information within a CA starts from the population space where the initial solutions are getting evolved for the first time. The best found solutions within the first generation will be transferred to the belief space. The belief space

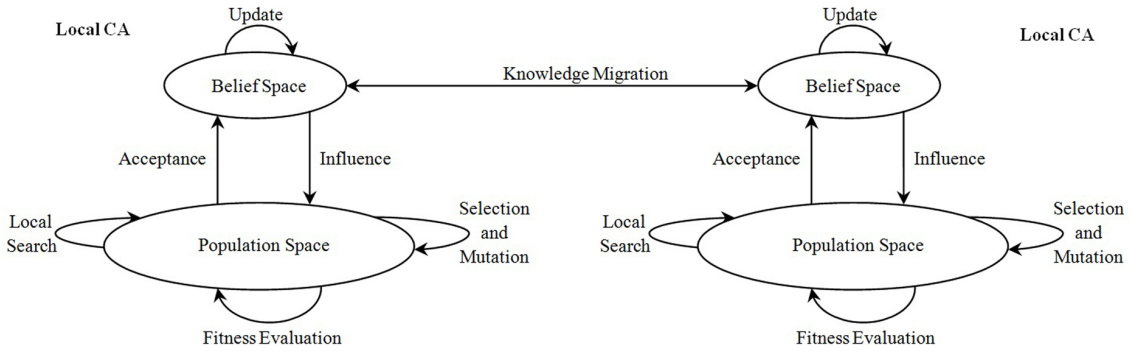


Figure 1.2: MP-CA Architecture [171]

extracts knowledge from the new solutions and updates its own knowledge. The updated knowledge will be transferred back to the population space. From now on, the population space incorporates knowledge-based evolutionary operators instead of the random ones. This routine continues until it converges or the termination criteria have been met.

Although a wide range of successful applications of CAs has been reported in the literature, they are still suffering from their immature convergence. This is mainly due to the fact that they are not capable to preserve the population diversity over generations. Various strategies have been proposed in order to help CAs to overcome this limitation such as rejecting duplicate solutions, incorporating a high mutation rate, and removing solutions with the same objective value. One of the most common strategy is to incorporate multiple populations with lower size instead of a big single population. In Multi-Population Cultural Algorithm (MP-CA), the single population is divided into a number of sub-populations and each sub-population is directed by a local CA. Digalakis and Margaritis [51] introduced the first MP-CA which was designed to schedule electrical generators.

The main architecture of MP-CAs is illustrated in Figure 1.2. In this architecture, there are a number of local CAs each of which has its own belief space. The local CAs communicate with each other by exchanging their extracted knowledge.

There are different architectures to implement MP-CAs such as the MP-CAs with solution migrations between local CAs, the MP-CAs incorporating knowledge migration, and the MP-CAs with a shared belief space among local CAs. Chapter 4 represents a detailed literature survey on the existing architectures to implement MP-CAs. These architectures are categorized based on their problem domains.

One of the best ways to benefit from the multi population concept in MP-CAs in order to preserve the population diversity is to use knowledge migration between local CAs. Chapter 5 represents two proposed MP-CAs incorporating knowledge migration to solve a combinatorial optimization problem. These proposed methods which are published as conference papers include:

- A MP-CA incorporating normative and topographic knowledge (described in Section 5.2).
- A MP-CA with a new proposed knowledge called structured belief (characterized in Section 5.3).

Generally, the existing architectures to implement MP-CAs can be categorized into two classes as follows:

- Homogeneous Local CAs: In these architectures, the local CAs work exactly the same. Although these local CAs may use different algorithm parameters or even different evolutionary approaches, all the local CAs contribute to the same goal which is to solve the whole problem.
- Heterogeneous Local CAs: In these architectures, the local CAs cooperate with each other by working on different parts of the given problem. In other words, the goal of each local CA is to solve a sub-problem instead of dealing with the whole problem.

The architectures in the category of heterogeneous local CAs need the problems to be decomposed into a number of sub-problems. There are a number of approaches

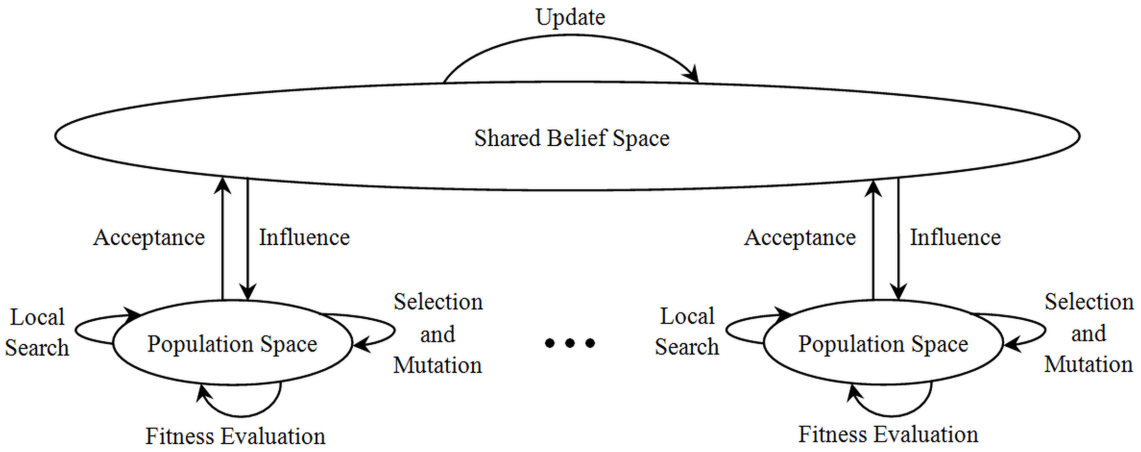


Figure 1.3: HMP-CA Architecture [171]

for problem decomposition which is further described in Chapter 6. This chapter also describes the main contribution of this dissertation which is the proposed Heterogeneous Multi-Population Cultural Algorithm (HMP-CA) which is published as a research paper. The proposed HMP-CA is capable to deal with both static and dynamic dimension decomposition techniques. Figure1.3 illustrates the architecture of the proposed HMP-CA. As presented in the figure, instead of one local belief space for each local CA, there is only belief space which is shared among the local CAs.

The proposed HMP-CA is further improved by incorporating an adaptive dimension decomposition technique. Chapter 6 characterizes the proposed HMP-CA with more details in four sections including:

- Proposed HMP-CA to solve a class of global optimization problems (represented in Section 6.2).
- Investigating the performance of different static decomposition techniques to the performance of HMP-CA (characterized in Section 6.3).
- A HMP-CA incorporating a dynamic decomposition technique to deal with a class of global optimization problems (illustrated in Section 6.4).

- Incorporating an adaptive decomposition technique in HMP-CA to solve a class of large scale global optimization problems (described in Section 6.5).

1.4 Dissertation Organization

As described before, the remainder of this dissertation is structured as follows. The problem domains are presented precisely in Chapter 2. Chapter 3 describes the proposed MAs with more details. A literature survey on the existing architectures to implement MP-CAs is represented in Chapter 4 followed by characterizing the proposed MP-CAs in Chapter 5. The proposed HMP-CA with different dimension decomposition techniques, the key contribution of this dissertation, is described in Chapter 6. Finally, Chapter 7 represents the concluding remarks and future directions for this research.

Chapter 2

Problem Domains

2.1 Introduction

In order to evaluate the proposed methods, both combinatorial and global optimization problems are taken into account. Job Shop Scheduling Problem (JSSP) is considered as a class of combinatorial optimization problems to evaluate the performance of the proposed methods. JSSP is defined as the task of scheduling a number of jobs to be processed on a number of machines. JSSP is a well-known class of combinatorial problems in various areas specially in manufacturing systems.

Within the global optimization problems, numerical optimization is selected for the evaluation. In numerical optimization, the goal is to find the minimum value of a mathematical function with respect to an upper and a lower bounds for input parameters. The optimization function is considered as a black box such that there is no prior knowledge available about it.

The rest of this chapter is organized as follows. JSSP is precisely described in Section 2.2 followed by characterizing numerical optimization in Section 2.3.

2.2 Job Shop Scheduling Problem

Job Shop Scheduling Problem (JSSP) is an instance of combinatorial optimization problems. JSSP is the task of scheduling various jobs to be processed on different machines in a manufacturing system. A lot of work has been done in the last decades to improve the output of a job shop system. In each sample JSSP problem, there are a number of jobs and a number of machines, and each job has a fixed sequence of operations.

In the literature, there are different types of JSSP with different specifications. Classical Job Shop Scheduling Problem is a type of JSSP in which each operation has to be processed only on one machine. This limitation removes machine selection flexibility for operations in classical JSSP. In other words, classical JSSP is the process of sequencing operations of different jobs which are to be processed on the same machine. Flexible Job Shop Scheduling (FJSS), another instance of JSSP, provides the flexibility for each operation to be processed on different machines. Bruker and Schlie [25] published one of the first research considering flexibility in JSSP.

Since there is a flexibility in FJSS to select applicable machine for each operation, FJSS includes two different tasks, machine selection and operation sequencing, while classical JSSP has only operation sequencing task. Therefore, it is obvious that FJSS is more complex than the classical JSSP. To solve FJSS, there are two approaches. The first one is to consider both machine selection and operation sequencing tasks as one problem, while in the second approach each task is solved separately. Hurink *et al.* [97] published a method solving both tasks at the same time and Brandimarte [22] proposed a method to solve them separately.

FJSS could be divided into two subcategories including Total Flexible Job Shop Scheduling (T-FJSS) and Partial Flexible Job Shop Scheduling (P-FJSS). T-FJSS is called to the problems where all the operations have the flexibility to be processed on every available machines. Otherwise, it is called P-FJSS. Zhang *et al.* [221] mentioned

that P-FJSS is more complex than T-FJSS with respect to the computational cost and search space.

Uncertainty is also considered for FJSS problems such as incorporating stochastic processing time. Such problems are called Stochastic Flexible Job Shop (SFJS). A simulation-based decision support system is proposed by Mahdavi *et al.* [126] to deal with a stochastic job shop problem.

Open shop scheduling is also a scheduling problem, but it is very different from JSSP such that the operations of a job in open shop scheduling are sequence independent. A Memetic Algorithm (MA) is proposed by Naderi *et al.* to solve an open shop scheduling problem.

The most important goal of optimizing a scheduling problem is to utilize the manufacturing system. Since the objectives of different manufacturing system could be different, they have their own optimization function. However in most cases, the main objective of a scheduling problem is to minimize the total completion time of processing all the jobs. The total completion time in the area of scheduling is so-called *makespan*. An objective function could be also defined based on different parameters. The problems with such a objective function are called Multi-Objective Job Shop Scheduling Problem (MO-JSSP). For an instance, the objective of a system is to minimize the maximum machine workload, to minimize the total machine workload, as well as to maximize the machine utilization. One approach to optimize such problems is to determine an objective function of parameters with different weights corresponding to their importance. Another approach is to define one as the main objective and set others as constraints. Mahdavi *et al.* [126] used generalized response surface methodology to do so.

2.2.1 Problem Definition

In all types of JSSP, there are a number of jobs to be processed on a set of machines. Machines are denoted by m_k, where k refers to the machine's index ranging from 1 to M, the number of machines, and jobs are denoted by J_i, where i refers to the job's index which is between 1 and N, the number of jobs. Each job consists of a number of operations to be processed on a predefined sequence. The number of operations for each job is denoted by NO_i, and O_{ij} refers to the j^{th} operation of the i^{th} job. In deterministic JSSP, the processing time of each operation on each applicable machine is known at the beginning, which is denoted by $PT\left(O_{ij}, m_k\right)$, meaning the processing time of j^{th} operation of the i^{th} job on k^{th} machine.

A sample T-FJSS problem is represented in Table 2.1 which is the 4^{th} benchmark incorporated by Xing *et al.* [208]. In this test problem, there are 10 jobs including 3 operations to be processed on 10 different machines.

2.2.2 Rules

In JSSP problems, there are a number of rules to be considered which are as follows:

- All the jobs and machines are available at the beginning.
- There is no due date for the jobs.
- The part movement time between machines and the machine set up time are assumed to be negligible.
- The sequence of operation for a job is fixed. The third operation of a job, for instance, cannot be processed until the second operation of the same job has been done.
- The operations of different jobs are independent to each other.

Table 2.1: 4^{th} FJSS benchmark incorporated by Xing et al. [208].

Job	Operation	Processing Time on Different Machines									
		m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
J_1	O_{11}	1	4	6	9	3	5	2	8	9	5
	O_{12}	4	1	1	3	4	8	10	4	11	4
	O_{13}	3	2	5	1	5	6	9	5	10	3
J_2	O_{21}	2	10	4	5	9	8	4	15	8	4
	O_{22}	4	8	7	1	9	6	1	10	7	1
	O_{23}	6	11	2	7	5	3	5	14	9	2
J_3	O_{31}	8	5	8	9	4	3	5	3	8	1
	O_{32}	9	3	6	1	2	6	4	1	7	2
	O_{33}	7	1	8	5	4	9	1	2	3	4
J_4	O_{41}	5	10	6	4	9	5	1	7	1	6
	O_{42}	4	2	3	8	7	4	6	9	8	4
	O_{43}	7	3	12	1	6	5	8	3	5	2
J_5	O_{51}	7	10	4	5	6	3	5	15	2	6
	O_{52}	5	6	3	9	8	2	8	6	1	7
	O_{53}	6	1	4	1	10	4	3	11	13	9
J_6	O_{61}	8	9	10	8	4	2	7	8	3	10
	O_{62}	7	3	12	5	4	3	6	9	2	15
	O_{63}	4	7	3	6	3	4	1	5	1	11
J_7	O_{71}	1	7	8	3	4	9	4	13	10	7
	O_{72}	3	8	1	2	3	6	11	2	13	3
	O_{73}	5	4	2	1	2	1	8	14	5	7
J_8	O_{81}	5	7	11	3	2	9	8	5	12	8
	O_{82}	8	3	10	7	5	13	4	6	8	4
	O_{83}	6	2	13	5	4	3	5	7	9	5
J_9	O_{91}	3	9	1	3	8	1	6	7	5	4
	O_{92}	4	6	2	5	7	3	1	9	6	7
	O_{93}	8	5	4	8	6	1	2	3	10	12
J_{10}	$O_{10,1}$	4	3	1	6	7	1	2	6	20	6
	$O_{10,2}$	3	1	8	1	9	4	1	4	17	15
	$O_{10,3}$	9	2	4	2	3	5	2	4	10	23

- Each machine can process only one operation at a time which cannot be interrupted.
- At any given time, only one operation of a job can be processed.

There are a number of concepts in JSSP which will be used later in this article. These concepts are clarified as follows:

- Critical Path: A sequence of immediately consecutive operations starting from time zero to the completion time is called a critical path. A critical path can be found by tracing the schedule from its completion time back to the beginning.
- Critical Operation: An operation belongs to one or more critical path is called a critical operation. In other words, critical operations are the ones for which any delay in their processing increases the makespan.
- Critical Block: A set of immediately consecutive critical operations on the same machine is called a critical block.
- Active Schedule: As defined by Croce *et al.* [43], if there is no operation in a schedule that can be started earlier without delaying another operation, the schedule is called active.
- Job's Operation Sequence: Each job has its own fixed sequence of operations.
- Job-Predecessor Operation: This operation is the preceding operation of an operation in a job's operation sequence.
- Job-Successor Operation: This operation is the succeeding operation of an operation in a job's operation sequence.
- Machine's Operation Sequence: The sequence of operations to be processed on the same machine.

Table 2.2: A sample classical job shop scheduling problem

Operation Index	1	2	3
J_1	$m_2, 1$	$m_1, 2$	$m_3, 3$
J_2	$m_1, 2$	$m_2, 1$	$m_3, 2$
J_3	$m_1, 2$	$m_3, 4$	$m_2, 1$

- Machine-Predecessor Operation: This operation is the operation to be processed exactly before an operation on the same machine.

- Machine-Successor Operation: This operation is the operation to be processed exactly after an operation on the same machine.

The concept of an active schedule is very important in JSSP, since it can limit the search space in order to find the optimal solution faster. Based on the definition of an active schedule, an optimal solution is more likely to be an active schedule, and if it is inactive, it has an equivalent active solution which is optimal too. Therefore, instead of searching the whole solution space, just the active solutions are investigated.

If each operation is assigned to its earlier appropriate interval on its corresponding machine, the final schedule will be active. This rule is used by Hasan *et al.* [87] as Gap Reduction rule, and Becerra and Coello Coello [14] called this rule as permissible left shift and used it as a priory knowledge in their CA.

2.2.3 A Simple Problem

In order to illustrate the concept of an active schedule, a simple classical JSSP is considered which is represented in Table 2.2. The sample problem contains 3 jobs including 3 operations to be processed on 3 machines. The table shows the applicable machine with its corresponding processing time for each operation. For an instance, the third operation of the second job O_{23}, should be processed for 2 time units on the third machine. Figure 2.1 depicts a sample schedule for this sample problem.

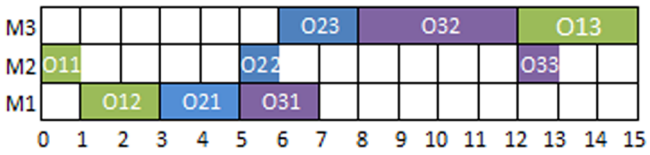


Figure 2.1: Sample Schedule

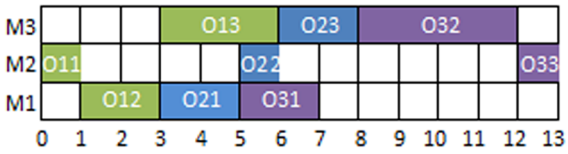


Figure 2.2: Active Schedule

Clearly, the sample schedule represented in Figure 2.1 is not an active schedule, since the 3^{rd} operation of the 1^{st} job can be processed before other operations without delaying them. By reassigning the operation to the earliest appropriate interval, the new schedule will be an active one which is shown in Figure 2.2. This example also shows that how much this rule can improve a schedule.

2.2.4 JSSP Representation

JSSP representation is more critical than the representation of other optimization problems since JSSP cannot be normally represented as a permutation. Various JSSP representations have been introduced in the literature. Nine different representations are introduced by Cheng *et al.* [32] including operation-based representation, job-based representation, preference list-based representation, job pair relation-based representation, priority rule-based representation, disjunctive graph-based representation, completion time-based representation, machine-based representation, and random keys representation.

Overall, chromosome representations are categorized into two sections: the direct representations and indirect ones. The representations encoding schedules are called direct, which include operation-based representation, job-based representation, job pair relation-based representation, completion time-based representation, and random

keys representation, while others that encode the schedule construction rules are considered as indirect representation [32].

Ponnambalam *et al.* [159] conducted a comparison among four different representations including operation-based representation, job-based representation, preference list-based representation, and priority rule-based representation. The authors claimed that as the results of the comparison preference list-based representation offers better results, while the operation-based representation is the fastest one.

One of the popular representation is MSOS representation which is proposed by Zhang *et al.* [221]. This representation is one of the operation-based representations and it is also called modified operation-based representation. Its encoding and decoding procedures are easy and efficient. All the possible schedules encoded by this representation are feasible. So there is no need to any repair mechanisms. This representation consist of two parts, namely Machine Selection (MS) and Operation Sequence (OS).

The MS part determines a machine for each operation to be processed on. The MS part is not applicable in classical JSSP since there is only one applicable machine for each operation. In other words, the MS part remains the same for all possible schedules. The MS part is a string of machines indices with the length of the total number of operations. Each entry of MS string represents the index of one of the applicable machines for the corresponding operation.

The OS part which is also known as permutation with repetition represents the sequence of operations to be processed in order. The OS part is a string of job indices with the same length as MS. Each index denotes one operation such that the job index determines the job that operation belongs to, and the occurrence of that job index from the beginning of the string shows the index of the operation in its job's

operation sequence. Consider

$$OS = [1, 1, 2, 3, 2, 1, 2, 3, 3]$$

as a sample OS for the sample problem which represents the following sequence of operations.

$$O_{11} \prec O_{12} \prec O_{21} \prec O_{31} \prec O_{22} \prec O_{13} \prec O_{23} \prec O_{32} \prec O_{33}$$

This sequence for the sample problem produces the following schedule.

$$m_1 : O_{12} \prec O_{21} \prec O_{31}$$

$$m_2 : O_{11} \prec O_{22} \prec O_{33}$$

$$m_3 : O_{13} \prec O_{23} \prec O_{32}$$

This schedule is illustrated in Figure 2.2.

However, there are different presentations introduced for JSSP. Each one has its own strengths and weaknesses. Some of them are designed for specific purpose.

2.3 Numerical Optimization

Numerical optimization is a class of global optimization problems where the goal is to find the optimal objective value of a mathematical optimization function with a set of continuous input parameters. Equations 2.1 and 2.2 are two sample numerical optimization functions which are called sphere model and generalized Rosenbrock's function, respectively. The goal of these problems is to find the minimum value while satisfying both upper and lower bounds for each dimension.

$$f_1(X) = \sum_{i=1}^{D} {x_i}^2 \tag{2.1}$$

$$-100 \leq x_i \leq 100$$

$$min(f_1) = f_1(0, ..., 0) = 0$$

$$f_2(X) = \sum_{i=1}^{D-1} \left| 100 \left({x_{i+1}} - {x_i}^2 \right)^2 + (x_i - 1)^2 \right| \tag{2.2}$$

$$-30 \leq x_i \leq 30$$

$$min(f_2) = f_2(1, ..., 1) = 0$$

The optimization problems with a large number of dimensions are called large scale global optimizations. The common number of dimensions in this area is 1000.

In numerical optimization, the optimization function is considered as a black box which gets a D-dimensional vector as its input parameters and returns a real value as the objective value corresponding to the given input vector.

The numerical optimization functions are usually categorized into three classes based on the interdependency of their input variables. These classes include fully

separable functions, partially separable functions and non-separable functions. A fully separable function is defined in Definition 2.3.1 [193].

Definition 2.3.1. A function $f(X)$ is fully separable if and only if

$$\arg \min_{(x_1,...,x_n)} f(x_1, ..., x_n) = \left(\arg \min_{x_1} f(x_1, ...), ..., \arg \min_{x_n} f(..., x_n) \right) \tag{2.3}$$

The fully separable functions are the functions that can be optimized by optimizing their dimensions separately since there is no interdependencies between their dimensions. These interdependencies are also known as variable interactions. The partially separable functions and the non-separable functions are also defined in Definition 2.3.2.

Definition 2.3.2. The optimization functions where the interacting variables can be grouped into a number of independent subsets are called partially separable functions, and the functions where there is only one independent group including all the dimensions are called non-separable functions.

Based on Definitions 2.3.1 and 2.3.2, the sphere model represented in Equation 2.1 is a fully separable function while the generalized Rosenbrock's function illustrated in Equation 2.2 is a non-separable function. Function f_{PS} represented in Equation 2.4 which is a composite function of two generalized Rosenbrock's functions is a sample of partially separable functions.

$$f_{PS}(x_1, ..., x_5, x_6, ..., x_D) = f_2(x_1, ..., x_5) + f_2(x_6, ..., x_D) \tag{2.4}$$

2.3.1 Well-Known Numerical Optimization Functions

There are a number of numerical optimization functions which are very well-known such that researchers in this area mostly use them to evaluated their proposed methods. These functions include (but not limited to):

f_1 - Sphere Model:

$$f_1(X) = \sum_{i=1}^{D} {x_i}^2 \tag{2.5}$$

$$-100 \leq x_i \leq 100$$

$$min(f_1) = f_1(0, ..., 0) = 0$$

f_2 - Generalized Rosenbrock's Function:

$$f_2(X) = \sum_{i=1}^{D-1} \left| 100 \left(x_{i+1} - {x_i}^2 \right)^2 + (x_i - 1)^2 \right| \tag{2.6}$$

$$-30 \leq x_i \leq 30$$

$$min(f_2) = f_2(1, ..., 1) = 0$$

f_3 - Generalized Schwefel's Problem 2.26:

$$f_3(X) = \sum_{i=1}^{D} \left(-x_i sin \left(\sqrt{|x_i|} \right) \right) \tag{2.7}$$

$$-500 \leq x_i \leq 500$$

$$min(f_3) = f_3(420.9687, ..., 420.9687) \simeq -12569.5$$

f_4 - Generalized Rastrigin's Function:

$$f_4(X) = \sum_{i=1}^{D} \left[{x_i}^2 - 10cos\left(2\pi x_i\right) + 10 \right] \tag{2.8}$$

$$-5.12 \leq x_i \leq 5.12$$

$$min(f_4) = f_4(0, ..., 0) = 0$$

f_5 - Ackley's Function:

$$f_5(X) = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} cos\left(2\pi x_i\right)\right) + 20 + e \quad (2.9)$$

$$-32 \leq x_i \leq 32$$

$$min(f_5) = f_5(0, ..., 0) = 0$$

f_6 - Generalized Griewank's Function:

$$f_6(X) = \frac{1}{4000}\sum_{i=1}^{D} {x_i}^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (2.10)$$

$$-600 \leq x_i \leq 600$$

$$min(f_6) = f_6(0, ..., 0) = 0$$

f_7 and f_8 - Generalized Penalized Functions:

$$f_7(X) = \frac{\pi}{D}\left\{10\sin^2\left(\pi y_1\right) + \sum_{i=1}^{D-1}\left(y_i - 1\right)^2\left[1 + 10\sin^2\left(\pi y_{i+1}\right)\right] + \left(y_D - 1\right)^2\right\} + \sum_{i=1}^{D} u\left(x_i, 10, 100, 4\right) \quad (2.11)$$

$$-50 \leq x_i \leq 50$$

$$min(f_7) = f_7(-1, ..., -1) = 0$$

$$f_8(X) = 0.1\Bigg\{ \sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_{i+1})\right] + (x_D - 1)^2 \left[1 + \sin^2(2\pi x_D)\right] \Bigg\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4) \quad (2.12)$$

$$-50 \leq x_i \leq 50$$

$$min(f_8) = f_8(1, ..., 1) = 0$$

where

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & \text{if } x_i > a \\ 0 & \text{if } -a \leq x_i \leq a \\ k(-x_i - a)^m & \text{if } x_i < -a \end{cases}$$

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

f_9 - Schwefel's Problem 1.2:

$$f_9(X) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2 \quad (2.13)$$

$$-500 \leq x_i \leq 500$$

$$min(f_9) = f_9(0, ..., 0) = 0$$

f_{10} - Schwefel's Problem 2.21:

$$f_{10}(X) = max_i(|x_i|, 1 \leq i \leq D) \quad (2.14)$$

$$-500 \leq x_i \leq 500$$

$$min(f_{10}) = f_{10}(0, ..., 0) = 0$$

f_{11} - Schwefel's Problem 2.22:

$$f_{11}(X) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i| \tag{2.15}$$

$$-500 \leq x_i \leq 500$$

$$min(f_{11}) = f_{11}(0, ..., 0) = 0$$

f_{12} - Step Function:

$$f_{12}(X) = \sum_{i=1}^{D} \lfloor x_i + 0.5 \rfloor^2 \tag{2.16}$$

$$-100 \leq x_i \leq 100$$

$$min(f_{12}) = f_{12}(0, ..., 0) = 0$$

In addition to these functions, there are a number of benchmarks available in the literature to provide a better platform to compare different optimization methods. The benchmark functions for the CEC'2010 competition on large scale global optimization [193] is one of the well-known benchmarks in the literature. This benchmark includes twenty 1000-dimensional numerical optimization functions which can be categorized into the three following classes based on Definitions 2.3.1 and 2.3.2:

1. Shifted Fully Separable Functions ($f_1 - f_3$).

2. Shifted Partially Separable Functions ($f_4 - f_{18}$).

3. Shifted Nonseparable Functions ($f_{19} - f_{20}$).

Chapter 3

Memetic Algorithm

3.1 Introduction

Most of the recent work in job shop scheduling hybridized GAs with local search algorithms to improve the final results as well as to help the convergence rate.

As defined by Moscato [139], Memetic Algorithm (MA) is a hybrid method including a population-based global search combined with a local search heuristic to be used to solve an optimization problem. Ong *et al.* [149] considered population-based search as exploration to find the promising regions in the search space, and local search as exploitation to find the best individual in those neighborhoods. Based on this definition, all the hybridizations of GAs or other EAs with local search heuristics can be considered as MAs.

For instance, the MA proposed by Yang *et al.* [213] is a combination of Clonal Selection and Simulated Annealing. A combination of GA and Tabu Search was also proposed by Ombuki [147]. Kacem *et al.* [103] combined an EA with a localization approach. They also proposed a combination of EA with fuzzy logic [104]. Hassan *et al.* [84, 85, 86] proposed different GAs combined with different priority rules.

Recently, Hasan *et al.* [87] proposed three different MAs which are combinations of GAs with three different priority rules to solve JSSP.

Incorporating a local search heuristic within a population-based global search has two major effects on the search performance including:

- Improves the quality of the final results.
- Speeds up the convergence process.

This chapter represents my proposed MAs to deal with optimization problems which are published as research papers including:

- M. R. Raeesi N. and Z. Kobti, "A machine operation lists based memetic algorithm for job shop scheduling," in IEEE Congress on Evolutionary Computation (CEC), New Orleans, LA, USA, June 5-8 2011, pp. 2436-2443 [166]:

  In this research paper, a MA is proposed to solve classical JSSPs. The proposed MA is a combination of a GA with a local search heuristic. Furthermore, a new representation for JSSP is introduced in this paper which is called Machine Operation Lists (MOL) representation. The proposed MA is designed to deal with the MOL representation. This research paper is represented in Section 3.2.

- M. R. Raeesi N. and Z. Kobti, "A memetic algorithm for job shop scheduling using a critical-path-based local search heuristic," Memetic Computing, vol. 4 (3), pp. 231-245, 2012 [169]:

  This research paper also proposed a MA to deal with JSSPs. Unlike the previous paper, it is a more general method applicable on various types of JSSPs. The results of its successful applications on three different types of JSSP have been reported in the published paper which include classical, flexible and multi-objective flexible JSSPs. In addition to the proposed MA, a new fitness function

is introduced which is called Priority-Based Fitness Function. This function incorporates the main fitness evaluation as the highest priority level and various useful parameters as the next priority levels. Therefore, if a tie happens in selection method based on the main fitness value, instead of selecting a solution randomly, the better solution will be selected based on its other parameters considered as the next priority levels. The details of the proposed MA and its new fitness function is further described in Section 3.3.

- M. R. Raeesi N. and Z. Kobti, "Incorporating a genetic algorithm to improve the performance of variable neighborhood search," in The Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC), Mexico City, Mexico, November 5-9 2012, pp. 144-149 [167]:

  The main contribution of this research paper is to show that VNS is more likely a local search approach and it can be highly improved if joined with a population-based global search. Therefore, the proposed MA in this research paper is a combination of a GA and a VNS to deal with JSSPs. Section 3.4 further describes the proposed MA.

- M. R. Raeesi N. and Z. Kobti, "Incorporating highly explorative methods to improve the performance of variable neighborhood search," Transactions on Computational Science XXI, vol. 8160, pp. 315-338, 2013 [172]:

  This research paper is an extension of the previous one, which incorporates more explorative global search to improve the performance of VNS. The proposed MA in this published paper is a combination of a DE and a VNS. Since DE has a very effective search space exploration mechanism, it is a good candidate to be hybridized with local search heuristics. Furthermore, it has been shown that incorporating multiple populations within the DE makes it even more explorative. The proposed MA is more characterized in Section 3.5.

3.2 A Machine Operation Lists Based Memetic Algorithm for Job Shop Scheduling

Abstract. In this article, a new Memetic Algorithm (MA) has been proposed to solve Job Shop Scheduling Problems. The proposed MA is based on Machine Operation Lists (MOL), which is the exact sequence of operations for each machine. Machine Operation Lists representation is a modification of Preference List-Based representation. Linear Order Crossover (LOX) and Random operations are first considered as crossover and mutation operators for the proposed MA. Local Search heuristic (LS) of the proposed MA reconsiders all the operations of a job. It chooses a job and removes all of its operations and finally reassigns them again one by one in their sequencing order to improve the fitness value of the schedule. The proposed algorithm has been applied on the well-known benchmark of classical Job Shop Scheduling Problems (JSSP). Comparing it with the existing methods shows that the proposed MA and the proposed Genetic Algorithm (GA) without LS are effective in JSSP. Moreover, comparing the results of MA and GA shows that using LS not only improves the final results but also helps GA to converge to the final solution.

3.2.1 Introduction

Job shop scheduling, which is the scheduling of various operations on different machines, is an open problem in manufacturing systems. Because of its applicability in different fields, job shop scheduling is a well-known optimization problem. Since job shop scheduling is an NP-Complete problem, there is no exact solution for it. It has been prooved by Garey *et al.* [65]that job shop scheduling systems with more than two machines are NP-Complete problems.

Makespan is the main goal for almost all the manufacturing systems, while some of them combine it with different parameters such as machine utilization and total job completion time to define the their own main goal. Makespan is the maximum jobs completion time, which is the end time of the whole schedule. There are two important definitions in scheduling including active schedule and non-delay schedule. Croce *et al.* [43] defined the active schedule as an schedule in which there is no operation that can be started earlier without delaying another operation while obeying the sequencing constraints. Non-delay schedules defined by French [59] are schedules in which there is no waiting operation while the applicable machine is idle. Based on these definitions, the non-delay schedules are a subset of active schedules. Moreover, the optimal solution is always an active schedule, not necessarily a non-delay one.

There are a lot of research that has been done in job shop scheduling. Different types of algorithms have been proposed to solve the scheduling problem. The proposed algorithms contain a wide range of algorithm types such as heuristics, meta-heuristics and evolutionary algorithms. Since each type of algorithm has its own strengths and weaknesses, it is not possible to select an algorithm as the best one. Huang *et al.* [96] and Asano *et al.* [9] proposed two heuristics for job shop scheduling, bottleneck shifting procedure and tree search method, respectively. Simulated Annealing (SA) and Tabu Search (TS) are instances of meta-heuristic approaches used by Kolonko [108] and Pezzella *et al.* [157], respectively.

Ant Colony Optimization (ACO), Genetic Algorithms (GA), hybrid GAs, and Memetic Algorithms (MA) are the proposed methods in Evolutionary Algorithm (EA) category. Due to the characteristics of job shop scheduling problem, the most important issue of population-based algorithms in this field is chromosome representation, based on which the population operators such as crossover are determined. Various chromosome representations have been presented in the literature which are described in the next subsection.

In this article, a new chromosome representation is proposed which is a modification of the preference list-based representation. The proposed representation is called Machine Operation Lists (MOL). Based on MOL representation, a Memetic Algorithm is proposed to solve scheduling problems. The existing chromosome representations for job shop scheduling are presented in Subsection 3.2.2. Subsection 3.2.3 describes the definition of the classical job shop scheduling problem. The proposed MA is described in details in Subsection 3.2.4. The results of applying the proposed algorithm on a well-known job shop scheduling benchmark are presented in Subsection 3.2.5, followed by presenting conclusions in Subsection 3.2.6.

3.2.2 Related Works

There are different evolutionary algorithms proposed for job shop scheduling. The proposed EAs in this field can be divided into two categories, the plain EAs and the ones combined with a Local Search (LS). The combination of any kind of Evolutionary Algorithm (EA) with a Local Search heuristic is called Memetic Algorithm (MA) by Moscato [139]. He defined MA as a joint of a population-based global search and neighborhood local search. The population-based global searches explore the search space to find the best region and then local search looks for the best individual in that neighborhood region [149].

As the first EA used in scheduling problems, Lawrence [112] proposed a GA to solve the job shop scheduling. The Ant Colony Optimization (ACO) method proposed by Wang *et al.* [202], and the GA proposed by Mattfeld *et al.* [130] are the samples of plain Evolutionay Algorithms proposed for job shop scheduling.

Different MAs are proposed by Park et al. [154] and Caumond et al. [28] which are the combination of a GA with local search heuristics. Huang *et al.* [95] combined ACO with TS to find the optimal solution, and Ombuki [147] presented the combination of

GA with TS. Yang *et al.* [213] proposed a Memetic Algorithm which is a combination of Clonal Selection and Simulated Annealing (SA).

The combination of GAs with different priority rules are presented by Hassan *et al.* [84, 85, 86]. The most recent such combination proposed by Hassan *et al.* [87] considered different priority rules including Partial Reordering, Gap Reduction, and Restricted Swapping. Patial Reordering reorders the operations by assigning the first operation of the bottleneck job to the first position on the applicable machine. The process of assigning the first appropriate interval to the operation is called Gap Reduction which reduces the gaps. Gap Reduction rule makes all the schedules to be active. The rule which tries to find the smaller makespan using swapping the adjacent operation on the same machine is called Restricted Swapping.

As mentioned before, the main characteristic of an Evolutionary Algorithm in job shop scheduling problems is its chromosome representation.Various chromosome representations have been introduced in literature. Cheng *et al.* [32] described nine different representations including operation-based representation, job-based representation, preference list-based representation, job pair relation-based representation, priority rule-based representation, disjunctive graph-based representation, completion time-based representation, machine-based representation, and random keys representation.

Representation can be divided into two classes: direct and indirect ones. Direct representations are those ones which encode their corresponding schedule, while indirect ones encode some rules instead of the schedule and use them to construct their corresponding schedule. In other words, in direct representation for instance, a Genetic Algorithm is used to evolve the schedules while, in indirect approach, GA evolves the encoding rules. Based on this classification, operation-based representation, job-based representation, job pair relation-based representation, completion

time-based representation, and random keys representation are considered as direct representations, while others are indirect [32].

Ponnambalam *et al.* [159] compared four different representations; two directs and two indirects. The compared representations contain operation-based representation, job-based representation, preference list-based representation, and priority rule-based representation. The result of comparison shows that the best schedule which has the smallest makespan belongs to preference list-based representation. As it is expected, they presented that the operation-based representation is the best one with regards to the CPU time. The main reason is that this representation is direct, so there is no need to construct a schedule using some priority rules.

Operation-based representation ,one of the most well-known representation, is a string of operation IDs which denotes the sequence of operations. The length of the string is equal to the total number of all operations of all jobs. This representation is proposed by Fang *et al.* [54]. Since all the possible sequence of operations are not feasible schedules, it needs a procedure to check the feasibility and may be to correct it. Gen *et al.* [66] modified the operation-based representation such that each operation is denoted by its job ID. In this representation, the operation ID for each gene is determined by the occurence of the corresponding job ID. So all the possible chromosome instances are feasible schedules.

The proposed representation in this article is a modification of preferece list-based representation. In preference list-based representation, there is a list of operations for each machine which determines the priority of each operation for its applicable machine. Like operation-based representation, the length of preference list-based representation is equal to the total operation number. Prefence list-based representation is first proposed by Davis [47]. Croce *et al.* [43] used this representation for their genetic algorithm. They used a procedure called lookahead evaluation to schedule

Table 3.1: A sample classical job shop scheduling problem

Operation Index	1	2	3
J_1	$m_2, 1$	$m_1, 2$	$m_3, 3$
J_2	$m_1, 2$	$m_2, 1$	$m_3, 2$
J_3	$m_1, 2$	$m_3, 4$	$m_2, 1$

the higher priority operations first. The proposed method is described in details in the Sub-subsection 3.2.4.1.

3.2.3 Problem Definitions

Classical job shop scheduling, as Baker [11] defined it, is a process of scheduling a set of jobs on a set of machines. The number of machines and jobs are denoted by M and N, respectively. Machines and jobs are denoted by m_k and J_i, respectively, where k is the machine index, between 1 to M, and i is the job index which is from 1 to N. Each job includes a sequence of operations by O_{ij} which means the j^{th} operation of the i^{th} job. Each operation is applicable on only one machine for which the processing time is known. Machine set up time and part movement time between machines are considered negligible. Table 3.1 presents a sample classical job shop scheduling problem which contains 3 jobs to be processed on 3 machines. This table shows the applicable machine with the corresponding processing time for each operation. The third operation of the second job O_{23}, for example, is applicable on the third machine, and it needs 2 time units to be processed.

There are some rules for classical job shop scheduling which are as follows: only one operation of each job is applicable on a specific machine. Only one operation at a time can be processed on a machine, which can not be interrupted. The operations of a job are independent to the other jobs operations. All jobs and machines are available at the beginning, and there is no due date.

3.2.4 Proposed Memetic Algorithm

In this article, a new Memetic Algorithm has been proposed which is based on a new chromosome representation called Machine Operation Lists (MOL). The proposed MA method is an appropriate Genetic Algorithm considering the new representation, combined with a Local Search which is also based on the proposed representation. The Local Search heuristic reassigns all the operations of a job to find the best schedule with the smallest makespan. The reassigning procedure for each operation is independent of the others, which makes the whole reassigning step faster.

3.2.4.1 Chromosome Representation

The proposed MOL representation is a modification of the preference list-based representation. It consists of the operation lists for each machine. The operation list determines the sequence of operations for the corresponding machine. Since all the jobs have only one operation for each machine, the operations are denoted by their job ID in each list. A possible chromosome for the sample problem presented in Table 3.1 could be

$$\{(2,3,1),(1,2,3),(2,3,1)\}$$

which determines the following operation lists for each machine.

$$m_1 : O_{21} \prec O_{31} \prec O_{12}$$

$$m_2 : O_{11} \prec O_{22} \prec O_{33}$$

$$m_3 : O_{23} \prec O_{32} \prec O_{13}$$

The decoding procedure for this representation starts with the first operations in each list. The first operations which are ready to be scheduled are removed from the

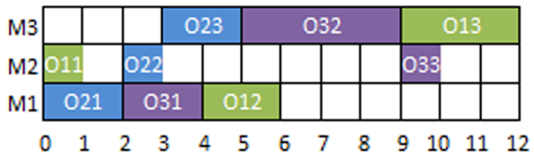


Figure 3.1: A sample constructed schedule

lists and assigned to the first appropriate interval on the applicable machine. This routine continues until all operations are scheduled. Figure 3.1 shows the constructed schedule for the sample chromosome.

As it can be seen in this representation, there are some chromosomes whose corresponding schedules are not feasible. In those cases, a deadlock will occur in the decoding procedure. To remove the infeasible chromosomes, another concept is defined. The concept is that the operation lists can be considered as the exact operation sequence or a preference list. The exact operation sequence is a fixed sequence of operation for which the priority of operations should be saved, while for preference list, the sequence of operations are flexible due to the deadlocks. For example, to generate a random chromosome, first, for each machine, a random operation list is generated. Each list includes all the operations which should be processed on the corresponding machine in random order. Then, to decode the representation, all the operation lists are considered as preference list, such that in deadlock situations the next operation in the list will be selected to be scheduled. Since, in the decoding step, the operation lists are modified to construct a feasible schedule, afterwards they will be considered as exact operation sequence.

After applying the genetic operators, both crossover and mutation, the operation lists which are remained the same as before are considered as exact operation sequence, while the changed lists will be set as preference lists. In this way, if any deadlock occurs, only the preference lists will be modified. Since the unchanged lists construct a feasible schedule, they are consistent and do not need to be modified. In other words, there is at least one feasible schedule which has the same operation lists

as the unchanged lists. For instance, consider the mutation operator changes from:

$$\{(2,3,1),(1,2,3),(2,3,1)\}$$

to:

$$\{(2,3,1),(1,3,2),(2,3,1)\}$$

where the positions of jobs on the second machine have been swaped. The operation lists would be

$$m_1 : O_{21} \prec O_{31} \prec O_{12}$$

$$m_2 : O_{11} \prec O_{33} \prec O_{22}$$

$$m_3 : O_{23} \prec O_{32} \prec O_{13}$$

where operations O_{33} and O_{23} should be scheduled before operations O_{32} and O_{22}, which is a deadlock. To remove the deadlock, the changed list for the second machine is considered as a preference list, so it has the flexibility to schedule the operations with lower priority first. So the modified list is modified again and returns to its previous state, which results to the same feasible schedule. Using this methodology, all the possible chromosomes construct feasible solutions.

It should be mentioned here that to make all the schedules active each operation will be assigned to the first appropriate interval on the corresponding machine for both exact operation sequences and preference lists. This methodology ia able to modify the exact operation sequence. Consider another mutation operation on the first sample:

$$\{(2,3,1),(1,2,3),(2,3,1)\}$$

to produce chromosome:

$$\{(1,2,3),(1,2,3),(2,3,1)\}$$

where the first job gains the first position of the first machine operation list. The operation lists for this chromosome would be:

$$m_1 : O_{12} \prec O_{21} \prec O_{31}$$

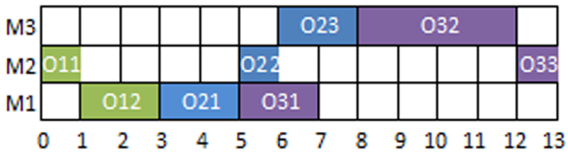
$$m_2 : O_{11} \prec O_{22} \prec O_{33}$$

$$m_3 : O_{23} \prec O_{32} \prec O_{13}$$

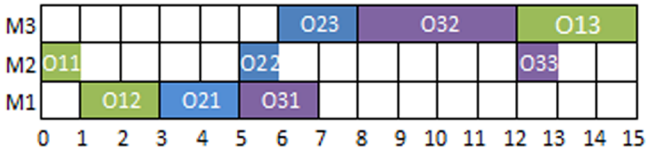
which is a feasible schedule. After assigning all the operations except the last one of the third machine, O_{13}, the partial schedule would be as what is represented in Figure 3.2a. Considering the exactness of operation sequence, operation O_{13} should be scheduled after operation O_{32}, which is no longer an active schedule. Figure 3.2b shows the complete schedule constructed using this way which is not active. If the decoding procedure considers the first appropriate interval for operation scheduling for both exact operation sequence and preference lists, operation O_{13} also can be scheduled before operation O_{23}. The result of this procedure is presented in Figure 3.2c. Therefore, since in the proposed representation the first appropriate interval is considered for operation scheduling for exact and preference lists, the final schedule is active.

3.2.4.2 Genetic Operators

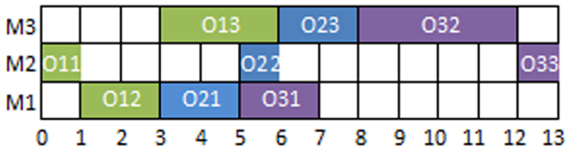
Based on the proposed representation, appropriate genetic operators are used. The crossover operator is applied on different operation lists independently. For each machine operation list, the lists of the children could be the same as their parents'



(a) Partial schedule of all operations except O_{13}



(b) Complete non-active schedule



(c) Complete active schedule

Figure 3.2: Constructing active schedules for chromosome $\{(1,2,3),(1,2,3),(2,3,1)\}$

list, or a crossover of them by the equal chance of 50%. In other words, for each machine, a random number between 0 and 1 is generated. If a generated random number is less than 0.5, the parents' operation lists will be assigned to the children as exact operation sequence. Otherwise, the crossover of the parents lists will be assigned to the children as a preference list.

For the crossover operator on a machine operation list, Linear Order Crossover (LOX) is used. LOX is based on Order Crossover (OX) proposed by Goldberg *et al.* [68]. The main characteristic of OX is that it tries to reserve the relative sequence. Since OX is proposed for the Travelling Salesman Problem (TSP), it is considered for circular chromosomes. LOX proposed by Falkenauer and Bouffoix [53] is a linear version of OX.

The mutation operator selects a machine operation list randomly, then in that list, it selects an operation randomly again. The selected operation is removed and reassigned to the first position of that list. The rates of crossover and mutation

are managed using two parameters; $CrossoverProbability$ and $MutationProbability$, respectively.

3.2.4.3 Local Search

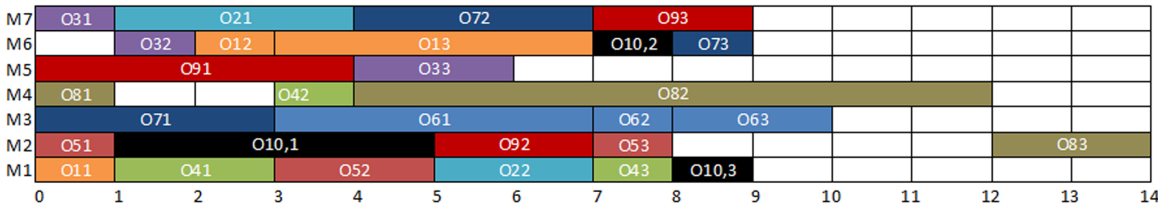
The proposed Local Search is also based on Machine Operation Lists. It selects the bottleneck job or a random one, and removes all the operations from all operation lists. Then, it reassigns removed operations one by one based on their operation ID starting from the first operation of the selected job. To reassign each operation, all possible positions for that operation are considered in order to select the best one. However, after assigning the first operations, they will not be rescheduled again.

Figure 3.3 shows all the steps of the proposed Local Search method. A sample schedule is shown in Figure 3.3a. As it can be seen, the bottleneck operation is O_{83} which belongs to job J_8. All the operations of the bottleneck job are removed from the schedule which is presented in Figure 3.3b. Then, the removed operations will be reassigned one by one to complete the schedule. As shown in Figure 3.3e, the modified schedule has less makespan than the initial one.

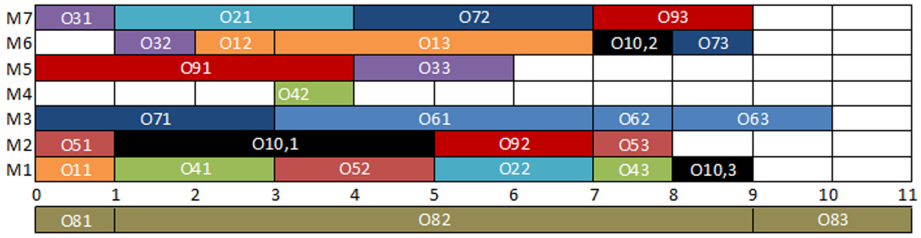
The proposed LS is very fast, since all the operations are assigning independently. However, the LS is applied only on the best portion of chromosomes for each generation, which is denoted by the $TopBest$ parameter.

3.2.4.4 MA Framework

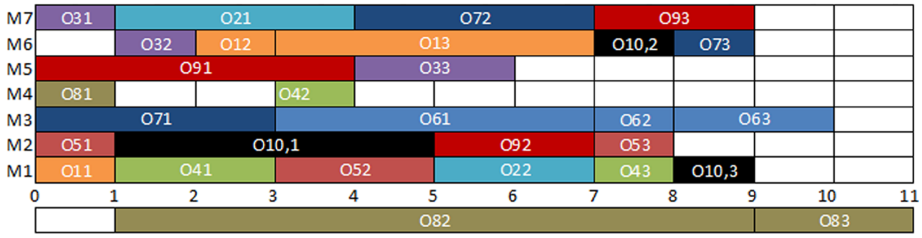
Figure 3.4 shows the framework of the proposed MA which is as follows. First, it generates the initial population P_0 of random individuals. The size of the population is denoted by $PopSize$ parameter. $IterationNo$ parameter determines the number of iterations for the proposed MA. Since the fitness function for job shop scheduling is their makespan, all the chromosomes are sorted based on their makespan; the smaller, the better. To construct a new generation P_{t+1}, crossover and mutation operators are



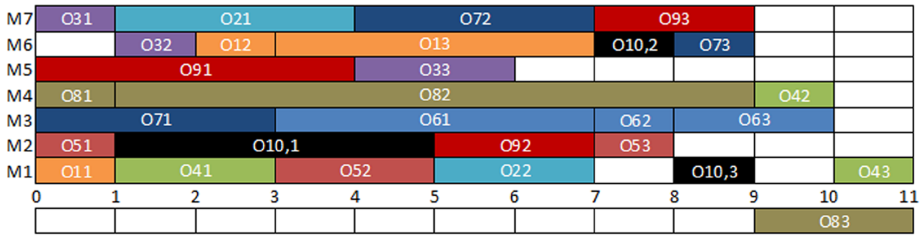
(a) A sample schedule



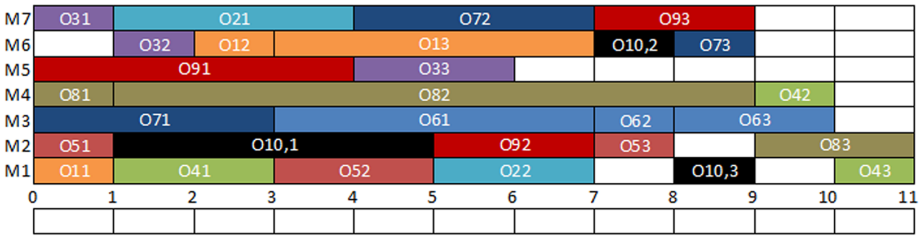
(b) Schedule after removing all the operations of bottleneck job



(c) First operation is reassigned



(d) Second Operation is reassigned



(e) New schedule after local search

Figure 3.3: Steps of proposed Local Search heuristic

PROCEDURE: MA Framework
INPUT: Test Problems and Algorithm Parameters
OUTPUT: Optimal or Near-Optimal Schedules

BEGIN
  Generate the initial population P_0
  FOR ($IterationNo$)
    Evaluate current population P_t and sort it
    Apply crossover and mutation to generate P_{t+1}
    Apply local search method on the $BestTop$
  END
  Output the best individual
END

Figure 3.4: MA Framework

Table 3.2: Parameters of proposed algorithm

Proposed Method Parameters	Value
$PopSize$	1000
$IterationNo$	200
$TopBest$	30%
$CrossoverProbability$	90%
$MutationProbability$	5%

applied on the current population P_t. Finally, the proposed Local Search method will be applied on the best portion of the new generation to find better chromosomes.

To show the efficiency of the proposed chromosome representation, the proposed Genetic Algorithm itself is applied on some benchmark problems without using the LS heuristic.

3.2.5 Results

The proposed algorithm is implemented and tested using the $JAVA$ programming language version 1.6.0.18 on a system with $Intel(R)\ Core(TM)2Quad\ 2.50GHz\ CPU$ and $8.00GB\ RAM$. Table 3.2 presents all the parameters of the proposed algorithm and their corresponding values.

Table 3.3: Comparing best makespan found by different chromosome representations [159] and proposed MOL

Problem	Size	Operation based	Job based	Preference list based	Priority rule based	Proposed MOL
la16	10×10	1523	997	1001	1225	**973**
la36	15×15	2168	1562	1486	2023	**1315**
la37	15×15	2431	1632	1581	2086	**1467**
la38	15×15	2153	1494	1374	1953	**1258**
la39	15×15	2310	1631	1353	1879	**1262**
la40	15×15	2146	1614	1374	1844	**1248**

The best benchmark for classical job shop scheduling is presented by Lawrence [113]. His benchmark consists of 40 different test problems of different size ranging from 10 jobs and 5 machines to 30 jobs and 10 machines. The expriments show the results of 10 independent runs on each test problem.

Table 3.4 shows the results of both proposed GA without local search and proposed MA on Lawrence [113] benchmark. The results show that the final result of the proposed MA is better than the proposed GA results for all the 40 test problems. Moreover, comparing the standard deviation of the different runs shows that the Local Search heuristic not only improves the results but also helps the GA to converge faster. The proposed GA finds the best solution for 25 test problems out of 40, and the proposed MA which outperforms the proposed GA without LS finds the best schedule for 27 test problems.

To show the efficiency of the proposed chromosome representation, the results of the proposed GA without using Local Search method are compared with the results of four different representations presented by Ponnambalam *et al.* [159]. The best makespan which are bolded in Table 3.3 all belongs to the proposed Machine Operation List representation.

To show the efficiency of the proposed MA, the results are compared with those of another memetic algorithm recently proposed by Hasan *et al.* [87]. The best known solution and Hasan *et al.* [87] results are presented in Table 3.4 as well. As these

tables show, the proposed GA outperforms the GA proposed by Hasan *et al.* [87] for all 40 test problems. Hasan *et al.* [87] GA find the best schedule for only 15 test problems, while the proposed GA find the best schedule for 25 problems out of 40.

While both proposed MA and Hasan *et al.* [87] MA found best solution for 27 problems, the proposed method outperforms another one, regarding the standard deviation and near optimal solutions for other test problems. Moreover, they used bigger values for their $PopSize$ and $IterationNo$ parameters, 2500 and 1000, respectively, instead of 1000 and 200 which is used in the proposed Memetic Algorithm.

Table 3.4: Results on Lawrence [113] test problems (la01-la40)

Problem	Algorithm	Best	Average	SD	Median	Worst
la01	Hasan et al. [87] GA	667	676.07	6.31	678	688
	Hasan et al. [87] MA	**666**	667.60	2.90	667	678
10×5	Proposed GA	**666**	666.00	0.00	666	666
666	Proposed MA	**666**	666.00	0.00	666	666
la02	Hasan et al. [87] GA	**655**	658.43	5.06	655	666
	Hasan et al. [87] MA	**655**	656.27	3.39	655	666
10×5	Proposed GA	**655**	660.00	5.62	656	667
655	Proposed MA	**655**	655.00	0.00	655	655
la03	Hasan et al. [87] GA	617	624.70	9.41	619.5	642
	Hasan et al. [87] MA	**597**	613.93	7.63	617	619
10×5	Proposed GA	599	608.00	5.60	607	614
597	Proposed MA	**597**	597.00	0.00	597	597
la04	Hasan et al. [87] GA	606	607.27	2.30	606	613
	Hasan et al. [87] MA	**590**	593.33	2.44	595	595
10×5	Proposed GA	**590**	594.60	2.72	595	598
590	Proposed MA	**590**	590.60	1.26	590	593

Table 3.4 – Continued on next page

Table 3.4: **(continued from previous page)**

Problem	Algorithm	Best	Average	SD	Median	Worst
la05	Hasan et al. [87] GA	**593**	593.00	0.00	593	593
	Hasan et al. [87] MA	**593**	593.00	0.00	593	593
10×5	Proposed GA	**593**	593.00	0.00	593	593
593	Proposed MA	**593**	593.00	0.00	593	593
la06	Hasan et al. [87] GA	**926**	926.00	0.00	926	926
	Hasan et al. [87] MA	**926**	926.00	0.00	926	926
15×5	Proposed GA	**926**	926.00	0.00	926	926
926	Proposed MA	**926**	926.00	0.00	926	926
la07	Hasan et al. [87] GA	**890**	896.50	4.27	897	906
	Hasan et al. [87] MA	**890**	890.00	0.00	890	890
15×5	Proposed GA	**890**	890.00	0.00	890	890
890	Proposed MA	**890**	890.00	0.00	890	890
la08	Hasan et al. [87] GA	**863**	863.00	0.00	863	863
	Hasan et al. [87] MA	**863**	863.00	0.00	863	863
15×5	Proposed GA	**863**	863.00	0.00	863	863
863	Proposed MA	**863**	863.00	0.00	863	863
la09	Hasan et al. [87] GA	**951**	951.00	0.00	951	951
	Hasan et al. [87] MA	**951**	951.00	0.00	951	951
15×5	Proposed GA	**951**	951.00	0.00	951	951
951	Proposed MA	**951**	951.00	0.00	951	951
la10	Hasan et al. [87] GA	**958**	958.00	0.00	958	958
	Hasan et al. [87] MA	**958**	958.00	0.00	958	958
15×5	Proposed GA	**958**	958.00	0.00	958	958
958	Proposed MA	**958**	958.00	0.00	958	958

Table 3.4 – Continued on next page

Table 3.4: **(continued from previous page)**

Problem	Algorithm	Best	Average	SD	Median	Worst
la11	Hasan et al. [87] GA	**1222**	1222.00	0.00	1222	1222
	Hasan et al. [87] MA	**1222**	1222.00	0.00	1222	1222
20×5	Proposed GA	**1222**	1222.00	0.00	1222	1222
1222	Proposed MA	**1222**	1222.00	0.00	1222	1222
la12	Hasan et al. [87] GA	**1039**	1039.00	0.00	1039	1039
	Hasan et al. [87] MA	**1039**	1039.00	0.00	1039	1039
20×5	Proposed GA	**1039**	1039.00	0.00	1039	1039
1039	Proposed MA	**1039**	1039.00	0.00	1039	1039
la13	Hasan et al. [87] GA	**1150**	1150.00	0.00	1150	1150
	Hasan et al. [87] MA	**1150**	1150.00	0.00	1150	1150
20×5	Proposed GA	**1150**	1150.00	0.00	1150	1150
1150	Proposed MA	**1150**	1150.00	0.00	1150	1150
la14	Hasan et al. [87] GA	**1292**	1292.00	0.00	1292	1292
	Hasan et al. [87] MA	**1292**	1292.00	0.00	1292	1292
20×5	Proposed GA	**1292**	1292.00	0.00	1292	1292
1292	Proposed MA	**1292**	1292.00	0.00	1292	1292
la15	Hasan et al. [87] GA	**1207**	1241.40	17.64	1243	1265
	Hasan et al. [87] MA	**1207**	1207.13	0.52	1207	1209
20×5	Proposed GA	**1207**	1207.00	0.00	1207	1207
1207	Proposed MA	**1207**	1207.00	0.00	1207	1207
la16	Hasan et al. [87] GA	994	1002.83	10.05	994	1028
	Hasan et al. [87] MA	**945**	968.27	15.46	979	982
10×10	Proposed GA	973	980.50	2.92	982	982
945	Proposed MA	**945**	947.40	4.58	945	956

Table 3.4 – Continued on next page

Table 3.4: **(continued from previous page)**

Problem	Algorithm	Best	Average	SD	Median	Worst
la17	Hasan et al. [87] GA	794	822.50	10.18	820	839
	Hasan et al. [87] MA	**784**	788.93	4.18	792	793
10 × 10	Proposed GA	**784**	789.70	5.64	789.5	797
784	Proposed MA	**784**	784.00	0.00	784	784
la18	Hasan et al. [87] GA	861	895.33	14.56	901	912
	Hasan et al. [87] MA	**848**	859.27	4.57	861	861
10 × 10	Proposed GA	**848**	858.60	8.30	861	875
848	Proposed MA	**848**	848.00	0.00	848	848
la19	Hasan et al. [87] GA	890	915.73	9.04	914	929
	Hasan et al. [87] MA	**842**	855.47	7.76	855	869
10 × 10	Proposed GA	**842**	852.80	9.30	860	860
842	Proposed MA	**842**	844.20	2.90	842	848
la20	Hasan et al. [87] GA	967	979.67	12.62	977	1016
	Hasan et al. [87] MA	**907**	910.00	2.54	912	912
10 × 10	Proposed GA	914	918.80	2.53	920	920
902	Proposed MA	**907**	909.90	2.02	911	912
la21	Hasan et al. [87] GA	1098	1145.07	20.07	1145	1185
	Hasan et al. [87] MA	1079	1097.60	12.48	1096	1124
15 × 10	Proposed GA	1087	1111.90	15.10	1112.5	1137
1046	Proposed MA	**1057**	1062.20	4.83	1061.5	1074
la22	Hasan et al. [87] GA	986	1027.50	17.59	1031	1055
	Hasan et al. [87] MA	960	981.00	13.58	979	1000
15 × 10	Proposed GA	946	957.30	8.94	958	968
927	Proposed MA	**935**	937.80	3.39	936	943

Table 3.4 – Continued on next page

Table 3.4: **(continued from previous page)**

Problem	Algorithm	Best	Average	SD	Median	Worst
la23	Hasan et al. [87] GA	1043	1077.13	17.60	1080	1108
	Hasan et al. [87] MA	**1032**	1032.00	0.00	1032	1032
15 × 10	Proposed GA	**1032**	1032.40	1.26	1032	1036
1032	Proposed MA	**1032**	1032.00	0.00	1032	1032
la24	Hasan et al. [87] GA	984	1023.70	24.94	1020.5	1080
	Hasan et al. [87] MA	959	996.40	16.21	1003	1011
15 × 10	Proposed GA	971	979.30	6.70	982.5	986
935	Proposed MA	**944**	954.90	7.26	955	967
la25	Hasan et al. [87] GA	1077	1116.97	19.06	1114	1154
	Hasan et al. [87] MA	991	1016.67	19.64	1013	1076
15 × 10	Proposed GA	1020	1029.00	9.39	1024.5	1049
977	Proposed MA	**983**	987.30	3.33	986	991
la26	Hasan et al. [87] GA	1295	1324.13	18.52	1322.5	1369
	Hasan et al. [87] MA	**1218**	1234.27	16.12	1228	1266
20 × 10	Proposed GA	**1218**	1238.50	14.91	1240	1267
1218	Proposed MA	**1218**	1218.60	1.58	1218	1223
la27	Hasan et al. [87] GA	1328	1356.87	17.70	1355	1389
	Hasan et al. [87] MA	1286	1306.33	13.17	1301	1333
20 × 10	Proposed GA	1299	1302.00	4.50	1299.5	1312
1235	Proposed MA	**1269**	1278.30	12.14	1272.5	1299
la28	Hasan et al. [87] GA	1328	1356.87	17.70	1355	1389
	Hasan et al. [87] MA	1286	1306.33	13.17	1301	1333
20 × 10	Proposed GA	1251	1290.10	18.66	1300	1306
1216	Proposed MA	**1223**	1235.20	8.93	1240	1244

Table 3.4 – Continued on next page

Table 3.4: **(continued from previous page)**

Problem	Algorithm	Best	Average	SD	Median	Worst
la29	Hasan et al. [87] GA	1265	1294.07	20.01	1291	1334
	Hasan et al. [87] MA	1221	1240.47	6.98	1240.5	1249
20×10	Proposed GA	1259	1270.20	9.47	1274	1289
1157	Proposed MA	**1191**	1214.80	12.54	1216.5	1233
la30	Hasan et al. [87] GA	1377	1465.43	35.82	1476	1555
	Hasan et al. [87] MA	**1355**	1362.33	8.23	1359.5	1380
20×10	Proposed GA	**1355**	1358.00	5.10	1355	1369
1355	Proposed MA	**1355**	1355.00	0.00	1355	1355
la31	Hasan et al. [87] GA	**1784**	1784.00	0.00	1784	1784
	Hasan et al. [87] MA	**1784**	1784.00	0.00	1784	1784
30×10	Proposed GA	**1784**	1784.00	0.00	1784	1784
1784	Proposed MA	**1784**	1784.00	0.00	1784	1784
la32	Hasan et al. [87] GA	**1850**	1869.60	19.79	1872	1916
	Hasan et al. [87] MA	**1850**	1850.00	0.00	1850	1850
30×10	Proposed GA	**1850**	1850.00	0.00	1850	1850
1850	Proposed MA	**1850**	1850.00	0.00	1850	1850
la33	Hasan et al. [87] GA	**1719**	1729.00	11.22	1725	1765
	Hasan et al. [87] MA	**1719**	1719.00	0.00	1719	1719
30×10	Proposed GA	**1719**	1719.00	0.00	1719	1719
1719	Proposed MA	**1719**	1719.00	0.00	1719	1719
la34	Hasan et al. [87] GA	1748	1770.90	13.31	1770.5	1814
	Hasan et al. [87] MA	**1721**	1721.00	0.00	1721	1721
30×10	Proposed GA	**1721**	1721.20	0.63	1721	1723
1721	Proposed MA	**1721**	1721.00	0.00	1721	1721

Table 3.4 – Continued on next page

Table 3.4: **(continued from previous page)**

Problem	Algorithm	Best	Average	SD	Median	Worst
la35	Hasan et al. [87] GA	1898	1947.20	30.94	1942.5	2018
	Hasan et al. [87] MA	**1888**	1888.00	0.00	1888	1888
30×10	Proposed GA	**1888**	1888.00	0.00	1888	1888
1888	Proposed MA	**1888**	1888.00	0.00	1888	1888
la36	Hasan et al. [87] GA	1388	1432.60	29.14	1432	1497
	Hasan et al. [87] MA	1307	1328.67	11.59	1327	1346
15×15	Proposed GA	1315	1337.50	15.86	1334.5	1357
1268	Proposed MA	**1281**	1293.20	7.38	1297	1301
la37	Hasan et al. [87] GA	1561	1606.23	25.27	1605.5	1656
	Hasan et al. [87] MA	1442	1473.60	11.51	1479	1487
15×15	Proposed GA	1467	1480.10	10.32	1474.5	1496
1397	Proposed MA	**1429**	1444.60	8.49	1447.5	1452
la38	Hasan et al. [87] GA	1367	1412.53	17.07	1415	1439
	Hasan et al. [87] MA	1266	1309.13	14.52	1314	1329
15×15	Proposed GA	1258	1271.80	8.56	1271.5	1288
1196	Proposed MA	**1208**	1222.90	9.96	1222.5	1242
la39	Hasan et al. [87] GA	1338	1368.47	18.29	1367	1413
	Hasan et al. [87] MA	1252	1282.60	16.62	1277	1301
15×15	Proposed GA	1262	1281.90	19.33	1278.5	1315
1233	Proposed MA	**1248**	1250.40	0.97	1251	1251
la40	Hasan et al. [87] GA	1357	1366.30	13.57	1360	1407
	Hasan et al. [87] MA	1252	1279.60	17.84	1289	1303
15×15	Proposed GA	1248	1265.40	10.09	1267.5	1278
1222	Proposed MA	**1234**	1246.90	6.84	1246	1256

3.2.6 Conclusions

Job shop scheduling is an open problem which is well-known in the area of optimization problems, especially in manufacturing systems. The proposed Memetic Algorithm to solve classical job shop scheduling is based on a new chromosome representation. The proposed chromosome representation which is called Machine Operation Lists outperforms the most well-known existing representations.

Moreover, the results of proposed MA show that the Local Search method is also efficient when compared with another memetic algorithm recently presented by Hasan *et al.* [87]. The proposed GA without LS and the proposed MA outperform their GA and MA, respectively.

In this article, the proposed method is designed for only classical job shop scheduling, while the idea is applicable for various types of job scheduling problems such as flexible job shop scheduling and multi-objective one. However, the proposed MA algorithm and especially the proposed chromosome representation can be generalized to consider almost all job scheduling problems.

Acknowledgments

This work has been made possible in part by an NSERC Discovery grant and the National Science Foundation.

3.3 A Memetic Algorithm for Job Shop Scheduling Using A Critical-Path-Based Local Search Heuristic

Abstract. In this article, a new memetic algorithm has been proposed to solve Job Shop Scheduling Problems. The proposed method is a genetic-algorithm-based approach combined with a local search heuristic. The proposed local search heuristic is based on critical operations. It removes the critical operations and reassigns them to a new position to improve the fitness value of the schedule. Moreover, in this article, a new fitness function is introduced for job shop scheduling problems. The new fitness function called priority-based fitness function is defined in three priority levels to improve the selection procedure. To show the generality of our proposed method, we apply it to three different types of job scheduling problems including classical, flexible and multi-objective flexible job shop scheduling problems. The experiment results show the efficiency of the proposed fitness function. In addition, the results shows that incorporating local search not only offers better solutions but also improves the convergence rate. Compared to the state-of-the-art algorithms, the proposed method outperforms the existing methods in classical job shop scheduling problems and offers competitive solutions in other types of scheduling problems.

3.3.1 Introduction

Job shop scheduling is the process of scheduling various jobs between different machines in a manufacturing system. In the last decades, a lot of work has been done to

control a job shop system. There are different types of job shop scheduling such as classical and flexible. Each job shop includes different jobs such that each job has a fixed sequence of operations to be processed in order. The number of operations for different jobs could also be different. Each operation has to be processed only on one machine, while different operations of a job could be processed on different machines.

In the classical Job Shop Scheduling Problem (JSSP), each operation can be processed only on a specific machine. It means that there is no machine selection flexibility for operations in JSSP. So, JSSP is the process of determining the sequence of operations of different jobs which have to be processed on the same machine. In other words, JSSP determines an ordered list of operations for each machine. Unlike JSSP, in Flexible Job Shop Scheduling (FJSS), there is the flexibility for each operation to be processed on different machines. One of the first research work which considered flexibility in job shop scheduling was published by Bruker and Schlie [25]. If all the operations have the flexibility of being processed on all available machines, then the problem is called Total Flexible Job Shop Scheduling (T-FJSS). Otherwise, it is called Partial Flexible Job Shop Scheduling (P-FJSS). Considering the computational cost and search space, Zhang *et al.* [221] mentioned that the P-FJSS is more complex than T-FJSS. Few researchers also considered the uncertainty for FJSS such as stochastic processing time, and called problem as Stochastic Flexible Job Shop (SFJS). Mahdavi *et al.* [126] proposed a simulation-based decision support system to control the stochastic job shop system.

However, it is clear that FJSS is more complex than JSSP; in JSSP there is only the operation sequencing problem for each machine, while in FJSS, it is combined with the machine selection problem as well. Some researchers considered both machine selection and operation sequencing as one problem, such as Hurink *et al.* [97], while others solved them as separate problems such as Brandimarte [22].

Open shop scheduling is something different than job shop scheduling. The main difference is that in open shop scheduling operations of a job are sequence independent, while in job shop scheduling the sequence of operations for each job is fixed. Naderi *et al.* [141] proposed a memetic algorithm for open shop scheduling.

The main goal of job shop scheduling problems is to utilize the manufacturing system. Each manufacturing system has its own objective. In most cases, the objective of the scheduling is to minimize the maximum completion time, also called *makespan*. Moreover, there are some systems looking for various parameters as their objectives, which are called Multi-Objective Flexible Job Shop Scheduling (MO-FJSS). For instance, the objective of a system can be both minimizing maximum machine workload and maximizing machine utilization. To deal with such systems, some researchers such as Mahdavi *et al.* [126] used generalized response surface methodology, in which one response is the goal and others are considered as constraint variables.

In this article, a memetic algorithm is proposed to solve scheduling problems. The proposed algorithm is applied to different types of job shop scheduling problems including JSSP, FJSS and MO-FJSS problems. It should be mentioned that stochastic parameters are beyond the scope of this work and not considered to evaluating our method in this article, but these can be considered in future work. The outline of this paper starts with the related work in Subsection 2, followed by a more detailed treatment of the problem definition in Subsection 3. Subsection 4 describes the proposed memetic algorithm in details. The evaluation results of our proposed algorithm are discussed in Subsection 5. Finally, conclusions are presented in Subsection 6.

3.3.2 Related Works

Job shop scheduling is well known in different fields, and gained popularity mainly because of its application in various manufacturing systems. It is proved by Garey *et al.* [65] that job shop scheduling for systems with more than two machines are

NP-complete problems which means there is no exact algorithm able to find the best schedule for all types of problems.

There are various types of algorithms proposed to solve job shop scheduling problems, including heuristics, and meta-heuristics. Bottleneck shifting procedure used by Huang *et al.* [96] and tree search method presented by Asano *et al.* [9] are instances of heuristic approaches. Meta-heuristic approaches include Simulated Annealing (SA), Tabu Search (TS), Ant Colony Optimization (ACO), Genetic Algorithm (GA), hybrid GA, and so on. Kolonko [108] was one of the first researchers to apply Simulated Annealing (SA) on job shop scheduling. A combination of Tabu Search and Bottleneck shifting is used by Pezzella *et al.* [157]. Wang *et al.* [202] proposed an ACO approach for JSSP. A combination of ACO and TS is also proposed by Huang *et al.* [95].

One of the most widely used algorithms in job shop scheduling is GA either as a single procedure or combined with a Local Search (LS) heuristic. The application of GA to job shop scheduling was first introduced by Lawrence [112]. Four more different GAs were also presented by Chen *et al.* [30], Mattfeld *et al.* [130], Pezzella *et al.* [158] and Zhang *et al.* [221] as a single efficient procedure to solve the job shop scheduling problem.

Most of the recent work in job shop scheduling hybridized GAs with LS algorithms to improve the final results as well as to help the convergence rate. Moscato [139] defined Memetic Algorithm (MA) as a population-based global search method combined with a local search heuristic to be used to solve an optimization problem. Ong *et al.* [149] considered population-based search as exploration to find the best region in the search space, and local search as exploitation to find the best individual in a neighborhood. Based on this definition, all the hybridizations of GAs or other Evolutionary Algorithms (EAs) with LS heuristics can be considered as MAs. Park

et al. [154], Chiang *et al.* [33], Gao *et al.* [62, 63], and Caumond *et al.* [28] proposed different MAs with different heuristics.

The MA proposed by Yang *et al.* [213] is a combination of Clonal Selection and SA. A combination of GA and Tabu Search was also proposed by Ombuki [147]. Kacem *et al.* [103] combined an EA with a localization approach. They also proposed a combination of EA with fuzzy logic [104]. Hassan *et al.* [84, 85, 86] proposed different GAs combined with different priority rules. Recently, Hasan *et al.* [87] proposed three different MAs which are combinations of GAs with three different priority rules. Among these three, they claimed that the most effective priority rule is joint of gap reduction and restricted swapping. Gap reduction is the rule of moving operations to the appropriate existing gaps in which the corresponding machine is idle. Restricted swapping rule tries to swap the adjacent operations on a machine to decrease the makespan.

Moreover, there are more different algorithms applied in job shop scheduling such as Neural Network algorithm used by Fonseca *et al.* [58], and Artificial Immune Algorithm (AIA) proposed by Bagheri *et al.* [10] and Mobini *et al.* [137]. A simulation modeling algorithm is proposed by Xing *et al.* [208]. Mahdavi *et al.* [126] combined the simulation modeling with a Decision Support System (DSS) to improve the final results.

Recently, we proposed a MA to deal with classical JSSPs, in which we introduced a new chromosome representation called Machine Operation List (MOL) [164]. Since MOL is based on the operations on their applicable machine, it is appropriate only for classical JSSP. Therefore, it has not been used in this article. The local search of our previous work deals with critical jobs and their operations, while here a new concept is introduced which is selected critical operations.

3.3.3 Problem Definitions

In all types of job shop scheduling problems, there are a set of machines and a set of jobs to be processed on those machines. Machines are denoted by m_k, where k refers to the index of each machine which is between 1 to M, the number of machines. Jobs are denoted by J_i , where i refers to the job index ranging from 1 to N, the total jobs number. Each job consists of a number of operations, denoted by NO_i. The operations are denoted by O_{ij}, meaning the j^{th} operation of the i^{th} job. The processing time of each operation is known and denoted by $PT(O_{ij}, m_k)$, meaning the processing time of operation O_{ij} on k^{th} machine. It should be mentioned that the moving time for jobs between machines and the set up time for each machine are assumed to be negligible. Table 3.5 shows the 4^{th} benchmark of Xing *et al.* paper [208] which is a sample data for the T-FJSS. In this case, there are 10 jobs with the same number of operations as 3 operations per job to be processed on 10 different machines. The reason for representing this sample is that our proposed method outperforms Xing *et al.* [208] approach on this test problem. The improved result are shown in Figure 3.12.

The rules of the job shop scheduling are as follows: The sequence of operation for each job is fixed. For example, the third operation of a job cannot be processed until the second operation has been done. However, the operations of different jobs are independent. At any given time, only one operation of a job can be processed on one and only one machine. It is considered that all the jobs and all the machines are available at the starting time.

It should be mentioned here that there are two types of operation sequences: job-operation sequence and machine-operation sequence. Job-operation sequence is the fixed sequence of operations for each job. So the job-preceding operation of an operation is its preceding operation in job-operation sequence. Machine-operation sequence is the sequence of operations which are processed on the same machine.

Table 3.5: 4^{th} FJSS benchmark used by Xing et al. [208].

Jobs	Operations	Machines									
		m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
	O_{11}	1	4	6	9	3	5	2	8	9	5
J_1	O_{12}	4	1	1	3	4	8	10	4	11	4
	O_{13}	3	2	5	1	5	6	9	5	10	3
	O_{21}	2	10	4	5	9	8	4	15	8	4
J_2	O_{22}	4	8	7	1	9	6	1	10	7	1
	O_{23}	6	11	2	7	5	3	5	14	9	2
	O_{31}	8	5	8	9	4	3	5	3	8	1
J_3	O_{32}	9	3	6	1	2	6	4	1	7	2
	O_{33}	7	1	8	5	4	9	1	2	3	4
	O_{41}	5	10	6	4	9	5	1	7	1	6
J_4	O_{42}	4	2	3	8	7	4	6	9	8	4
	O_{43}	7	3	12	1	6	5	8	3	5	2
	O_{51}	7	10	4	5	6	3	5	15	2	6
J_5	O_{52}	5	6	3	9	8	2	8	6	1	7
	O_{53}	6	1	4	1	10	4	3	11	13	9
	O_{61}	8	9	10	8	4	2	7	8	3	10
J_6	O_{62}	7	3	12	5	4	3	6	9	2	15
	O_{63}	4	7	3	6	3	4	1	5	1	11
	O_{71}	1	7	8	3	4	9	4	13	10	7
J_7	O_{72}	3	8	1	2	3	6	11	2	13	3
	O_{73}	5	4	2	1	2	1	8	14	5	7
	O_{81}	5	7	11	3	2	9	8	5	12	8
J_8	O_{82}	8	3	10	7	5	13	4	6	8	4
	O_{83}	6	2	13	5	4	3	5	7	9	5
	O_{91}	3	9	1	3	8	1	6	7	5	4
J_9	O_{92}	4	6	2	5	7	3	1	9	6	7
	O_{93}	8	5	4	8	6	1	2	3	10	12
	O_{101}	4	3	1	6	7	1	2	6	20	6
J_{10}	O_{102}	3	1	8	1	9	4	1	4	17	15
	O_{103}	9	2	4	2	3	5	2	4	10	23

The machine-preceding operation of an operation is its preceding operation which is processed on the same machine exactly before it.

3.3.4 Proposed Memetic Algorithm

The proposed MA is a combination of a GA and a LS heuristic, which is the main contribution of this article. The proposed GA is a simple GA with a sophisticated fitness function. In each generation after fitness evaluation the LS is applied to some selected individuals. The heuristic used in LS is moving critical operations to improve the fitness value. Moving critical operations introduced in the literature is restricted such that it only tries to find an idle time interval, while in our proposed method, rescheduling is used to find the best positions. Furthermore, the proposed method is a generic method applicable for various types of job shop scheduling such as JSSP, FJSS and MO-FJSS.

3.3.4.1 Chromosome Representation

In the literature, there are various chromosome representations proposed by different researchers. In this article, MSOS representation proposed by Zhang *et al.* [221] has been used, because it is efficient for coding and decoding procedures. This representation always generates feasible schedules so the repair mechanisms are not required. MSOS representation includes two parts: Machine Selection (MS) and Operation Sequence (OS).

Machine Selection determines the selected machine for each operation among its applicable machines. Hence it is a string of machines indices with the length of total number of operations. For example, if an operation has the flexibility to be processed on machine 2 and machine 4, the 1 for its corresponding MS entry means the 2^{nd} machine, and 2 means the 4^{th} one. Therefore, in JSSP where there is no flexibility for machine selection, the MS is always a string of 1s.

Operation Sequence is also a string of numbers determining the sequence of operations. In this string the operations are denoted by their job indices and the occurrence of job indices determines the index of each operation for each job. Consider the Operation Sequence

$$OS = [3, 2, 3, 1, 2, 2, 1]$$

which represents the following sequence of operations.

$$O_{31} \prec O_{21} \prec O_{32} \prec O_{11} \prec O_{22} \prec O_{23} \prec O_{12}$$

Since the operations are denoted by their job indices and the operation indices are determined automatically, all the variations of OS is thus feasible and there is no need for any repair mechanism.

In chromosome decoding, we aim to generate only active schedules to limit the search space. Active schedules are ones where there is no operation to be processed earlier without delaying another operation. By assigning the operations to the first appropriate intervals, the resulting schedule would be active. By this definition, an optimal schedule is either an active schedule or has an equivalent active schedule, so it can be reached by exploring only the active schedules. Active schedule concept is introduced by Croce *et al.* [43] and used by Becerra and Coello Coello [14] and Hasan *et al.* [87] as permissible left shift and gap reduction rule, respectively.

3.3.4.2 Genetic Operators

The characteristics of each GA are the initial generation, the crossover operation and the mutation method. The initial generation for MS is exactly the same as what Zhang *et al.* [221] used. It includes all the global selection, local selection and random selection with the same percentage as their suggestion. The OS part of the initial chromosomes are produced by Long Job First Selection and Random Selection by

the same percentage of 50%. In Long Job First Selection, for each OS entry, the job which has more remaining time to be done will be selected. The remaining time for each job is the sum of its not processed operations' working time. In JSSP where there is no MS, the OS of the first generation is initialized randomly.

In MSOS representation each part has its own genetic operators. For the MS part we use Two-Point Crossover used by Watanabe *et al.* [206] and Uniform Crossover presented by Gao *et al.* [63] by the same chance of 50%. Precedence preserving Order-based Crossover (POX) proposed by Lee *et al.* [114] is also used for OS part. The parents selection is as follows: one parent is randomly selected among the top best individuals, a proportion of the whole population denoted by parameter $TopBest$. The second parent is selected among the whole population randomly.

Mutation operation for MS entries is done by selecting another applicable machine which has the minimum processing time among others. Swapping of two OS entries is the mutation for the OS part. The rates of crossover and mutation are managed using two parameters $CrossoverProbability$ and $MutationProbability$, respectively.

3.3.4.3 Local Search

Our proposed Local Search heuristic is based on critical operations. Critical operations are ones for which any delay in their processing increases the makespan. The main idea behind the heuristic is to rearrange the critical operations on their corresponding machines. This heuristic first identifies critical operations, then it removes one of the critical operations and reassigns it again to find a better position on its corresponding machine. To save the time complexity of the proposed search method it is applied to a portion of the population, denoted by $TopBest$ parameter.

In order to identify the critical operations critical paths are defined. A sequence of critical operations starting from time zero to the completion time is called a critical path. A critical path can be found by tracing the schedule from its completion time

Table 3.6: A sample classical JSSP

	O_1	O_2	O_3	O_4
J_1	$m_3, 2$	$m_2, 3$	$m_4, 5$	$m_1, 3$
J_2	$m_4, 3$	$m_1, 4$	m_2,4	$m_3, 4$
J_3	$m_2, 3$	$m_4, 4$	$m_3, 6$	$m_1, 6$
J_4	$m_3, 6$	$m_1, 2$	$m_2, 5$	$m_4, 3$

to the beginning. To do so, the machines which are working until the last time of the schedule are considered as critical machines. Since the makespan is equal to their completion time any delay in processing their last operation will postpone the whole schedule's completion time. Thus the last operations of the critical machines are critical operations. The next critical operation can be one of the following:

1. The machine-preceding operation: if there is no idle time between the completion of machine-preceding operation and the start of the critical operation.

2. The job-preceding operation: if the finish time of the job-preceding operation equals to the start time of the critical operation.

3. Both case 1 and case 2.

This routine continues until time zero is reached.

To show critical operations finding procedure, a sample classical JSSP with four jobs and four machines is considered. The applicable machine for each operation with its corresponding processing time are presented in Table 3.6. For instance, the third operation of the first job, namely O_{13}, can be processed only on machine m_4 in 5 time units. Consider the following operation sequence

$$OS = [2, 3, 2, 1, 1, 4, 4, 3, 4, 2, 3, 1, 3, 1, 2, 4]$$

as a sample OS for the problem given above. Since the problem is a classical JSSP, there is no MS part for chromosome representation. The sequence of operations for

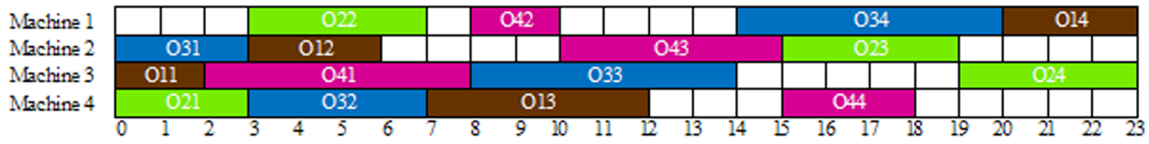


Figure 3.5: A sample schedule for sample JSSP to show critical paths

this sample OS is

$$O_{21} \prec O_{31} \prec O_{22} \prec O_{11} \prec O_{12} \prec O_{41} \prec O_{42} \prec O_{32} \prec$$

$$O_{43} \prec O_{23} \prec O_{33} \prec O_{13} \prec O_{34} \prec O_{14} \prec O_{24} \prec O_{44}$$

which is depicted in Figure 3.5. Both the first and the third machines are working until the schedule completion time. This means that their last operations, namely O_{14} and O_{24}, are critical operations. Consequently, there are at least two critical paths. The preceding critical operations for O_{14} and O_{24} are a machine-preceding operation and a job-preceding operation, namely O_{34} and O_{23}, respectively. The routine for finding the critical operations continues until time zero is reached. Thus the critical paths are as follows. Based on critical paths there are 9 critical operations in this schedule.

1. $O_{11} \rightarrow O_{41} \rightarrow O_{33} \rightarrow O_{34} \rightarrow O_{14}$
2. $O_{11} \rightarrow O_{41} \rightarrow O_{42} \rightarrow O_{43} \rightarrow O_{23} \rightarrow O_{24}$

In order to improve the search heuristic performance, only the critical operations which satisfy the two following criteria will be selected for reassigning:

1. The critical operation is assigned in the middle of the critical path, not at the first or last.
2. Its preceding and proceeding critical operations on the critical path are either its job-preceding and machine-proceeding or its machine-preceding and job-proceeding, respectively.

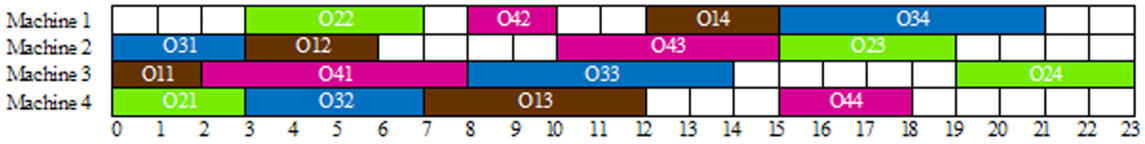


Figure 3.6: The sample schedule after repositioning O_{34}

Among those 9 critical operations only 5 operations satisfy these criteria which are O_{41}, O_{33}, O_{34}, O_{43} , and O_{23}. We call these operations Selected Critical Operations (SCOs). The SCOs will be repositioned in the corresponding OS between their job-preceding operation positions and their job-proceeding operation positions. For instance, the SCO O_{33} can be repositioned between the positions of O_{32} and O_{34}, while SCO O_{34} can be reassigned to the positions from O_{33}'s position to the end of OS. Therefore, the possible OSs for the new position of O_{34} are as follows:

$$\{O_{21}, O_{31}, ..., O_{33}, \mathbf{O_{34}}, O_{13}, O_{14}, O_{24}, O_{44}\}$$

$$\{O_{21}, O_{31}, ..., O_{33}, O_{13}, \mathbf{O_{34}}, O_{14}, O_{24}, O_{44}\}$$

$$\{O_{21}, O_{31}, ..., O_{33}, O_{13}, O_{14}, \mathbf{O_{34}}, O_{24}, O_{44}\}$$

$$\{O_{21}, O_{31}, ..., O_{33}, O_{13}, O_{14}, O_{24}, \mathbf{O_{34}}, O_{44}\}$$

$$\{O_{21}, O_{31}, ..., O_{33}, O_{13}, O_{14}, O_{24}, O_{44}, \mathbf{O_{34}}\}$$

Since operation O_{34} and O_{13} are from different jobs and have to be processed on different machines, their orders on the OS does not make any difference on the schedule. In this case, the first two OSs are equivalent, and their schedule is represented in Figure 3.5. Again since there is a similar situation for operations O_{24}, O_{34}, andO_{44}, their orders are not important. The last three OSs are equivalent too and their schedule is depicted in Figure 3.6.

Among the operations from O_{33} to end of OS, there is no operation of the third job and operation O_{14} is the only operation which has to be processed on the same

machine as operation O_{34}. Thus there is only two different schedules: one which considers O_{34} before O_{14}, and another which assigns O_{34} after O_{14}. Since the former is the current schedule, there is only one new OS which assigns O_{34} exactly after O_{14}.

$$\{O_{21}, O_{31}, ..., O_{33}, O_{13}, \mathbf{O_{14}}, \mathbf{O_{34}}, O_{24}, O_{44}\}$$

Since schedules are independent of the order of operations which are not from the same job and not to be processed on the same machine, the number of new OSs for reassigning a SCO is limited. This number for a SCO equals to the number of operations which have to be processed on the same machine as the SCO between the positions of its job-preceding operation and its job-proceeding operation on the corresponding OS. So using this routine, we have the following situations:

1. No new OS; that is there is no new position. So the LS cannot alter the schedule for this SCO. Consider operation O_{33} which is a SCO that have to be processed on machine m_3. In the range between operations O_{32} and O_{34}, there are the following operations:

   $$\{O_{43}, O_{23}, O_{33}, O_{13}, O_{14}\}$$

   None of these operations have to be processed on machine m_3. Thus, there is no new position for operation O_{33}.

2. One new OS; it means there is only one new position. So the LS generates a new individual by repositioning the SCO. This is the same routine that has been illustrated for operation O_{34}.

3. Two or more OSs; Since there is more than one OS, an OS is selected randomly and a new individual is generated.

It should be mentioned here that in FJSSs, since the SCOs can be processed on different machines, in order to find the new OSs all the positions for all the applicable

machine should be evaluated. Consider the case where operation O_{33} can be processed on machine m_2. In that range, there are two operations O_{43} and O_{23} that have to be processed on machine m_2. It means there are two new OSs when operation O_{33} is processed on machine m_2. In this case, not only the OS is changed, but also the MS is changed too.

Overall, our proposed LS works as follows for each individual: It starts with finding SCOs. Then for each SCO, it generates a new individual using the new OS if applicable. So there would be a number of new individuals which is equal to the number of SCOs for each individual. Between these new individuals, the best one which has the best fitness value will be added to the population. Since the LS is only applied to the best individuals of each generation its time complexity depends on both parameters $PopSize$ and $TopBest$. Therefore, the time complexity of the proposed LS is

$$TimeComplexity = IterationNo \times (TopBest \times PopSize) \times Average\,(SCO\#)$$

where $SCO\#$ denotes the number of SCOs for each individual.

In order to estimate the time complexity a population of 10,000 random individuals is selected and the average number of SCOs over that population is calculated, which is represented in Table 3.7. The maximum number of these average values is 12.38 which is the average value for problem la23. Therefore, the proposed local search time complexity for Lawrence [113] test problems would be:

$$TimeComplexity = 12.38 \times IterationNo \times (TopBest \times PopSize) \tag{3.1}$$

Table 3.7: The average numbers of SCOs for Lawrence [113] test problems

Problem	la1	la2	la3	la4	la5	la6	la7	la8	la9	la10
Average	0.26	3.55	3.27	3.28	0.64	0.74	2.84	3.46	2.08	1.42
Problem	la11	la12	la13	la14	la15	la16	la17	la18	la19	la20
Average	1.86	2.71	3.15	0.32	3.31	4.99	6.13	4.87	8.47	7.64
Problem	la21	la22	la23	la24	la25	la26	la27	la28	la29	la30
Average	7.59	4.68	12.38	8.24	8.63	8.20	9.01	9.42	7.08	5.98
Problem	la31	la32	la33	la34	la35	la36	la37	la38	la39	la40
Average	7.26	7.21	9.38	9.10	6.17	9.47	10.03	9.81	7.66	9.68

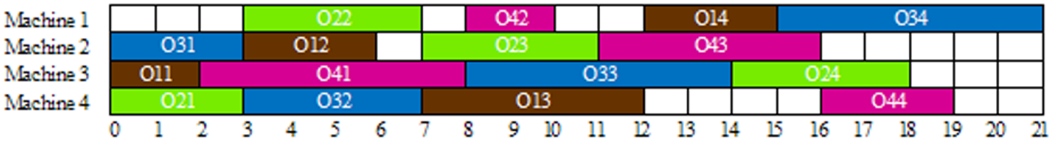


Figure 3.7: The sample schedule after repositioning O_{34}, followed by repositioning O_{23}

3.3.4.4 Priority-Based Fitness Function

In the case where two different individuals have the same fitness value, we do not resort to random selection. Instead, we evaluate different parameters of those individuals and select the one which is more likely to be able to direct the search process toward a better solution. In other words, it would be better to select the individual which is closer to a better solution. For example, consider the two sample schedules before and after repositioning of operation O_{34}. The makespans of both schedules are the same, but the numbers of critical machines are different. The former schedule has two critical machines and the latter has only one. So it is more likely for the latter to become a better solution just by one more repositioning. Therefore, it is better to select the latter schedule than the former if we have to select only one of them. By applying the LS to this schedule, we will have a schedule with a better makespan, represented in Figure 3.7.

Based on this procedure a new fitness function is introduced. Since the proposed fitness function calculates different parameters with different priority, it is called Priority-Based Fitness Function (PBFF). The first priority of the PBFF is

the main objective of the system. For example, if the optimization goal is to minimizing makespan, the first priority is thus the makespan; the lower, the better. In multi-objective job shop scheduling problems, the first priority of the fitness function is the same as the function of those objectives which is defined by the problem domain. Xing *et al.* [208], for instance, considered the following function as the multi-objective function with different weights:

$$F(s) = 0.5 \times F_1(s) + 0.2 \times F_2(s) + 0.3 \times F_3(s) \tag{3.2}$$

where $F(s)$ is the fitness value for schedule s; $F_1(s)$, $F_2(s)$, and $F_3(s)$ are makespan, total workload and maximum workload of schedule s, respectively.

If we do not consider the next priorities, the PBFF works as the traditional fitness function. The next priorities are therefore considered to direct the search process when a tie occurs. We consider the number of critical machines as the second priority which is very important to the effectiveness of the algorithm. The lower the number of critical machines, the greater probability to reach a new schedule with a better makespan. The third priority which is applicable only for flexible job shop scheduling is the maximum workload. If there are two schedules with the same makespan, the one which has lower maximum workload is more likely to be the direction to the optimal solution.

Therefore, each schedule has three fitness values with different priorities. The comparison between two different schedules is done based on the priorities of each value. First, the values of the highest priority are compared. If they are not equal, the schedule with the lower value is selected as the better schedule. Otherwise, the values for the next priority will be examined and it continues to the last priority. If the values of all priorities are equal, the better schedule is selected randomly.

PROCEDURE: MA Framework
INPUT: Test Problems and Algorithm Parameters
OUTPUT: Optimal or Near-Optimal Schedules

BEGIN
    Generate the initial population P_0
    FOR ($IterationNo$)
        Evaluate current population P_t and sort it
        Apply crossover and mutation to generate P_{t+1}
        Apply local search method on the $BestTop$
            individuls of both P_{t+1} to generate S_{t+1}
        Finalized P_{t+1} with the best individuals
    END
    Output the best individual
END

Figure 3.8: MA Framework

3.3.4.5 MA Framework

The framework of our proposed MA presented in Figure 3.8 is as follows. The proposed algorithm starts by the initial generation P_0 with a fixed number of individuals. Then, for a fixed number of iterations denoted by $IterationNo$, the genetic operators and local search method are applied to generate new generations.

In each iteration, all the individuals are sorted based on their fitness value calculated by PBFF. Then, genetic operators are applied to generate offsprings. The current population and the new one are denoted by P_t and P_{t+1}, respectively. At the next step, the LS is applied to the best individuals of the new population P_{t+1} to generate a small search population S_{t+1}. The best individuals are a top proportion of a sorted population. Finally, the new population P_{t+1} is finalized with its own best individuals and those resulting from the search heuristic.

It should be mentioned here that in order to show the efficiency of the GA itself the proposed method has been applied to some test problems without using the LS. In those cases, there is no search population.

3.3.5 Results

The proposed algorithm is implemented using *Java* version 1.6.0.18 on a system with $Intel(R)\ Core(TM)2\ Quad\ 2.50GHz\ CPU$ and $8.00GB\ RAM$. Since the proposed method was applied to various test problems with different sizes, we set the population size up to 1000 and run the experiments for up to 200 iterations. The bests individuals are defined as the top 10% of a sorted population. The crossover and mutation probabilities are set to 90% and 5%, respectively.

In order to show the efficiency of the proposed algorithm it has been applied to different types of job shop scheduling including JSSP, FJSS and MO-FJSS. Lawrence's benchmark [113] is considered as a dataset for JSSP, which includes 40 case problems ranging from 10 jobs on 5 machines to 30 jobs on 10 machines. The Brandimarte's dataset [22], denoted by *BRData*, has been used as a benchmark for FJSS, which consists of 10 different case problems ranging from 10 jobs on 6 machines to 20 jobs on 15 machines.

To show the effectiveness of the proposed Priority-Based Fitness Function two versions of the proposed method are applied to the test problems, one with the traditional fitness function and another with the PBFF. The results of these applications on some JSSPs from Lawrence's dataset [113] have been depicted in Figure 3.9. This figure shows the best and average result of 10 different runs using population size of 500 for 100 iterations. The bars show the best solution and the extensions illustrate the averages. As can be seen clearly in Figure 3.9, the PBFF offers better solutions in almost all the sample problems. Moreover, in the cases that both methods find the optimal solutions, the PBFF offers the lower average hence a better convergence rate.

This experiment is also done on the FJSS problems from BRData [22]. Since the results for these test problems are very close to the best solutions, the experiment includes 100 different runs using population size of 1000 in 200 iterations. Figure

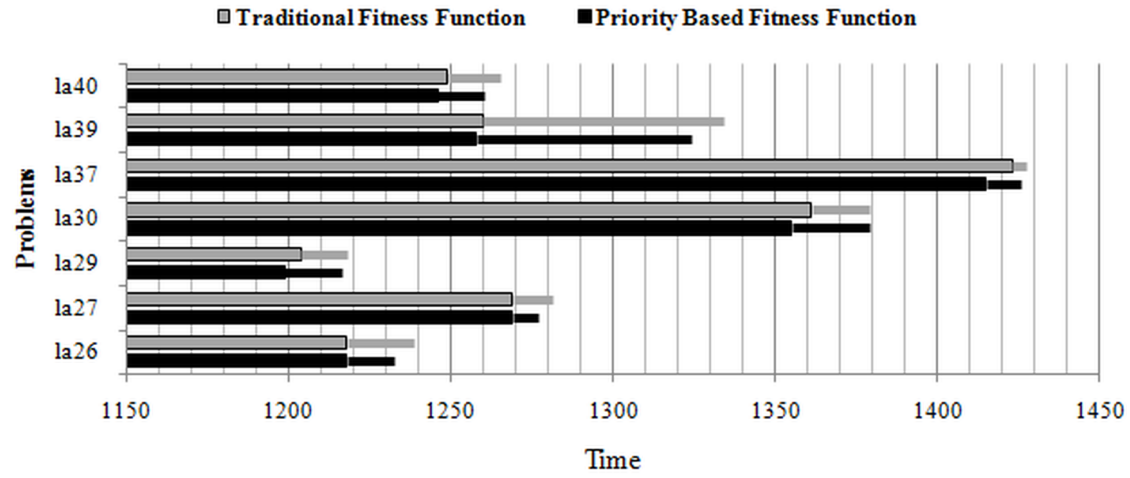


Figure 3.9: The best (main bar) and the average (extended portion) values of 10 runs using the traditional fitness function versus PBFF on Lawrence's dataset [113]

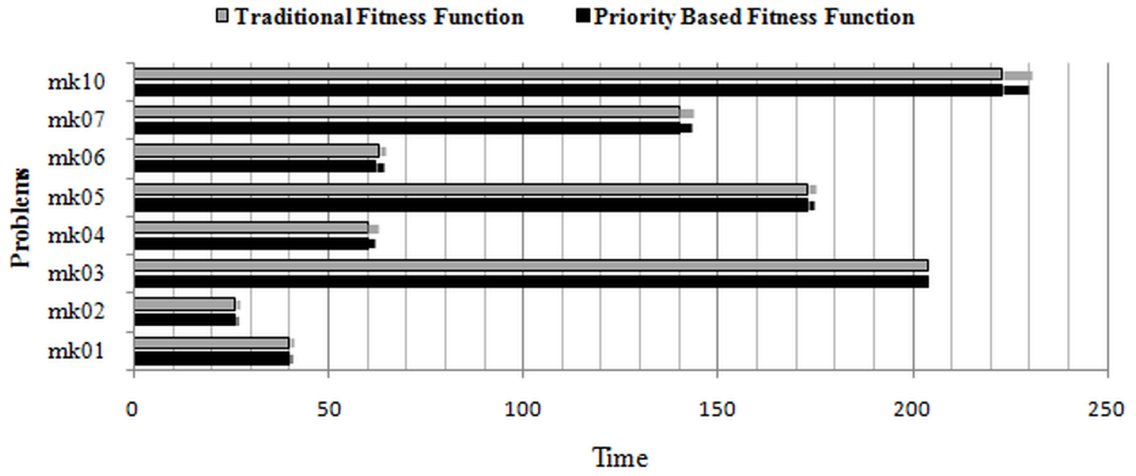


Figure 3.10: The best (main bar) and the average (extended portion) values of 100 runs using the traditional fitness function versus PBFF on BRData [22]

3.10 depicts the best and average results of the experiment. As the figure illustrates, incorporating PBFF offers better solution for problem mk06 and makes better average for others. For problem mk03, since both methods find the optimal solution in all runs, the average is the same as the best result. Considering all the results into account, we can say that the PBFF compared to the traditional fitness function offers better solutions as well as a better convergence rate.

After evaluating the effectiveness of the proposed fitness function, the performance of the proposed LS is examined. So the proposed MA as well as the proposed GA

itself without LS are applied to Lawrence's benchmark [113]. The results have been presented in Table 3.8 which show that LS heuristic improves the overall results. In all the 40 test problems, the proposed MA has the same or better results than the GA itself. So, it is possible to say that the proposed MA outperforms the proposed GA in all the test problems. The proposed GA can find the optimal solution for 27 problems out of 40, and the proposed MA found the best solution for two more problems. Moreover, the average, standard deviation, and median show that incorporating LS makes the proposed method more stable to find optimal or near-optimal solutions. In other words, it improves the convergence rate of the GA.

Table 3.8: Results on Lawrence's benchmark [113] (la01-la40)

Problem	Algorithm	Best	Average	SD	Median	Worst
la01	MA [87]	**666**	667.60	2.90	667	678
10×5	Proposed GA	**666**	666.00	0.00	666	666
666	Proposed MA	**666**	666.00	0.00	666	666
la02	MA [87]	**655**	656.27	3.39	655	666
10×5	Proposed GA	**655**	658.22	3.66	657	669
655	Proposed MA	**655**	655.00	0.00	655	655
la03	MA [87]	**597**	613.93	7.63	617	619
10×5	Proposed GA	**597**	606.40	9.34	603	617
597	Proposed MA	**597**	597.00	0.00	597	597
la04	MA [87]	**590**	593.33	2.44	595	595
10×5	Proposed GA	**590**	592.92	2.60	593	602
590	Proposed MA	**590**	590.60	1.26	590	593
la05	MA [87]	**593**	593.00	0.00	593	593
10×5	Proposed GA	**593**	593.00	0.00	593	593
593	Proposed MA	**593**	593.00	0.00	593	593

Table 3.8 – Continued on next page

Table 3.8: **(continued from previous page)**

Problem	Algorithm	Best	Average	SD	Median	Worst
la06	MA [87]	**926**	926.00	0.00	926	926
15×5	Proposed GA	**926**	926.00	0.00	926	926
926	Proposed MA	**926**	926.00	0.00	926	926
la07	MA [87]	**890**	890.00	0.00	890	890
15×5	Proposed GA	**890**	890.00	0.00	890	890
890	Proposed MA	**890**	890.00	0.00	890	890
la08	MA [87]	**863**	863.00	0.00	863	863
15×5	Proposed GA	**863**	863.00	0.00	863	863
863	Proposed MA	**863**	863.00	0.00	863	863
la09	MA [87]	**951**	951.00	0.00	951	951
15×5	Proposed GA	**951**	951.00	0.00	951	951
951	Proposed MA	**951**	951.00	0.00	951	951
la10	MA [87]	**958**	958.00	0.00	958	958
15×5	Proposed GA	**958**	958.00	0.00	958	958
958	Proposed MA	**958**	958.00	0.00	958	958
la11	MA [87]	**1222**	1222.00	0.00	1222	1222
20×5	Proposed GA	**1222**	1222.00	0.00	1222	1222
1222	Proposed MA	**1222**	1222.00	0.00	1222	1222
la12	MA [87]	**1039**	1039.00	0.00	1039	1039
20×5	Proposed GA	**1039**	1039.00	0.00	1039	1039
1039	Proposed MA	**1039**	1039.00	0.00	1039	1039
la13	MA [87]	**1150**	1150.00	0.00	1150	1150
20×5	Proposed GA	**1150**	1150.00	0.00	1150	1150
1150	Proposed MA	**1150**	1150.00	0.00	1150	1150

Table 3.8 – Continued on next page

Table 3.8: **(continued from previous page)**

Problem	Algorithm	Best	Average	SD	Median	Worst
la14	MA [87]	**1292**	1292.00	0.00	1292	1292
20×5	Proposed GA	**1292**	1292.00	0.00	1292	1292
1292	Proposed MA	**1292**	1292.00	0.00	1292	1292
la15	MA [87]	**1207**	1207.13	0.52	1207	1209
20×5	Proposed GA	**1207**	1207.00	0.00	1207	1207
1207	Proposed MA	**1207**	1207.00	0.00	1207	1207
la16	MA [87]	**945**	968.27	15.46	979	982
10×10	Proposed GA	**945**	964.85	15.33	966	982
945	Proposed MA	**945**	952.40	8.72	950	973
la17	MA [87]	**784**	788.93	4.18	792	793
10×10	Proposed GA	**784**	784.81	2.44	784	804
784	Proposed MA	**784**	784.00	0.00	784	784
la18	MA [87]	**848**	859.27	4.57	861	861
10×10	Proposed GA	**848**	859.37	4.55	861	876
848	Proposed MA	**848**	848.50	1.41	848	852
la19	MA [87]	**842**	855.47	7.76	855	869
10×10	Proposed GA	**842**	859.34	8.09	860	876
842	Proposed MA	**842**	849.00	3.13	849	854
la20	MA [87]	907	910.00	2.54	912	912
10×10	Proposed GA	907	910.76	3.46	911	932
902	Proposed MA	**902**	906.90	2.13	907	911
la21	MA [87]	1079	1097.60	12.48	1096	1124
15×10	Proposed GA	1055	1074.22	11.69	1075	1116
1046	Proposed MA	1053	1058.20	4.02	1059	1067

Table 3.8 – Continued on next page

Table 3.8: **(continued from previous page)**

Problem	Algorithm	Best	Average	SD	Median	Worst
la22	MA [87]	960	981.00	13.58	979	1000
15×10	Proposed GA	937	954.28	8.53	951	981
927	Proposed MA	**927**	934.20	4.54	933.5	941
la23	MA [87]	**1032**	1032.00	0.00	1032	1032
15×10	Proposed GA	**1032**	1032.20	0.99	1032	1038
1032	Proposed MA	**1032**	1032.00	0.00	1032	1032
la24	MA [87]	959	996.40	16.21	1003	1011
15×10	Proposed GA	950	970.41	10.28	967	998
935	Proposed MA	941	954.70	11.61	953	967
la25	MA [87]	991	1016.67	19.64	1013	1076
15×10	Proposed GA	990	1012.13	12.40	1009	1039
977	Proposed MA	984	986.90	2.60	986	992
la26	MA [87]	**1218**	1234.27	16.12	1228	1266
20×10	Proposed GA	**1218**	1246.57	13.98	1245	1285
1218	Proposed MA	**1218**	1218.00	0.00	1218	1218
la27	MA [87]	1286	1306.33	13.17	1301	1333
20×10	Proposed GA	1269	1295.13	10.76	1294	1326
1235	Proposed MA	1256	1267.10	6.44	1269	1275
la28	MA [87]	1286	1306.33	13.17	1301	1333
20×10	Proposed GA	1229	1262.12	11.75	1262	1296
1216	Proposed MA	1223	1231.80	6.01	1234	1241
la29	MA [87]	1221	1240.47	6.98	1240.5	1249
20×10	Proposed GA	1201	1229.14	10.61	1228	1250
1157	Proposed MA	1187	1196.40	6.10	1198	1206

Table 3.8 – Continued on next page

Table 3.8: **(continued from previous page)**

Problem	Algorithm	Best	Average	SD	Median	Worst
la30	MA [87]	**1355**	1362.33	8.23	1359.5	1380
20×10	Proposed GA	**1355**	1376.06	13.59	1374	1409
1355	Proposed MA	**1355**	1355.00	0.00	1355	1355
la31	MA [87]	1784	1784.00	0.00	1784	1784
30×10	Proposed GA	**1784**	1784.00	0.00	1784	1784
1784	Proposed MA	**1784**	1784.00	0.00	1784	1784
la32	MA [87]	1850	1850.00	0.00	1850	1850
30×10	Proposed GA	**1850**	1850.41	1.19	1850	1856
1850	Proposed MA	**1850**	1850.00	0.00	1850	1850
la33	MA [87]	1719	1719.00	0.00	1719	1719
30×10	Proposed GA	**1719**	1719.00	0.00	1719	1719
1719	Proposed MA	**1719**	1719.00	0.00	1719	1719
la34	MA [87]	1721	1721.00	0.00	1721	1721
30×10	Proposed GA	**1721**	1734.29	6.84	1734	1750
1721	Proposed MA	**1721**	1721.00	0.00	1721	1721
la35	MA [87]	1888	1888.00	0.00	1888	1888
30×10	Proposed GA	**1888**	1888.02	0.14	1888	1889
1888	Proposed MA	**1888**	1888.00	0.00	1888	1888
la36	MA [87]	1307	1328.67	11.59	1327	1346
15×15	Proposed GA	1291	1310.97	10.06	1308.5	1334
1268	Proposed MA	1276	1285.70	10.50	1278	1300
la37	MA [87]	1442	1473.60	11.51	1479	1487
15×15	Proposed GA	1425	1434.90	18.81	1425	1483
1397	Proposed MA	1401	1420.40	8.68	1425	1425

Table 3.8 – Continued on next page

Table 3.8: **(continued from previous page)**

Problem	Algorithm	Best	Average	SD	Median	Worst
la38	MA [87]	1266	1309.13	14.52	1314	1329
15×15	Proposed GA	1219	1252.20	14.36	1252	1304
1196	Proposed MA	1208	1218.30	8.07	1218	1238
la39	MA [87]	1252	1282.60	16.62	1277	1301
15×15	Proposed GA	1258	1276.76	9.79	1275	1309
1233	Proposed MA	1240	1251.30	4.81	1251	1258
la40	MA [87]	1252	1279.60	17.84	1289	1303
15×15	Proposed GA	1244	1266.91	7.66	1269	1287
1222	Proposed MA	1233	1246.70	7.18	1246.5	1259

To compare the results, the best-known solutions and the results of another MA recently proposed by Hasan *et al.* [87] have been presented in Table 3.8 as well. Clearly, our proposed GA without LS outperforms their MA in 39 test problems out of 40. Only on test problem $la39$ we observe their result to be better than the result of our proposed GA. However, our proposed MA outperforms their MA for all the 40 test problems.

Furthermore, regarding algorithm efficiency, our proposed method outperforms Hasan *et al.*'s method [87]. As mentioned before, the time complexity of our proposed local search on Lawrence's benchmark [113] can be calculated using Equation 3.1. Based on values 1000 and 200 for $PopSize$ and $IterationNo$, the $TimeComplexity$ would be less than $250,000$ fitness evaluations. Therefor, the overall time complexity of our proposed method would be less than $450,000$ fitness evaluations, while the time complexity of their method is $2,500,000$ fitness evaluations. In the other words, our

proposed method offers better solutions while it only needs less than $\frac{1}{5}$ of the fitness evaluations used by their method.

The proposed method is also compared with other algorithms for classical JSSP including our previous work [164], the heuristic method introduced by Adams *et al.* [2] and the hybrid GA proposed by Goncalves *et al.* [70]. This comparison has been represented in Table 3.9. The optimal solutions are illustrated in bold face, and asterisk (*) determines the best solutions found by compared algorithms. All the results of our proposed method received a bold face or asterisk except two problems: la21 and la25. This means that the proposed method outperforms all the mentioned algorithms by offering the best solution in almost all test problems. To be able to compare the algorithms over all 40 problems, an Error Rate (ER) is defined as follows.

$$ER = \frac{C - LB}{LB} \times 100\%$$

where LB is the best-known solution, and C is the best solution found by the algorithms. Table 3.9 represents the ER values in brackets. As illustrated in this table, our proposed method has the minimum average of ER values for all the 40 test problems. To show more comprehensive comparisons, the ER values corresponding to the most challenging test problems have been illustrated in Figure 3.11.

Table 3.9: Results on Lawrence's benchmark [113] (la01-la40)

Problem	Heuristic [2]	Hybrid GA [70]	Hassan et al. MA [87]	Our Previous MA [164]	Proposed MA
LA01	**666**(0.00%)	**666**(0.00%)	**666**(0.00%)	**666**(0.00%)	**666**(0.00%)
LA02	669 (2.14%)	**655**(0.00%)	**655**(0.00%)	**655**(0.00%)	**655**(0.00%)
LA03	605 (1.34%)	**597**(0.00%)	**597**(0.00%)	**597**(0.00%)	**597**(0.00%)
LA04	593 (0.51%)	**590**(0.00%)	**590**(0.00%)	**590**(0.00%)	**590**(0.00%)
LA05	**593**(0.00%)	**593**(0.00%)	**593**(0.00%)	**593**(0.00%)	**593**(0.00%)

 Table 3.9 – Continued on next page

Table 3.9: **(continued from previous page)**

Problem	Heuristic [2]	Hybrid GA [70]	Hassan et al. MA [87]	Our Previous MA [164]	Proposed MA
LA06	**926**(0.00%)	**926**(0.00%)	**926**(0.00%)	**926**(0.00%)	**926**(0.00%)
LA07	**890**(0.00%)	**890**(0.00%)	**890**(0.00%)	**890**(0.00%)	**890**(0.00%)
LA08	**863**(0.00%)	**863**(0.00%)	**863**(0.00%)	**863**(0.00%)	**863**(0.00%)
LA09	**951**(0.00%)	**951**(0.00%)	**951**(0.00%)	**951**(0.00%)	**951**(0.00%)
LA10	959 (0.10%)	**958**(0.00%)	**958**(0.00%)	**958**(0.00%)	**958**(0.00%)
LA11	**1222**(0.00%)	**1222**(0.00%)	**1222**(0.00%)	**1222**(0.00%)	**1222** (0.00%)
LA12	**1039**(0.00%)	**1039**(0.00%)	**1039**(0.00%)	**1039**(0.00%)	**1039** (0.00%)
LA13	**1150**(0.00%)	**1150**(0.00%)	**1150**(0.00%)	**1150**(0.00%)	**1150** (0.00%)
LA14	**1292**(0.00%)	**1292**(0.00%)	**1292**(0.00%)	**1292**(0.00%)	**1292** (0.00%)
LA15	**1207**(0.00%)	**1207**(0.00%)	**1207**(0.00%)	**1207**(0.00%)	**1207** (0.00%)
LA16	978 (3.49%)	**945**(0.00%)	**945**(0.00%)	**945**(0.00%)	**945**(0.00%)
LA17	787 (0.38%)	**784**(0.00%)	**784**(0.00%)	**784**(0.00%)	**784**(0.00%)
LA18	859 (1.30%)	**848**(0.00%)	**848**(0.00%)	**848**(0.00%)	**848**(0.00%)
LA19	860 (2.14%)	**842**(0.00%)	**842**(0.00%)	**842**(0.00%)	**842**(0.00%)
LA20	914 (1.33%)	907 (0.55%)	907 (0.55%)	907 (0.55%)	**902**(0.00%)
LA21	1084 (3.63%)	**1046**(0.00%)	1079 (3.15%)	1057 (1.05%)	1053 (0.67%)
LA22	944 (1.83%)	935 (0.86%)	960 (3.56%)	935 (0.86%)	**927**(0.00%)
LA23	**1032**(0.00%)	**1032**(0.00%)	**1032**(0.00%)	**1032**(0.00%)	**1032** (0.00%)
LA24	976 (4.39%)	953 (1.93%)	959 (2.57%)	944 (0.96%)	941 (0.64%)*
LA25	1017 (4.09%)	986 (0.92%)	991 (1.43%)	983 (0.61%)*	984 (0.72%)
LA26	1224 (0.49%)	**1218**(0.00%)	**1218**(0.00%)	**1218**(0.00%)	**1218**(0.00%)
LA27	1291 (4.53%)	1256 (1.70%)*	1286 (4.13%)	1269 (2.75%)	1256 (1.70%)*
LA28	1250 (2.80%)	1232 (1.32%)	1286 (5.76%)	1223 (0.58%)*	1223 (0.58%)*
LA29	1239 (7.09%)	1196 (3.37%)	1221 (5.53%)	1191 (2.94%)	1187 (2.59%)*
LA30	**1355**(0.00%)	**1355**(0.00%)	**1355**(0.00%)	**1355**(0.00%)	**1355** (0.00%)

Table 3.9 – Continued on next page

Table 3.9: **(continued from previous page)**

Problem	Heuristic [2]	Hybrid GA [70]	Hassan et al. MA [87]	Our Previous MA [164]	Proposed MA
LA31	**1784**(0.00%)	**1784**(0.00%)	**1784**(0.00%)	**1784**(0.00%)	**1784** (0.00%)
LA32	**1850**(0.00%)	**1850**(0.00%)	**1850**(0.00%)	**1850**(0.00%)	**1850** (0.00%)
LA33	**1719**(0.00%)	**1719**(0.00%)	**1719**(0.00%)	**1719**(0.00%)	**1719** (0.00%)
LA34	**1721**(0.00%)	**1721**(0.00%)	**1721**(0.00%)	**1721**(0.00%)	**1721** (0.00%)
LA35	**1888**(0.00%)	**1888**(0.00%)	**1888**(0.00%)	**1888**(0.00%)	**1888** (0.00%)
LA36	1305 (2.92%)	1279 (0.87%)	1307 (3.08%)	1281 (1.03%)	1276 (0.63%)*
LA37	1423 (1.86%)	1408 (0.79%)	1442 (3.22%)	1429 (2.29%)	1401 (0.29%)*
LA38	1255 (4.93%)	1219 (1.92%)	1266 (5.85%)	1208 (1.00%)*	1208 (1.00%)*
LA39	1273 (3.24%)	1246 (1.05%)	1252 (1.54%)	1248 (1.22%)	1240 (0.57%)*
LA40	1269 (3.85%)	1241 (1.55%)	1252 (2.45%)	1234 (0.98%)	1233 (0.90%)*
Avg. ER	1.46%	0.42%	1.07%	0.42%	0.26%

Furthermore, our proposed MA is also evaluated for Flexible Job Shop Scheduling. The results have been presented in Table 3.10. The results are compared to the state-of-the-art approaches including Hybrid GA proposed by Gao *et al.* [63], Artificial Immune Algorithm proposed by Bagheri *et al.* [10], three Genetic Algorithms proposed by Chen *et al.* [30], Pezzella *et al.* [158] and Zhang *et al.* [221], and Clonal Selection approach suggested by Ong *et al.* [150]. The results show that our proposed method offers competitive solutions compared to the existing state-of-the-art methods in FJSS. The proposed method finds the best solution for 8 problems out of the 10. Though Ong *et al.* [150] claimed that they found 39 as the makespan for test problem $mk01$, the makespan of 39 for this test problem is not feasible and the best solution is 40. This issue has been proved in Appendix A.

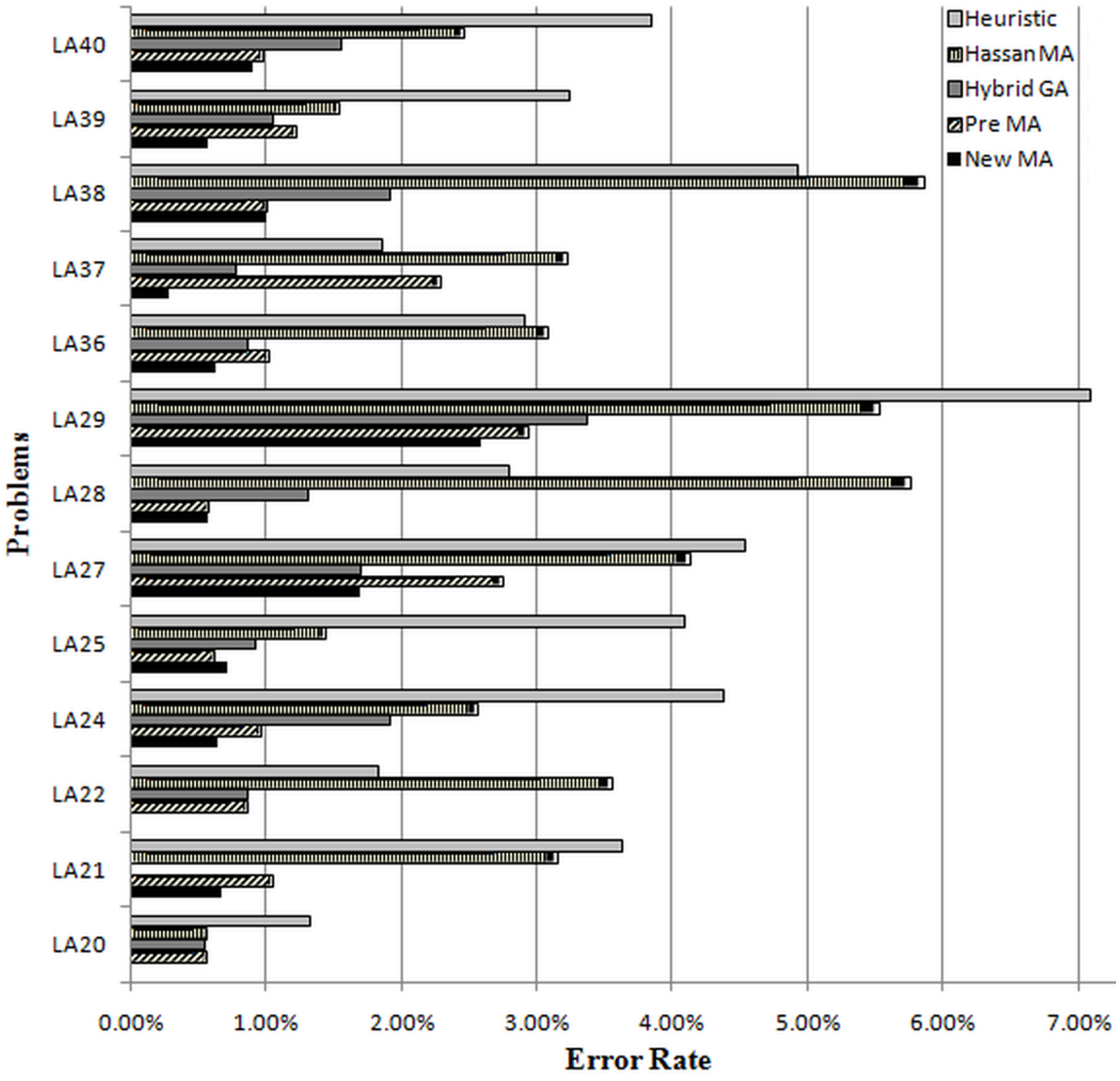


Figure 3.11: Comparison of different types of algorithms on Lawrence's benchmark [113]

Table 3.10: Comparison of different types of algorithms on BRData [22]

Problem	Size	hGA [63]	AIA [10]	GA Chen[30]	GA Pezzella[158]	GA Zhang[221]	Clonal [150]	Proposed MA
mk01	10×6	**40**	**40**	**40**	**40**	**40**	39	**40**
mk02	10×6	**26**	**26**	29	**26**	**26**	27	**26**
mk03	15×8	**204**	**204**	**204**	**204**	**204**	-	**204**
mk04	15×8	**60**	**60**	63	**60**	**60**	65	**60**
mk05	15×4	**172**	173	181	173	173	173	**172**
mk06	10×15	**58**	63	60	63	**58**	70	59
mk07	20×5	**139**	140	148	**139**	144	145	**139**
mk08	20×10	**523**	**523**	**523**	**523**	**523**	**523**	**523**
mk09	20×10	**307**	312	308	311	**307**	311	**307**
mk10	20×15	**197**	214	212	212	198	-	216

The proposed MA was also applied in MO-FJSS problems. The benchmarks presented in Xing *et al.* [208] are considered as test problems. The scheduling goal is to minimize a multi-objective function with different weight represented in Equation 3.2. The results of both our proposed MA and their algorithm have been presented in Table 3.11. Our proposed MA found the best solutions for all the problems. Moreover, it improves the solution for the 4^{th} benchmark. The solution found for this benchmark, whose problem specification is presented in Table 3.5, has been depicted in Figure 3.12.

Table 3.11: Results on FJSS benchmarks used by Xing et al. [208]

Benchmarks	Xing et al. [208]				Proposed MA			
	F_1	F_2	F_3	F	F_1	F_2	F_3	F
Instance 1	12	32	8	14.8	12	32	8	14.8
Instance 2	14	77	12	26	14	77	12	26
Instance 3	11	62	10	20.9	11	62	10	20.9
Instance 4	7	42	6	13.7	7	43	5	**13.6**
Instance 5	11	91	11	27	11	91	11	27

3.3.6 Conclusions

In this article, a Memetic Algorithm is proposed to deal with different types of job shop scheduling problems. The main contributions of our proposed methods are the

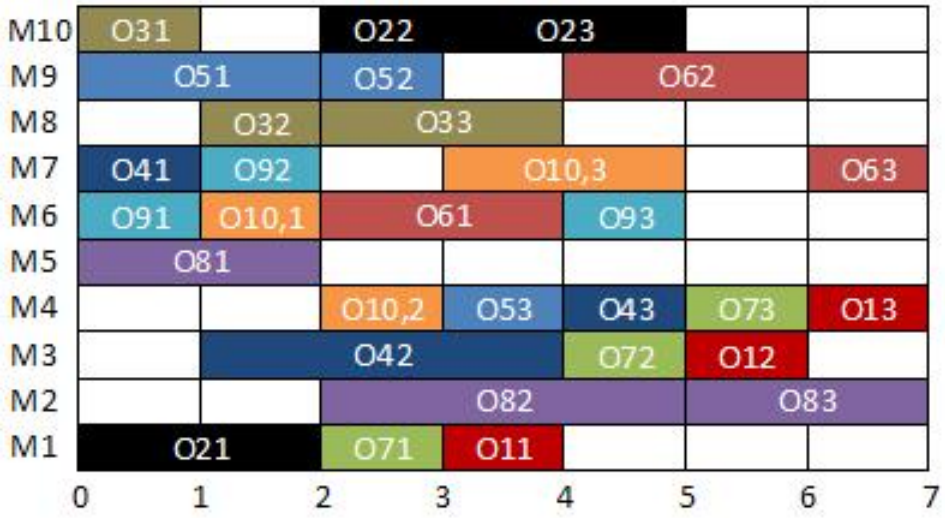


Figure 3.12: Results on 4^{th} FJSS benchmark used by Xing et al. [208]

introduction of the Priority-Based Fitness Function and a new heuristic for local search approaches.

The experiment results show that incorporating our proposed fitness function improves the final solutions by saving the individuals which have better characteristics. The proposed GA method itself without the LS outperforms the MA recently proposed by Hasan *et al.* [87] for 39 test problems out of 40 in JSSP. Moreover, the proposed GA is improved by incorporating the LS, such that it finds the optimal solutions for two more test problems. The better statistical information for MA such as standard deviation determines that the LS helps the GA to offers more stable results, which means it helps the GA to converge faster.

The proposed method is also effective in other types of job shop scheduling problem. Compared to the state-of-the-art methods in FJSS, our proposed method offers competitive results by obtaining the optimal solutions for 8 test problems and finding the near-optimal solutions for the rest two problems. In addition, the application of the proposed method in MO-FJSS shows that it is efficient for these problems as well. Our proposed method finds the best solutions for all the test problems and more importantly it improves the final solution for one test problem.

The application of the proposed algorithm on Stochastic Flexible Job Shop Scheduling will be considered in future work. Moreover, the parameters of the algorithm have not been investigated to find the best set of parameters.

3.3.7 Appendix A

Test problem $mk01$ is one of the *BRData* problems published by Brandimarte [22]. The best makespan found for this test problem by the proposed methods and almost all the existing methods is 40, while Ong *et al.* [150] claimed that they found 39. This Appendix is to prove that the makespan of 39 for this test problem is not feasible.

Claim 3.3.1. *The makespan of 39 or less for test problem* $mk01$ *of BRData is not feasible.*

Proof. The specification of problem $mk01$ which is a sample of P-FJSS problems has been presented in Table 3.12. It consists of 10 different jobs including either 5 or 6 operations which should be processed on 6 different machines. For each operations, the machines which are not applicable are denoted by X.

As can be seen in Table 3.12, there is no flexibility for operations O_{21}, O_{31}, O_{42}, O_{53}, O_{64} and O_{84}. They are all supposed to be processed on machine m_2. The processing times for all of them are the same as 6 time units. So, machine m_2 is supposed to work on these operations from time 0 to 36. It means that one of these operations has to be processed in time interval $(30-36)$. As all of these operations are not the last operations of their job, they have one or more job-proceeding operations. The minimum processing time needed for the job-proceeding operations of operations O_{21}, O_{31}, O_{42}, O_{53}, O_{64} and O_{84} are 10, 8, 4, 14, 3 and 6, respectively. So, if there is any schedule with makespan 39, it should assign operation O_{64} in the time interval $(30-36)$, and assign O_{65} to machine m_1 from time 36 to 37, and O_{66} to machine m_4 from time 37 to 39. Therefore, if it can be proved that machine m_4 is not idle in

Table 3.12: Test problem $mk01$ of $BRData$ [22]

Operations		Machines						Operations		Machines					
		m_1	m_2	m_3	m_4	m_5	m_6			m_1	m_2	m_3	m_4	m_5	m_6
J_1	O_{11}	5	X	4	X	X	X	J_6	O_{61}	X	X	4	X	X	2
	O_{12}	X	1	5	X	3	X		O_{62}	2	X	X	X	X	X
	O_{13}	X	X	4	X	X	2		O_{63}	X	6	4	X	X	6
	O_{14}	1	6	X	X	X	5		O_{64}	X	6	X	X	X	X
	O_{15}	X	X	1	X	X	X		O_{65}	1	6	X	X	X	5
	O_{16}	X	X	6	3	X	6		O_{66}	3	X	X	2	X	X
J_2	O_{21}	X	6	X	X	X	X	J_7	O_{71}	X	X	X	X	X	1
	O_{22}	X	X	1	X	X	X		O_{72}	3	X	X	2	X	X
	O_{23}	2	X	X	X	X	X		O_{73}	X	6	4	X	X	6
	O_{24}	X	6	X	6	X	X		O_{74}	6	6	X	X	1	X
	O_{25}	1	6	X	X	X	5		O_{75}	X	X	1	X	X	X
J_3	O_{31}	X	6	X	X	X	X	J_8	O_{81}	X	X	4	X	X	2
	O_{32}	X	X	4	X	X	2		O_{82}	X	6	4	X	X	6
	O_{33}	1	6	X	X	X	5		O_{83}	1	6	X	X	X	5
	O_{34}	X	6	4	X	X	6		O_{84}	X	6	X	X	X	X
	O_{35}	1	X	X	X	5	X		O_{85}	X	6	X	6	X	X
J_4	O_{41}	1	6	X	X	X	5	J_9	O_{91}	X	X	X	X	X	1
	O_{42}	X	6	X	X	X	X		O_{92}	1	X	X	X	5	X
	O_{43}	X	X	1	X	X	X		O_{93}	X	X	6	3	X	6
	O_{44}	X	1	5	X	3	X		O_{94}	2	X	X	X	X	X
	O_{45}	X	X	4	X	X	2		O_{95}	X	6	4	X	X	6
									O_{96}	X	6	X	6	X	X
J_5	O_{51}	X	1	5	X	3	X	J_{10}	O_{101}	X	X	4	X	X	2
	O_{52}	1	6	X	X	X	5		O_{102}	X	6	4	X	X	6
	O_{53}	X	6	X	X	X	X		O_{103}	X	1	5	X	3	X
	O_{54}	5	X	4	X	X	X		O_{104}	X	X	X	X	X	1
	O_{55}	X	6	X	6	X	X		O_{105}	X	6	X	6	X	X
	O_{56}	X	6	4	X	X	6		O_{106}	3	X	X	2	X	X

time interval (37-39), it has been proved that there is no feasible schedule for problem $mk01$ with makespan 39 or less.

Operations O_{24}, O_{55}, O_{85}, O_{96} and O_{105} have the flexibility to be processed on either machine m_2 or m_4. Since their processing time on both machines are 6, and machine m_2 is already assigned from time 0 to 36, all the 5 operations should be processed on machine m_4. Since all the 5 operations are not the first operations of their corresponding jobs, they have a number of job-preceding operations. So, the minimum time needed for the job-preceding operations to be processed for operations O_{24}, O_{55}, O_{85}, O_{96} and O_{105} are 9, 12, 13, 11 and 8, respectively. So, in the best case, machine m_4 should process these operations from time 8 to 38. It means that machine m_4 is not idle from time 37 to 38. Therefore, it is proved that machine m_4 is not idle in the time interval (37-39) and the best starting time for operation O_{66} on machine m_4 is time 38, for which the processing lasts to time 40. Also, O_{66} can be processed on machine m_2 from time 37 to 40. For both possible cases, the makespan is 40, and not less. So, the claim has been proved. □

Acknowledgment

This work was supported in part by a grant from the NSERC Discovery No. 327482.

3.4 Incorporating a Genetic Algorithm to improve the performance of Variable Neighborhood Search

Abstract. Variable Neighborhood Search (VNS) is an efficient metaheuristics in solving optimization problems. Although VNS has been successfully applied on various problem domains, it suffers from its inefficient search exploration. To improve this limitation, VNS can be joined with a population-based search to benefit from its search exploration. In this article, a Memetic Algorithm (MA) is proposed which is based on a Genetic Algorithm (GA) incorporating VNS as a local search method. To evaluate the proposed method, it has been applied on the classical Job Shop Scheduling Problem (JSSP) as a well-known optimization problem. The experimental results show that the proposed MA outperforms the VNS method. Furthermore, compared to the state-of-the-art Evolutionary Algorithms (EAs) proposed to solve JSSP, the proposed method offers competitive solutions.

3.4.1 Introduction

Metaheuristics are general procedures to design heuristics for optimization problems. Heuristics try to find an optimal solution by working on the solution space, while metaheuristics incorporate different strategies to build a heuristic with a good performance on one problem domain. Variable neighborhood search (VNS) is a metaheuristic proposed to solve combinatorial and global optimization problems.

The main idea of VNS is to change the neighborhood within a local search to avoid trapping into local optima. A VNS incorporates a number of neighborhood structures

to switch among them at the time of local search execution. When local search finds an optimal solution with respect to a neighborhood structure, VNS switch to another neighborhood structure to scape from the local optima.

Although VNS is recently introduced by Mladenovic and Hansen [136], it has been successfully applied in various combinatorial optimization problems such as Traveling Salesman Problem (TSP) [55], Open Vehicle Routing problem [57], and Graph problems [24].

In spite of successful application of VNS in different problems due to its powerful exploitation, it suffers from its inefficient search space exploration. Based on the definition of neighborhood structures, it is possible for VNS to spins in some previously investigated regions. Therefore, it cannot jump to other regions to look for better solutions. There are a number of methods proposed to overcome this limitation such as Parallel VNS [42], Multi-Start VNS [127], and Population-based VNS [205].

In this article, a Genetic Algorithm (GA) is incorporated to overcome the limitation of VNS. In other words, in this article, a Memetic Algorithm (MA) is proposed which uses a GA as a population-based search and incorporates VNS as a local search method. The main contribution of this article is to show that the joint GA and VNS works better than the single VNS. In other words, VNS as a local search has better performance than as a global search method. To evaluate our proposed method, it has been applied on Job Shop Scheduling Problem (JSSP) which is a well-known combinatorial optimization problem, specially in manufacturing systems.

The structure of this article is as follows. VNS is briefly introduced in Subsection 3.4.2, followed by a concise description of MA in Subsection 3.4.3. The definition of JSSP problem domain is presented in Subsection 3.4.4. Subsection 3.4.5 describes our proposed method in details. Finally, the experimental results are represented in Subsection 3.4.6, followed by conclusion remarks in Subsection 3.4.7.

3.4.2 Variable neighborhood search (VNS)

Some heuristics start with a feasible solution and look for a better solution by applying some moves in the neighborhood of the current solution. The main drawback of these heuristics is their immature convergence. If they trap in a local optimum, they spins around the neighborhood of that local optimum and never scape from that region. So they converge into a local optimum instead of the global one. This issue is more challenging in large size problems where the chance of trapping into local optima is higher. The larger the search space, the more likely to trap into local optima.

To defeat this issue, metaheuristics are appeared where they try to design a heuristic, more powerful for a problem domain. VNS is one of the most recent metaheuristics in the literature proposed to deal with optimization problems. The literature shows that it has a remarkable performance on various optimization problems.

VNS defines a multiple neighborhood structures and switches among them systematically within a local search. VNS change the neighborhood structure when local search finds a local optimum with respect to one neighborhood structure. So the neighborhood structures should be complementary to each other to bring this capability for the VNS to scape from local optima. Therefore, designing a multiple neighborhood structures is very crucial to build an efficient VNS method.

The main idea of VNS comes from the following facts:

1. A local optimum in one neighborhood is not necessarily a local optimum in another neighborhood.

2. A global optimum is a local optimum for all possible neighborhood structures.

3. For many problems, local optima with respect to one or more neighborhood structures are relatively close to each other.

The last one is an empirical observation. It implies that a local optimum often provides some information about the global optimum.

3.4.3 Memetic Algorithm (MA)

Evolutionary Algorithm (EA) is another category of metaheuristics proposed to solve combinatorial optimization problems. Although EAs show a very good performance on optimization problems, most of the recent work in this area hybridized an EA with a local search method to improve its performance. This hybridization not only improves the quality of the final results, but also helps the EA to convergence faster. MA defined by Moscato [139] is a combination of a population-based global search and a local search heuristic to deal with an optimization problem. The population-based global search works as search space exploration, while the local search works as the exploitation of the regions where the best found solutions belongs to [149]. In other words, the global search finds the promising regions and the local search finds the local optimal solutions in those regions.

So based on the definition of Moscato [139], all the combinations of an EA with a local search heuristic can be considered as MAs. There are a large number of successful combinations of an EA and a local search in the literature, but as we applied our proposed method on the scheduling problems, we refer to the MAs published by Gao *et al.* [63], Caumond *et al.* [28] and Chiang *et al.* [33] as the efficient MAs proposed to deal with different scheduling problems.

Three different priority rules are incorporated by Hassan *et al.* [87] to proposed different MAs to solve JSSP. The authors showed that incorporating joint of Gap Reduction and Restricted Swapping offers the most efficient MA. Moving operations to the first available gap in their corresponding machine is called Gap Reduction rule, while Restricted Swapping rule is to swap the adjacent operations on a machine.

3.4.4 Job Shop Scheduling Problem (JSSP)

Job Shop Scheduling Problem (JSSP) is an optimization problem which is very well-known in different areas, as it is applicable in various fields of study. In JSSP, there

are a number of jobs to be processed on a number of machines. Each job consists of a number of operations which have to be processed in a predefined sequence. The task of assigning the operations to the machines is called job scheduling. There are different types of scheduling problems in the literature, each of which has its own definition and constraints. One of the main optimization function in scheduling problems is to minimize the *makespan* which is the maximum completion time of all the jobs. Since JSSP is still an open problem, it is a good case study to evaluate the new methods. Moreover, it has been proved by Garey *et al.* [65] that JSSP systems with more than two machines are NP-complete problems. It means that there is no exact algorithm capable to find the optimal solution for all the scheduling problems in acceptable time.

Classical JSSP is one class of scheduling problems which is defined by Baker [11] as a process of assigning N jobs denoted by J_i to M machines denoted by m_k, where i is the job index ranging from 1 to N, and k is the machine index ranging from 1 to M. Each job consists of a fixed sequence of operations which are denoted by O_{ij} where i is the job index and j is the operation index in that job. In classical JSSP, each operation can be processed on only one machine in a known processing time.

Moreover, there are a number of rules considered in classical JSSP which are as follows:

1. Jobs are independent to each other.
2. All jobs are available at the beginning.
3. There is no due date for the jobs.
4. Each machine processes each job only one time which cannot be interrupted.
5. The machine set up time and part movement time between machines are considered negligible.

Table 3.13: A sample classical JSSP

Operation Index	1	2	3
J_1	m_2,1	m_1,2	m_3,3
J_2	m_1,2	m_2,1	m_3,2
J_3	m_1,2	m_3,4	m_2,1

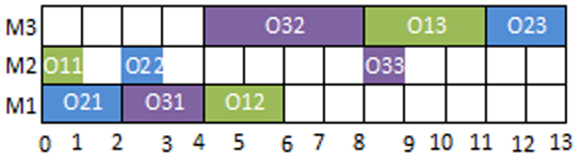


Figure 3.13: Sample schedule for the sample problem.

A sample classical JSSP is represented in Table 3.13. The sample problem consists of 3 jobs and 3 machines. The applicable machine and its corresponding processing time for each operation is presented in the table. For an instance, the second operation of the third job O_{32} has to be processed on machine m_3 for 4 time units. Fig. 3.13 illustrates a sample schedule for this sample problem with makespan of 13.

The concept of critical operations is described here which will be used later in the definition of neighborhood structures. The longest path of consecutive operations in a schedule which starts from time zero and ends at the makespan is called a critical path. A schedule may have one or more critical paths. The operations included in critical paths are called critical operations such that any delay in their processing time increases the makespan of the schedule. The adjacent critical operations on the same machine is called critical block.

Based on this definition, there is only one critical path in the sample schedule illustrated in Figure 3.13 which is as follows.

$$CriticalPath : O_{21} \prec O_{31} \prec O_{32} \prec O_{13} \prec O_{23}$$

So, there are the following two critical blocks.

$$CriticalBlock1 : O_{21} \prec O_{31}$$

$$CriticalBlock2 : O_{32} \prec O_{13} \prec O_{23}$$

The first operation in a critical block is called block head, the last one is called block rear, and the others are called internal operations.

3.4.5 Proposed Memetic Algorithm

As mentioned before, VNS suffers from its inefficient search space exploration. To improve its exploration, a population-based metaheuristic is incorporated such that VNS works as a local search method. The main contribution of this article is to compare the effectiveness of VNS, both as a single method and as a local search method for an EA. In this article, a GA is considered as the population-based search to improve the search exploration of VNS. The details of the proposed method are described in the following Sub-subsections.

3.4.5.1 Chromosome Representation

There are different chromosome representations for JSSP in the literature. In the proposed algorithm, an operation based representation is used. This representation which is introduced by Gen *et al.* [66] is also known as modified operation based representation. In this representation, a schedule is encoded into one string of numbers with the length of the total number of operations. Each number denotes one operation and the order of these numbers defines the operations processing sequence. In this representation, the operations are denoted by their job index in the string. Since the operations of the same job have to be processed in their fixed sequence, the operations with the same index are determined by the occurrence of that index. For

an instance, the second occurrence of the first job index in the string is decoded to the second operation of the first job.

This representation always generates feasible solutions, so there is no need for any repair mechanisms. One drawback of this representation is its n to 1 mapping. It means that different chromosomes in this representation could be decoded to the same schedule. Therefore, the number of possible chromosomes in this representation is much more than the number of different possible schedules.

The following string, for an instance, is a sample chromosome for the sample problem.

$$\{1, 2, 2, 3, 1, 3, 1, 2, 3\}$$

This chromosome is decoded into the following operation sequence.

$$O_{11} \prec O_{21} \prec O_{22} \prec O_{31} \prec O_{12}$$

$$\prec O_{32} \prec O_{13} \prec O_{23} \prec O_{33}$$

This sequence produces the following schedule.

$$m_1 : O_{21} \prec O_{31} \prec O_{12}$$

$$m_2 : O_{11} \prec O_{22} \prec O_{33}$$

$$m_3 : O_{32} \prec O_{13} \prec O_{23}$$

Figure 3.13 depicts the Gantt chart corresponding to this schedule.

3.4.5.2 Genetic Operators

Genetic operators for a GA include initialization, parent selection mechanisms, crossover and mutation operators, and fitness evaluation. To produce an initial

population, a number of individuals are generated randomly. This number is the population size which is a fixed parameter denoted by $PopSize$.

To generate offspring for the next generation, two parent individuals are selected. The first one is selected from a top best individuals of the whole population. This top proportion of the whole population is denoted by $TopBest$ parameter. The second parent is selected randomly from the whole population. After selecting two parents, the crossover operator is applied to generate two new individuals. In the proposed method, precedence preserving Order-based Crossover (POX) introduced by Lee *et al.* [114] is used as the crossover operator. This crossover selects one or more jobs randomly and copies all the operations of those jobs from the first parent to the first child. The remaining operations are inserted into the first child's chromosome by the same order as they have in the chromosome of second parent. The same procedure is applied to generate the second child from the selected parents.

A sample example of this crossover operation is presented as follows. Consider the following parents.

$$Parent1 : \{3, 4, 1, 4, 2, 1, 2, 3, 1, 3, 4, 4, 1, 2, 2, 3\}$$

$$Parent2 : \{1, 2, 4, 3, 3, 2, 1, 2, 4, 3, 1, 2, 3, 4, 4, 1\}$$

If both jobs 3 and 4 are selected by the crossover operator, after copying their corresponding operations, the children would be as follows.

$$Child1 : \{3, 4, -, 4, -, -, -, 3, -, 3, 4, 4, -, -, -, 3\}$$

$$Child2 : \{-, -, 4, 3, 3, -, -, -, 4, 3, -, -, 3, 4, 4, -\}$$

Finally, the gaps are filled by the remanning operations based on their order in the chromosome of another parent.

$$Child1 : \{3, 4, 1, 4, 2, 2, 1, 3, 2, 3, 4, 4, 1, 2, 1, 3\}$$

$$Child2 : \{1, 2, 4, 3, 3, 1, 2, 1, 4, 3, 1, 2, 3, 4, 4, 2\}$$

As a mutation operator, the simple swap method is incorporated which swaps two randomly selected operations. To evaluate each individual, the makespan of its schedule is considered as a fitness value. The lower the makespan, the better the individual. Since the goal is to minimize the makespan, it is better to call the evaluation function as a cost function not a fitness function. But like other published papers in this area, the well-known $fitnessfunction$ term has been used, while the goal is to minimize the fitness value.

3.4.5.3 Neighborhood Structures

There are various neighborhood structures proposed for JSSP in the literature. A list of six popular neighborhood structures for JSSP is provided by Blazewicz *et al.* [20]. The authors called these six structures as $N1$ to $N6$. It has to be mentioned that the successful neighborhood structures are based on critical operations. The first structure, so-called $N1$, is introduced by Van Laarhoven *et al.* [200]. $N1$ neighborhood structure is the simplest structure which generates neighbors by swapping any two adjacent critical operations.

Neighborhood structure $N4$ is introduced by Dell'Amico and Trubian [50] which includes moving an internal operation to the very beginning or the very end within a critical block. Nowicki and Smutnicki [146] introduced neighborhood structure $N5$ including swapping the first two operations or the last two operations of a critical

block. Among the six neighborhood structures, $N1$ and $N5$ structures generate the largest and the smallest neighborhood, respectively.

In our proposed method, we incorporate only $N4$ and $N5$ neighborhood structures. It should be mentioned here that in our implementation we did not apply any pre-processing or post-processing procedures for our moves. We just move one element in the chromosome exactly like mutation operator.

3.4.5.4 Proposed VNS

Our proposed VNS consists of one *Shake* method and one *LocalSearch* method. Applying *Shake* method followed by *LocalSearch* on an individual is called a *run*. Using the *Shake* method, a neighbor individual of the current individual is generated. Then the *LocalSearch* method finds the local optimum solution in the neighborhood of that neighbor individual. In the *Shake* method, we applied four random consecutive moves as follows.

$$Shake : N5 \rightarrow N4 \rightarrow N5 \rightarrow N4$$

LocalSearch method incorporates both $N4$ and $N5$ as the regular neighborhood searches and moreover it uses $N5$ as a nested neighborhood search. It should be mentioned that to save the computational time, we used the first improvement strategy instead of the best improvement one in our *LocalSearch* method.

3.4.5.5 MA Framework

The overall framework of the proposed method is as follows. First, the GA generates the initial population with *PopSize* random individuals. Then in each generation, after applying genetic operators and generating offspring, the proposed VNS is applied on the top best individuals of the current population and the offspring one. As mentioned before, the top best individuals is a proportion of the whole population

denoted by $TopBest$ parameter. This routine continues for a predefined number of iterations denoted by $IterationNo$ parameter.

3.4.6 Results

The proposed algorithm is implemented and evaluated using the *java* programming language version 1.6.0.18 on a system with $Intel(R)\ Core(TM)2Quad\ 2.50GHz\ CPU$ and $8.00GB\ RAM$. The algorithm parameters are adjusted using extensive experiments. We used a population of 100 individuals with the top 10 best individuals to be evolved for 400 iterations.

To evaluate the performance of the proposed method it has been applied on classical JSSP. One of the well-known benchmark in this area is introduced by Lawrence [113] including 40 different problems with different size. All the evaluations are experimented by 10 independent runs for each problem.

In order to compare the proposed method with a single VNS, the proposed VNS itself is also applied on this benchmark. Regarding the time complexity of proposed VNS and GA, it should be mentioned that the most time consuming part of these algorithms is fitness evaluation, and the time required for the rest is negligible compared to fitness evaluation time complexity. Therefore, the single VNS is applied for 200,000 runs (*Shake* followed by *LocalSearch*) in order to have the same number of fitness evaluations for both proposed MA and VNS. So the comparison would be fair. Table 3.14 presents some sample results [1] of applying both proposed MA and proposed VNS on LA benchmark.

The experimental results show that the proposed method finds the optimal solutions for 34 LA test problems out of 40, and the proposed VNS itself finds 33 optimal solutions. In other words, the proposed MA finds the optimal solution for test problem $la25$, while the proposed VNS cannot reach to that solution. Comparing all the

[1] To see all the results please refer to Subsection 3.4.8. These results are also accessible online at http://cs.uwindsor.ca/∼raeesim/NaBIC/Statictical-Analysis.pdf

Table 3.14: Sample Results on LA Benchmark

Method	Problem (BK)	Best	Average	SD	Median	Worst
VNS	la25 (**977**)	979	985.7	5.40	984.0	996
MA	15×10	**977**	981.1	3.11	981.5	984
VNS	la29 (**1152**)	1169	1187.0	15.19	1185.0	1216
MA	20×10	1163	1175.4	9.65	1178.0	1188
VNS	la36 (**1268**)	1291	1294.6	6.26	1291.5	1311
MA	15×15	1281	1285.2	5.43	1281.0	1292
VNS	la37 (**1397**)	**1397**	1420.5	12.55	1422.5	1442
MA	15×15	**1397**	1398.9	2.13	1398.0	1402

results show that the proposed MA always offers a good solution, while the range of obtained makespans by VNS are much wider. Consider test problem $la37$ for an instance. Our proposed MA finds the optimal solution for this test problem 5 times in 10 independent runs, while the single VNS finds the optimal solution only 1 time in 10 runs. Moreover, comparing the average, median, standard deviation and worst found solutions shows that the proposed MA is more stable to find good quality solutions than the single VNS. Comparing the results over all 40 test problems shows that the proposed MA outperforms the proposed VNS itself by offering better solutions as well as better convergence rate.

The results obtained by the VNS itself shows that it suffers from its inefficient search space exploration since it offers the results with a wide range of quality. This limitation is defeated by the efficient exploration of a GA. Therefore, this combination always offers good quality solutions. In order to show the performance of the proposed method, the state-of-the-art methods in this area are considered for comparison, including a hybrid EA and our published MA, and a hybrid GA. It should be mentioned that all these three algorithm are recently published. The hybrid EA proposed by Zobolas *et al.* [225] incorporates VNS as its local search method. Moreover, they used Differential Evolution (DE) to generate a valuable initial population. Our recently published MA [169] incorporate a simple local search method to find the optimal solutions. The hybrid GA proposed by Qing-dao-er-ji and Wang [45]

Table 3.15: Comparison among Different EAs proposed recently to solve JSSP

Problem	BK	hEA (2009) [225]	MA (2012) [169]	hGA (2012) [45]	Proposed VNS	Proposed MA
LA20	**902**	-	907 (0.55%)	907 (0.55%)	**902** (0.00%)	**902** (0.00%)
LA21	**1046**	**1046** (0.00%)	1053 (0.67%)	**1046** (0.00%)	**1046** (0.00%)	**1046** (0.00%)
LA22	**927**	**927** (0.00%)	**927** (0.00%)	935 (0.86%)	**927** (0.00%)	**927** (0.00%)
LA24	**935**	**935** (0.00%)	941 (0.64%)	953 (1.93%)	**935** (0.00%)	**935** (0.00%)
LA25	**977**	**977** (0.00%)	984 (0.72%)	981 (0.41%)	979 (0.20%)	**977** (0.00%)
LA27	**1235**	1236* (0.08%)	1256 (1.70%)	1236* (0.08%)	1244 (0.73%)	1238 (0.24%)
LA28	**1216**	1224 (0.66%)	1223 (0.58%)	**1216** (0.00%)	**1216** (0.00%)	**1216** (0.00%)
LA29	**1152**	1160* (0.69%)	1187 (3.04%)	1160* (0.69%)	1169 (1.48%)	1163 (0.95%)
LA36	**1268**	**1268** (0.00%)	1276 (0.63%)	1287 (1.50%)	1291 (1.81%)	1281 (1.03%)
LA37	**1397**	1408 (0.79%)	1401 (0.29%)	1407 (0.72%)	**1397** (0.00%)	**1397** (0.00%)
LA38	**1196**	1202* (0.50%)	1208 (1.00%)	**1196** (0.00%)	1208 (1.00%)	1208 (1.00%)
LA39	**1233**	**1233** (0.00%)	1240 (0.57%)	**1233** (0.00%)	1241 (0.65%)	1241 (0.65%)
LA40	**1222**	1229* (0.57%)	1233 (0.90%)	1229* (0.57%)	1233 (0.90%)	1233 (0.90%)
Avg. ER		0.27%	0.87%	0.56	0.52%	0.37%

introduced new genetic operators as well as a new local search method specified for JSSP.

To save the space in this article, the results on the most challenging test problems are represented in Table 3.15. To have a fair comparison over these 13 test problems, the Error Rate (ER) parameter is incorporated which is defined as follows.

$$ER = \frac{C - LB}{LB} \times 100\%$$

where C is the best solution found by an algorithm and LB is the best-known solution. The ER values for each test problem are represented in brackets in Table 3.15, and the last row shows the average ER over 13 test problems. This comparison shows that our proposed MA offers competitive solutions compared to state-of-the-art methods.

3.4.7 Conclusions

In this article, a VNS is combined with a GA to improve its performance. VNS which is one of the most recent metaheuristics in combinatorial optimization problem, has

been successfully applied in various research areas. In spite of its successfulness, it suffers from its inefficient search space exploration. The main contribution of this paper is to increase the exploration capability of a VNS by incorporating a population-based search such as a GA.

The experiments show that the single VNS offers solutions in different qualities. Sometimes it finds the optimal solution and sometimes it returns an awful solution. Combining VNS with a GA offers a more powerful algorithm than the VNS itself. The experimental results determine that the proposed MA always results good quality solutions and outperforms the simple VNS over all the test problems. In other words, the proposed MA benefits from the efficient search space exploration of a GA and the powerful exploitation strategy of a VNS. While the proposed MA is introduced to show the effectiveness of VNS search exploration, the results obtained by this method represents its performance over combinatorial optimization problems such that compared to the state-of-the-art methods, the proposed MA offers competitive solutions.

3.4.8 Complete Results

Table 3.16 represents the complete results of our proposed MA as well as single VNS on a well-known benchmark introduced by Lawrence [113].

Table 3.16: Results on the Lawrence [113] benchmark (la01-la40)

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
la01	10×5	**666**	VNS	**666**	**666**	0.00	**666**	**666**
			MA	**666**	**666**	0.00	**666**	**666**
la02	10×5	**655**	VNS	**655**	**655**	0.00	**655**	**655**
			MA	**655**	**655**	0.00	**655**	**655**

Table 3.16 – Continued on next page

Table 3.16: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
la03	10×5	**597**	VNS	**597**	**597**	0.00	**597**	**597**
			MA	**597**	**597**	0.00	**597**	**597**
la04	10×5	**590**	VNS	**590**	**590**	0.00	**590**	**590**
			MA	**590**	**590**	0.00	**590**	**590**
la05	10×5	**593**	VNS	**593**	**593**	0.00	**593**	**593**
			MA	**593**	**593**	0.00	**593**	**593**
la06	15×5	**926**	VNS	**926**	**926**	0.00	**926**	**926**
			MA	**926**	**926**	0.00	**926**	**926**
la07	15×5	**890**	VNS	**890**	**890**	0.00	**890**	**890**
			MA	**890**	**890**	0.00	**890**	**890**
la08	15×5	**863**	VNS	**863**	**863**	0.00	**863**	**863**
			MA	**863**	**863**	0.00	**863**	**863**
la09	15×5	**951**	VNS	**951**	**951**	0.00	**951**	**951**
			MA	**951**	**951**	0.00	**951**	**951**
la10	15×5	**958**	VNS	**958**	**958**	0.00	**958**	**958**
			MA	**958**	**958**	0.00	**958**	**958**
la11	20×5	**1222**	VNS	**1222**	**1222**	0.00	**1222**	**1222**
			MA	**1222**	**1222**	0.00	**1222**	**1222**
la12	20×5	**1039**	VNS	**1039**	**1039**	0.00	**1039**	**1039**
			MA	**1039**	**1039**	0.00	**1039**	**1039**
la13	20×5	**1150**	VNS	**1150**	**1150**	0.00	**1150**	**1150**
			MA	**1150**	**1150**	0.00	**1150**	**1150**
la14	20×5	**1292**	VNS	**1292**	**1292**	0.00	**1292**	**1292**
			MA	**1292**	**1292**	0.00	**1292**	**1292**

Table 3.16 – Continued on next page

Table 3.16: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
la15	20×5	**1207**	VNS	**1207**	**1207**	0.00	**1207**	**1207**
			MA	**1207**	**1207**	0.00	**1207**	**1207**
la16	10×10	**945**	VNS	**945**	**945**	0.00	**945**	**945**
			MA	**945**	**945**	0.00	**945**	**945**
la17	10×10	**784**	VNS	**784**	**784**	0.00	**784**	**784**
			MA	**784**	**784**	0.00	**784**	**784**
la18	10×10	**848**	VNS	**848**	**848**	0.00	**848**	**848**
			MA	**848**	**848**	0.00	**848**	**848**
la19	10×10	**842**	VNS	**842**	**842**	0.00	**842**	**842**
			MA	**842**	**842**	0.00	**842**	**842**
la20	10×10	**902**	VNS	**902**	906.2	1.58	907	907
			MA	**902**	906.0	2.11	907	907
la21	15×10	**1046**	VNS	**1046**	1059.1	12.79	1057.5	1085
			MA	**1046**	1049.5	4.09	1048.0	1058
la22	15×10	**927**	VNS	**927**	**927**	0.00	**927**	**927**
			MA	**927**	**927**	0.00	**927**	**927**
la23	15×10	**1032**	VNS	**1032**	**1032**	0.00	**1032**	**1032**
			MA	**1032**	**1032**	0.00	**1032**	**1032**
la24	15×10	**935**	VNS	**935**	941.9	4.51	982.0	984
			MA	**935**	941.9	3.93	942.5	946
la25	15×10	**977**	VNS	979	985.7	5.40	984.0	996
			MA	**977**	981.1	3.11	981.5	984
la26	20×10	**1218**	VNS	**1218**	**1218**	0.00	**1218**	**1218**
			MA	**1218**	**1218**	0.00	**1218**	**1218**

Table 3.16 – Continued on next page

Table 3.16: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
la27	20×10	**1235**	VNS	1244	1254.1	6.77	1255.0	1264
			MA	1238	1250.9	9.77	1249.5	1265
la28	20×10	**1216**	VNS	**1216**	1219.2	6.60	**1216**	1234
			MA	**1216**	1217.7	2.87	**1216**	1223
la29	20×10	**1152**	VNS	1169	1187.0	15.19	1185	1216
			MA	1163	1175.4	9.65	1178	1188
la30	20×10	**1355**	VNS	**1355**	**1355**	0.00	**1355**	**1355**
			MA	**1355**	**1355**	0.00	**1355**	**1355**
la31	30×10	**1784**	VNS	**1784**	**1784**	0.00	**1784**	**1784**
			MA	**1784**	**1784**	0.00	**1784**	**1784**
la32	30×10	**1850**	VNS	**1850**	**1850**	0.00	**1850**	**1850**
			MA	**1850**	**1850**	0.00	**1850**	**1850**
la33	30×10	**1719**	VNS	**1719**	**1719**	0.00	**1719**	**1719**
			MA	**1719**	**1719**	0.00	**1719**	**1719**
la34	30×10	**1721**	VNS	**1721**	**1721**	0.00	**1721**	**1721**
			MA	**1721**	**1721**	0.00	**1721**	**1721**
la35	30×10	**1888**	VNS	**1888**	**1888**	0.00	**1888**	**1888**
			MA	**1888**	**1888**	0.00	**1888**	**1888**
la36	15×15	**1268**	VNS	1291	1294.6	6.26	1291.5	1311
			MA	1281	1285.2	5.43	1281.0	1292
la37	15×15	**1397**	VNS	**1397**	1420.5	12.55	1422.5	1442
			MA	**1397**	1398.9	2.13	1398.0	1402
la38	15×15	**1196**	VNS	1208	1223.6	8.80	1225	1236
			MA	1208	1216.2	3.52	1217	1221

Table 3.16 – Continued on next page

Table 3.16: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
la39	15×15	**1233**	VNS	1241	1247.3	4.90	1247	1258
			MA	1241	1245.7	3.09	1246	1250
la40	15×15	**1222**	VNS	1233	1239.0	5.19	1239.5	1246
			MA	1233	1239.2	4.92	1239.5	1247

Acknowledgment

This work is made possible by a grant from the National Science Foundation and NSERC Discovery No. 327482.

3.5 Incorporating highly explorative methods to improve the performance of Variable Neighborhood Search

Abstract. Variable Neighborhood Search (VNS) is one of the most recently introduced metaheuristics. Although VNS is successfully applied on various problem domains, there is still some room for it to get improved. While VNS has an efficient exploitation strategy, it suffers from its inefficient solution space exploration. To overcome this limitation, VNS can be joined with explorative methods such as Evolutionary Algorithms (EAs) which are global population-based search methods. Due to its effective search space exploration, Differential Evolution (DE) is a popular EA which is a great candidate to be joined with VNS. In this article, two different DEs are proposed to be combined with VNS. The first DE uses explorative evolutionary operators and the second one is a Multi-Population Differential Evolution (MP-DE). Incorporating a number of sub-populations improves the population diversity and increases the chance of reaching to unexplored regions. Both proposed hybrid methods are evaluated on the classical Job Shop Scheduling Problems. The experimental results reveal that the combination of VNS with more explorative method is more reliable to find acceptable solutions. Furthermore, the proposed methods offer competitive solutions compared to the state-of-the-art hybrid EAs proposed to solve JSSPs.

3.5.1 Introduction

Optimization is an area of research where the goal is to optimize a system based on its input parameters. In general, an optimization problem is defined as finding the best set of input parameters to make a system as effective as possible. Optimization problems are categorized into two classes based on the type of their input parameters. The problems with continuous parameters are classified as global optimization problems, while the ones with discrete parameters are considered as combinatorial optimization problems. The focus of this paper is on a permutation problem which is a combinatorial optimization problem.

Various types of algorithms are proposed to solve optimization problems. A number of heuristics are presented to deal with very specific optimization problems. Although they perform well in their applicable domain, they cannot offer the same performance for related problem domains. Local search, for instance, is a heuristic which starts at an initial random solution, looks for a better solution in the neighborhood of its current solution, and returns the best found solution when it converges.

Due to the fact that heuristics are not successful in general, metaheuristics are presented which are general procedures to design heuristics for optimization problems. Metaheuristics are so-called complex heuristics. For example, there are a number of extended versions of simple local search such as Repeated Local Search (RLS) and Iterated Local Search (ILS). Since a local search might trap into local optimal regions, its execution for a number of times increases the chance of finding an acceptable solution. This strategy is known as RLS. ILS is an extended version such that it incorporates the best solutions found in previous executions. In fact, instead of starting from another random solution, ILS perturbs the recently found best solution and continues.

Variable Neighborhood Search (VNS) is one of the most recently introduced metaheuristics proposed by Mladenovic and Hansen [136]. VNS can also be considered as

an extended version of ILS. The idea of VNS is to incorporate a number of neighborhood structures and switch among them at the time of local search execution. VNS starts with searching locally with respect to one neighborhood structure and it switches to another one as soon as it converges. This strategy decreases the chance of immature convergence dramatically.

Due to its generality, VNS is successfully applied in various areas such as the Traveling Salesman Problem [55], the Open Vehicle Routing problem [57], the p-Median problem [56], and the Graph problem [24].

The performance of VNS is highly dependent to the definition of its neighborhood structures. Complement neighborhood structures provide VNS a powerful exploitation strategy. However, since VNS is a local search approach, there is still some chance of trapping into a local optimum by spinning in some previously investigated regions. Although a number of different strategies such as Parallel VNS [42], Multi-Start VNS [127], and Population-based VNS [205] are proposed to overcome this limitation, VNS cannot perform well as a global approach to deal with complex optimization problems; it suffers from its inefficient solution space exploration.

A more effective way to enhance VNS is to join it with a highly explorative method. In fact, the solution space exploration should be conducted by the joint method to find promising regions and VNS should be responsible for exploiting the promising regions. It is expected for this combination to have a great performance. In our recently published article [167], we incorporated a Genetic Algorithm (GA) [93] as the joint method. This combination which is a Memetic Algorithm (MA) [139] benefits a global population-based approach for its exploration and a powerful local search method as its exploitation strategy. The results of this combination show that the GA helps VNS to offer a better performance compared to the single VNS as a global method.

In this article, two more approaches are presented to improve the performance of a VNS. The first approach is to combine VNS with a Differential Evolution (DE) [191]. DE is a popular evolutionary method due to its powerful solution space exploration, and therefore it could be the best candidate to be combined with VNS. Although this combination is one of the bests in terms of incorporating both powerful exploration and exploitation strategies, there is still some chance of immature convergence. An approach to decrease this chance is to incorporate multiple populations for DE. Dividing the whole population into a number of sub-populations decreases the chance of trapping into local optimal regions. When a sub-population converges to a local optimum, it can recover itself by incorporating the knowledge migrated from other sub-populations. In other words, in this article two methods are proposed including the joint of DE and VNS and combination of Multi-Population Differential Evolution (MP-DE) and VNS.

The main contribution of this article is to represent the impact of different population-based searches with different levels of exploration on the performance of VNS. Since VNS is a powerful local search with a great exploitation strategy, it is expected that its combination with more explorative methods offer better performance. In order to evaluate the proposed methods, Job Shop Scheduling Problem (JSSP) is considered as our test bed. JSSP is a well-known combinatorial optimization problem which is considered as a complex problem based on the dependency of its input parameters to each other.

The remainder of this article is organized as follows. Subsection 3.5.2 briefly introduces VNS, followed by an introduction of DE and MP-DE in Subsections 3.5.3 and 3.5.4, respectively. The problem domain of JSSP is concisely defined in Subsection 3.5.5. The proposed methods are described in details in Subsection 3.5.6, followed by representing the experimental results in Subsection 3.5.7. Finally the last subsection illustrates the conclusion remarks.

3.5.2 Variable Neighborhood Search

A simple local search method determines a neighborhood area for each solution which is defined based on a neighborhood structure. It starts with a feasible solution which is usually selected randomly from the solution space. It then searches for a better solution by applying some moves with respect to the neighborhood structure. The main drawback of this simple structure is its immature convergence. If a local search is trapped in a local optimum, it spins around the local optimal region and never escape from that region. Therefore, instead of converging to the global optimum, it converges to a local one. This issue is more crucial in complex optimization problems, specially large ones, where the chance of trapping into local optima is higher.

To defeat this issue, VNS is introduced which incorporates a number of neighborhood structures. VNS starts searching with respect to the first neighborhood structure and as soon as it finds a local optimum it switches to the next neighborhood structure. This routine continues until an optimum with respect to all neighborhood structures is reached. Therefore, complement neighborhood structures provides a more powerful VNS.

It can be concluded that the idea of VNS comes from the following facts:

1. A local optimal solution in one neighborhood area is not necessarily a local optimum in another neighborhood area.

2. A global optimal solution is a local optimal solution for all possible neighborhood areas.

3. In general, local optimal solutions within one or more neighborhood structures are relatively close to each other.

Although the last one is not really a fact for all optimization problems, it is an empirical observation implying that a local optimal solution may carry some useful information about other local optimal solutions and even the global optimum.

3.5.3 Differential Evolution

Evolutionary Algorithms (EAs) are a class of algorithms inspired by the natural selection. EAs are global population-based search methods incorporating evolutionary operators including recombination, modification and selection. Differential Evolution (DE) is the most recently introduced EA proposed by Storn and Price [191] . DE which is designed to solve continuous optimization problems shows remarkable performance on various continuous optimization problems such as space trajectory optimization [201] and multi-area economic dispatch [183].

The key characteristic of DE which makes it a popular EA is its effective solution space exploration. DE incorporates a number of mathematical equations in order to highly explore the solution space. DE defines each solution as a d-dimensional vector of real numbers and uses its mutation and crossover equations to generate offspring. Although DE has a strong exploration strategy, due to its inefficient exploitation mechanism it is not able to perform well on complex optimization problems. Therefore, DE is a good candidate to be joined with a local search method. Recently, a number of successful hybrid DEs are reported in the literature such as the combination of a DE and an adaptive local search published by Noman and Iba [144].

As mentioned before, DE was designed to deal with continuous optimization problems. In order to apply it on combinatorial optimization problems, two approaches have been introduced. The first approach is to incorporate a transformation procedure such that each solution can be mapped from continuous domain to discrete one and vice versa. It should be noted that the mapping from continuous domain to discrete must be a many-to-one mapping ($n \rightarrow 1$). Incorporating this strategy provides a continuous domain for DE to be applied on. There are a number of successful applications of this approach on permutation optimization problems such as the ones published by Onwubolu and Davendra [151], Qian *et al.* [162], and Zhang and Wu [223]. The second approach is to modify DE's operators to be applicable in

a discrete domain. Modifying DE's operators makes it a specific algorithm for the problem domain and there is no guarantee to obtain the same performance. Two successful modified DEs are published by Pan *et al.* [153] and Wang *et al.* [203] to solve flow-shop scheduling. As presented in details in Subsection 3.5.6, the former approach is incorporated in our proposed methods.

All the components of DE are defined mathematically. The solution space S is a d-dimensional domain of real numbers and each solution s is determined as a d-dimensional vector:

$$S = D_0 \times D_1 \times ... \times D_{d-1} \tag{3.3}$$

$$s = [x_0, x_1, ..., x_{d-1}], \qquad x_j \in D_j \tag{3.4}$$

where x_j represents the value of solution s for dimension j ranging from 0 to $d-1$.

DE starts with an initial population of randomly generated solutions. The population is evolved over a number of generations. A solution within the population so-called target vector is denoted by the generation number and an index. For instance, target vector $X_{i,g}$ represents the i^{th} target vector of generation g:

$$X_{i,g} = [x_{0,i,g}, \quad x_{1,i,g}, \quad ..., \quad x_{d-1,i,g}] \tag{3.5}$$

where $x_{j,i,g}$ represents the value of target vector $X_{i,g}$ for dimension j ranging from 0 to $d-1$.

The recombination and modification operators of DE are also defined as mathematical formulae. The mutation operator applies on a target vector $X_{i,g}$ and generates a new vector which is called mutant vector denoted by $V_{i,g}$. The basic mutation equation is presented in Equation 3.6:

$$V_{i,g} = X_{r1,g} + F \times (X_{r2,g} - X_{r3,g}) \tag{3.6}$$

where $X_{r1,g}$, $X_{r2,g}$, and $X_{r3,g}$ are three different randomly selected target vectors. $X_{r1,g}$ is called the base vector and the other two are called perturbing vectors. F is a scale factor to determine how much to perturb the base vector.

There are various mutation equation incorporated by different researchers. Each equation is determined based on its base vector and the number of perturbations. The basic mutation equation illustrated in Equation 3.6, for instance, is called $DE/rand/1$ which means that the base vector is a randomly selected target vector and the perturbation is done only one time. Four more equations are presented by Price *et al.* [161] which are presented as follows:

$DE/rand/2$:

$$V_{i,g} = X_{r1,g} + F_1 \times (X_{r2,g} - X_{r3,g}) + F_2 \times (X_{r4,g} - X_{r5,g}) \tag{3.7}$$

$DE/best/1$:

$$V_{i,g} = X_{best,g} + F \times (X_{r1,g} - X_{r2,g}) \tag{3.8}$$

$DE/best/2$:

$$V_{i,g} = X_{best,g} + F_1 \times (X_{r1,g} - X_{r2,g}) + F_2 \times (X_{r3,g} - X_{r4,g}) \tag{3.9}$$

$DE/current-to-best/1$:

$$V_{i,g} = X_{i,g} + F_1 \times (X_{best,g} - X_{i,g}) + F_2 \times (X_{r1,g} - X_{r2,g}) \tag{3.10}$$

where $X_{r1,g}$, $X_{r2,g}$, $X_{r3,g}$, $X_{r4,g}$, and $X_{r5,g}$ are randomly selected target vectors, $X_{best,g}$ denotes the best found solution so far, and F, F_1, and F_2 are scale factors.

Equation 3.11 represents one more mutation equation which is introduced by Wisittipanich and Kachitvichyanukul [207]:

$DE/localbest/1$:

$$V_{i,g} = X_{ilbest,g} + F \times (X_{r1,g} - X_{r2,g}) \tag{3.11}$$

where $X_{r1,g}$ and $X_{r2,g}$ are randomly selected target vector, $X_{ilbest,g}$ denotes the local best solution of target vector $X_{i,g}$, and F is a scale factor.

These equations use different base vectors and different levels of perturbation. The more important point is that although Equations 3.10 and 3.11 incorporates the target vector $X_{i,g}$ either explicitly or implicitly, other equations generate the mutant vector $V_{i,g}$ regardless of the target vector $X_{i,g}$. Nevertheless, the crossover operator incorporates both the target vector $X_{i,g}$ and the mutant vector $V_{i,g}$ to generate a trial vector $Z_{i,g}$. The most popular crossover operator for DE is binomial crossover which is illustrated in Equation 3.12:

$$z_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } r_j \leq Cr \text{ or } j = j_{rand} \\ x_{j,i,g} & \text{otherwise} \end{cases} \tag{3.12}$$

where r_j is a random number uniformly distributed in interval $[0, 1)$ selected for the j^{th} dimension, Cr is the crossover probability which could be either fixed or dynamic, and j_{rand} is a randomly selected dimension to ensure that the trial vector $Z_{i,g}$ differs from target vector $X_{i,g}$ at least in one dimension.

After generating trial vectors, a selection function is incorporated by DE to select the better solutions for the next generation. Comparing the target vector $X_{i,g}$ and the trial vector $Z_{i,g}$, this function selects the one with the better objective value as a target vector for the next generation denoted by $X_{i,g+1}$.

$$X_{i,g+1} = \begin{cases} Z_{i,g} & \text{if } f(Z_{i,g}) \leq f(X_{i,g}) \\ X_{i,g} & \text{otherwise} \end{cases} \tag{3.13}$$

3.5.4 Multi-Population Differential Evolution

In Multi-Population Differential Evolution (MP-DE), the whole population is divided into a number of sub-populations and each sub-population is evolved by a local DE. Local DEs may communicate with each other in order to exchange knowledge. The main reason to incorporate a number of sub-populations instead of a single population is to decrease the chance of premature convergence. In fact, this strategy helps the method to maintain the population diversity and consequently it increases the chance of exploring unvisited regions.

There are various MP-DEs with different characteristics proposed in the literature. A parallel DE is proposed by Tasoulis *et al.* [196] in which the local DEs communicate with each other by migrating their best found solutions in a ring topology. The migrated solutions replace the randomly selected solutions in the destination sub-populations. Tasgetiren and Suganthan [195] proposed a MP-DE which incorporates a re-grouping strategy. The mutation strategy in this method selects the random target vectors from the whole population instead of just the corresponding sub-population.

In addition to the solution migration, there are a number of strategies for MP-DE in which only the DE parameters are exchanged among the sub-populations. Yu and Zhang [218] proposed a MP-DE in which in each generation the successful local DEs send their own parameters to other local DE to adjust theirs. The factor value in mutation equation and the crossover probability are considered as the exchanging control parameters in this method.

Furthermore, there are a number of strategies where no exchange occurs at all. The MP-DE proposed by Mendes and Mohais [132] which is called DynDE defines an acceptable distance between the best solutions of different sub-populations as a threshold. If such a distance gets smaller than the threshold, one of the sub-populations will be re-initialized.

3.5.5 Job Shop Scheduling Problem

In order to evaluate our proposed methods, the Job Shop Scheduling Problem (JSSP) is selected as our test bed. Being applicable in various research areas makes JSSP a well-known class of combinatorial optimization problems. JSSP is defined as the process of sequencing a number of jobs to be completed on a number of machines in order to utilize the resources as efficient as possible. The most popular objective in JSSP is to decrease the total time required to perform all the existing jobs. Therefore, the goal is to minimize the maximum completion time of all the jobs, called the *makespan*.

Garey *et al.* [65] proved that the JSSP problems including more than two machines are NP-complete which implies that there is no exact algorithm capable to find the optimal solution for all the sample problems in an acceptable time. Since JSSP is a complex permutation optimization problem which is still an open problem, it is a great case study to evaluate the new proposed methods.

Although there are various versions of JSSP represented in literature, in general, JSSP is defined by Baker [11] as a task of scheduling N jobs denoted by J_i to be processed on M machines denoted by m_k, where i is the job index and k is the machine index ranging from 1 to N and 1 to M, respectively. Each job consists of a number of operations which have to be processed in a pre-defined sequence. Each operation is denoted by O_{ij} where i is the job index and j is the operation index in the i^{th} job. In classical JSSP, there are a number of rules which may be partially shared with other types of JSSP. These rules are presented as follows:

1. Jobs are independent to each other and are available at the beginning.

2. There is no due date for any jobs.

3. All the jobs have the same number of operations which is equal to the number of applicable machines.

Table 3.17: A sample 3×3 classical JSSP

Jobs	Operation Index		
	O_1	O_2	O_3
J_1	m_3,3	m_1,3	m_2,2
J_2	m_1,2	m_2,3	m_3,2
J_3	m_1,2	m_2,3	m_3,2

Machines	Jobs		
	J_1	J_2	J_3
m_1	O_{12},3	O_{21},2	O_{31},2
m_2	O_{13},2	O_{22},3	O_{32},3
m_3	O_{11},3	O_{23},2	O_{33},2

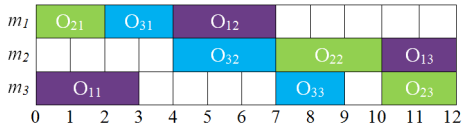


Figure 3.14: A sample schedule for the sample problem.

4. Each machine processes only one operation of a job which cannot be interrupted.

5. There is only one applicable machine for each operation such that there is no machine selection flexibility for operations.

6. The processing time of each operation on its applicable machine is known and the machine set up time and the movement time between machines are considered negligible.

A sample classical JSSP is illustrated in Table 3.17 in two different formats, namely job-based and machine-based. The sample problem is a 3×3 problem including 3 jobs to be processed on 3 machines. The table represents the applicable machine for each operation and its corresponding processing time. The second operation of the first job (O_{12}), for instance, has to be processed on the first machine (m_1) for 3 time units. A sample schedule for this problem is depicted in Fig. 3.14. The sample schedule has the makespan of 12 which is not the minimum makespan for this sample problem.

There are a number of important concepts in JSSP such that reaching the optimal solution without considering these concepts is almost impossible. The most important concepts are critical paths, critical blocks and critical operations. A critical path is the longest path of consecutive operations in a schedule starting from time zero and ending at the makespan. A schedule has at least one critical path. Critical paths

are important because they determine the makespan and the only way to decrease the makespan is to break all the critical paths. Therefore, in order to define efficient neighborhood structures for local search heuristics, breaking the critical paths should be considered as an approach to reach better solutions.

The operations on the critical paths are called critical operations. In other words, critical operations are such operations that any delay in their processing increases the makespan of the whole schedule. A critical operation may belongs to more than one critical paths. A sequence of adjacent critical operations on the same machine is called a critical block.

In the sample schedule presented in Fig. 3.14, for example, there are two critical paths including:

$$CriticalPaths: \begin{cases} O_{21} \prec O_{31} \prec O_{32} \prec O_{22} \prec O_{13} \\ O_{21} \prec O_{31} \prec O_{32} \prec O_{22} \prec O_{23} \end{cases}$$

Therefore, there are 6 critical operations in the sample schedule where 4 of them are on two different critical paths. The critical blocks are as follows.

$$CriticalBlocks: \begin{cases} m_1 : O_{21} \prec O_{31} \\ m_2 : O_{32} \prec O_{22} \prec O_{13} \\ m_3 : O_{23} \end{cases}$$

In fact, there is only one critical operation on the third machine and there is no sequence of critical operations. Therefore, considering it as a critical block does not have any effect on the process of optimization. The first and the last operations in a critical block are called block head and block rear, respectively, and others are called internal operations.

In addition to the critical operations, there are a number of terms which should be defined clearly. The following terms are used later to describe neighborhood struc-

tures. There are two different kinds of operation sequence in a schedule including job operation sequence and machine operation sequence. The former determines the sequence of operations of a job which is predefined in classical JSSP, while the latter one represents the sequence of operations which have to be processed on a specific machine. The adjacent operations, namely the previous operation and the next one, of an operation in a sequence are called its predecessor and successor operations, respectively. In fact, in a job operation sequence they are called Job-Predecessor operation and Job-Successor operation of an operation $O_{i,j}$ denoted by $JP(O_{i,j})$ and $JS(O_{i,j})$, respectively. $MP(O_{i,j})$ and $MS(O_{i,j})$ also represent Machine-Predecessor and Machine-Successor operations of an operation $O_{i,j}$, respectively. Since the operation sequence for each job is predefined, the following statements are always right, provided $O_{i,j+1}$ and $O_{i,j-1}$ exist:

$$JS(O_{i,j}) = O_{i,j+1}$$

$$JP(O_{i,j}) = O_{i,j-1}$$

Another valuable concept in JSSP is active schedule which is very useful to limit the solution space. An active schedule is defined by Croce *et al.* [43] as a schedule which does not have any operation that can be started earlier without delaying the process of another operation. Based on this definition, an optimal solution is more likely an active solution and even if it is not, it has an equivalent active schedule which is optimal as well. It should be noted here that an equivalent schedule is a schedule with the same makespan and the same critical paths which could have one or more different machine operation sequences. Each active schedule may have many equivalent non-active schedules and therefore the solution space of active schedules is much smaller than the main solution space. Consequently, exploring the active solution space is more efficient compared to searching in the whole solution space.

Various strategies are represented in order to incorporate the active schedule concept such as the gap reduction rule proposed by Hasan *et al.* [87] and the priori knowledge introduced by Becerra and Coello [14].

3.5.6 Proposed Methods

As mentioned before, VNS has a powerful exploitation strategy which suffers from its inefficient search space exploration. In order to enhance VNS, we published recently a combination of a GA and a VNS [167]. The results show that incorporating a population-based global search improves the results of VNS applications. In this article, more explorative methods are joined with VNS in order to illustrate their effects on the final results. It is expected that the results should be improved more compared to ones recently published [167]. Two methods are proposed in this article which are the combination of VNS with two different DEs. The first proposed method incorporates a simple DE while the second one benefits from a MP-DE which has a more explorative mechanism.

In our proposed methods, VNS is combined with EAs in order to improve its performance. As defined by Moscato [139], these combinations are considered as Memetic Algorithms (MAs). In general, a MA is defined as a combination of a population-based global search and a local search heuristic to solve optimization problems. The population-based global search provides an effective solution space exploration for a MA and the local search highly exploits the promising regions. Therefore, the performance of a MA should be higher than each of the combined methods. Various successful applications of MAs have been reported in the literature such as the MAs published by Gao *et al.* [63], Caumond *et al.* [28], and Chiang *et al.* [33] to deal with scheduling problems. Since both proposed methods in this article as well as the methods published previously [167] are instances of MA, in order to differentiate them

in this article they are called based on their combination as VNS+DE, VNS+MPDE and VNS+GA, respectively.

The details of both proposed methods are represented in the following sub-subsections starting with the description of solution representation in Sub-subsection 3.5.6.1. The neighborhood structures incorporated in our proposed VNS are described in Sub-subsection 3.5.6.2 followed by the definition of the genetic operators of the proposed DE and MPDE in Sub-subsection 3.5.6.3. Finally, the frameworks of both proposed methods are illustrated in Sub-subsection 3.5.6.4.

3.5.6.1 Solution Representation

As mentioned before, in order to apply DE on combinatorial optimization problems, there are two approaches. In our proposed methods, the transformation strategy is incorporated. Therefore, two different solution representations are considered; one in discrete domain and one in continuous. The proposed DE and MPDE deal with the representation in a continuous domain, while in order to evaluate each solution it should be transformed to a permutation domain to be considered as a JSSP solution.

Various representations with different characteristics are proposed for JSSP. Permutation with repetition representation introduced by Bierwirth [18] is one of the well-known representations. As an operation-based representation, this representation encodes a schedule based on the sequence of the operations into a string of digits. The length of the string equals to the total number of operations in a scheduling problem and each operation is denoted by its job index in the string. Therefore the operations of the same job are denoted by the same index. Considering the operation dependency within a job operation sequence, the operations with the same index are differentiated based on the occurrence number of the corresponding index. For instance, the second occurrence of the third job's index denotes the second operation of the third job (O_{32}). A sample solution for the sample problem illustrated in Table

3.17 is represented in permutation with repetition representation as follows:

$$\{1, 2, 3, 1, 3, 2, 3, 2, 1\}$$

The decoding of this sample solution results in the following operation sequence:

$$O_{11} \prec O_{21} \prec O_{31} \prec O_{12} \prec O_{32} \prec O_{22} \prec O_{33} \prec O_{23} \prec O_{13}$$

This operation sequence generates the following schedule which is depicted as a Gantt chart in Fig. 3.14.

$$m_1 : O_{21} \prec O_{31} \prec O_{12}$$

$$m_2 : O_{32} \prec O_{22} \prec O_{13}$$

$$m_3 : O_{11} \prec O_{33} \prec O_{23}$$

The key advantage of permutation with repetition representation is that all the possible permutations in this representation are feasible solutions. Therefore, the decoding mechanism is straightforward such that no repair mechanism is required. The main drawback of this representation is its inefficient many-to-one mapping ($n \rightarrow 1$). In fact, there could be a huge number of different permutations which are decoded to the same schedule. The schedule illustrated in Fig. 3.14 might be decoded from the following permutations (not limited to):

$$\{1, 2, 3, 1, 3, 2, 3, 2, 1\} \quad \{1, 2, 3, 3, 1, 2, 3, 2, 1\}$$

$$\{1, 2, 3, 3, 2, 1, 3, 2, 1\} \quad \{1, 2, 3, 3, 2, 3, 1, 2, 1\}$$

$$\{1, 2, 3, 3, 2, 3, 2, 1, 1\} \quad \{2, 1, 3, 3, 2, 3, 2, 1, 1\}$$

$$\{2, 3, 1, 3, 2, 3, 2, 1, 1\} \quad \{2, 3, 3, 1, 2, 3, 2, 1, 1\}$$

$$\{2,3,3,2,1,3,2,1,1\} \quad \{2,1,3,1,3,2,3,2,1\}$$

$$\{2,3,1,1,3,2,3,2,1\} \quad \{2,3,1,3,1,2,3,2,1\}$$

$$\{2,3,1,3,2,1,3,2,1\} \quad \{2,3,1,3,2,3,1,2,1\}$$

However, due to its coding and decoding efficiency, permutation with repetition representation is incorporated by various researchers. In our proposed method, we also use this representation.

As a representation in continuous domain, random key representation is selected for our proposed methods. In this representation, a solution is represented as a vector of real numbers. Each dimension in this vector corresponds to one operation. The following vector illustrates a sample solution for the sample problem described in Table 3.17.

$$\{0.49, 0.98, 0.38, 0.42, 0.73, 0.51, 0.48, 0.89, 0.63\}$$

The corresponding value for each operation in this sample solution is presented as follows. The minimum value, for instance, corresponds to O_{13} which means that this operation should be started first. Due to the operation dependency within a job operation sequence, O_{11} is processed first instead of O_{13}.

Operation	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
Job Index	1	1	1	2	2	2	3	3	3
Random Key	0.49	0.98	0.38	0.42	0.73	0.51	0.48	0.89	0.63

In order to transform a solution from random key representation to permutation with repetition representation, Smallest Position Value (SPV) rule [194] is incorporated, in which the job indices should be sorted based on their corresponding random key values ascendingly which is as follows:

Random Key	0.38	0.42	0.48	0.49	0.51	0.63	0.73	0.89	0.98
Job Index	1	2	3	1	2	3	2	3	1

The sorted string of job indices is the equivalent solution in the discrete domain which is

$$\{1, 2, 3, 1, 2, 3, 2, 3, 1\}$$

where its encoded operation sequence is presented as follows.

$$O_{11} \prec O_{21} \prec O_{31} \prec O_{12} \prec O_{22} \prec O_{32} \prec O_{23} \prec O_{33} \prec O_{13}$$

In addition to the fitness evaluation function, VNS works in the discrete domain as well. Therefore, the results of VNS should be transformable to the continuous domain. In order to do so, each solution in discrete domain updates its random key when a swapping or insertion occurs, accordingly.

3.5.6.2 Neighborhood Structures

In order to be able to compare the proposed methods with the published ones [167], the same neighborhood structures are incorporated in this article. In our recently published methods [167], two neighborhood structures $N4$ and $N5$ were incorporated. These structures are two of the six popular neighborhood structures reviewed by Blazewicz *et al.* [20] which are denoted by $N1$ to $N6$ by the authors. These neighborhood structures are described briefly as follows:

- Neighborhood Structure $N1$: This structure defines the largest neighborhood area by considering the swapping of two adjacent critical operations as a valid move [200].
- Neighborhood Structure $N2$: This neighborhood structure considers the swapping of two critical operations p and q as a valid move if either p is a block head or q is a block rear. In order to improved the neighbor solutions two additional moves are also considered which include the swapping of $MP(JP(p))$ and $JP(p)$ and the swapping of $JS(q)$ and $MS(JS(q))$ [129].

- Neighborhood Structure $N3$: Considering p and q are two adjacent critical operations, this structure looks into all permutations of three operations $\{MP(p), p, q\}$ as well as three operations $\{p, q, MS(q)\}$ in which p and q are swapped [50]. Since the neighborhood area of this structure is very large, a limited version is introduced which is called $N3'$ in which either p or q should be a block end.

- Neighborhood Structure $N4$: Moving an internal operation to the very beginning or to the very end of a block is considered as a valid move in this structure [50].

- Neighborhood Structure $N5$: This structure only swaps the first two operations or the last two operations of a critical block which makes the smallest neighborhood area [146].

- Neighborhood Structure $N6$: This structure is an extension of all previously described neighborhood structures. The valid moves in this structure include moving q right before p if $JP(p)$ belongs to the critical path and moving p right after q if $JS(q)$ belongs to the critical path, given p and q as two critical operations on a critical block [12].

As mentioned above, only two neighborhood structures $N4$ and $N5$ are considered in our proposed methods without any pre-processing or post-processing procedures. The structure of the incorporated VNS is exactly the same as the one recently published [167] which includes a *Shake* method followed by a *LocalSearch* method. The *Shake* method consists of four consecutive random moves with respect to the following neighborhood structures.

$$Shake : N5 \rightarrow N4 \rightarrow N5 \rightarrow N4$$

The *LocalSearch* method incorporates both the regular neighborhood search and the nested neighborhood search. In regular one, both neighborhood structures $N4$ and $N5$ are considered while for the nested search only the smaller neighborhood ($N5$) is incorporated. In terms of local search strategy, in order to save the computational time, the first improvement strategy is incorporated instead of the best improvement one [167].

3.5.6.3 Genetic Operators

As DE is incorporated in order to increase the exploration power, it should be designed using more explorative evolutionary operators. The most explorative mutation strategy is $DE/rand/1$ which is illustrated in Equation 3.6. Incorporating this strategy in our experiment declares that this strategy is not useful for JSSP because it does not keep any information regarding previous generations. Other strategies presented in Equations 3.7 through 3.11 are more exploitative than explorative. Therefore, a new mutation strategy is introduced in this article which is called $DE/current/1$ presented as follows:

$$V_{i,g} = X_{i,g} + F \times (X_{r1,g} - X_{r2,g}) \tag{3.14}$$

where $X_{i,g}$ is the current target vector, $X_{r1,g}$ and $X_{r2,g}$ are two different randomly selected target vectors, and F is a scale factor.

Although this strategy incorporates the existing target vectors, it is more explorative compared to other strategies while benefiting from the evolved solutions instead of just random ones. The combination of both $DE/current/1$ and $DE/rand/1$ is also evaluated on some JSSP benchmark problems. In this combination, for a top portion of a population $DE/current/1$ is incorporated while for the rest of the population $DE/rand/1$ is used. It was expected to obtain good performance for this combination, but it returns poor results. Therefore, only $DE/current/1$ is incorporated as

the mutation strategy in the proposed DE and MP-DE with a small difference. In order to provide a more explorative mutation strategy, a modification is considered for MP-DE which will be described later in this subsection. To incorporate more information of the best found solution so far in case of no improvement for a number of generations, the mutation strategy switches to $DE/current-to-best/1$ illustrated in Equation 3.10.

For the crossover operator, the binomial crossover illustrated in Equation 3.12 is considered for both proposed methods exactly the same. Like for the crossover operator, both proposed methods incorporate the same selection function which is presented in Equation 3.13. It should be noted here that in order to consider more information in selection procedure, instead of a simple evaluation function, a Priority-Based Fitness Function (PBFF) [169] is incorporated. The idea of this function comes from the fact that in simple evaluation functions when a tie happens the winner is selected arbitrarily while it could be selected based on a comparison with respect to another factor. In PBFF, a number of factors are defined with different priorities and then the comparison procedures considers lower priority factors in case of ties in higher priority factors.

In our proposed methods, a PBFF is incorporated such that the first priority factor is the makespan and the second one is the number of critical machines. Based on this fitness function, if two schedules have the same makespan, the selection function selects the one with the lower number of critical machines. If they have the same number of critical machines, one of them will be selected randomly.

3.5.6.4 Frameworks

The first proposed method is a combination of DE and VNS, so-called VNS+DE. The framework of the proposed DE+VNS is illustrated in Fig. 3.15 which starts with an initial population of random solutions (line 03). *PopSize* denotes the size of the

PROCEDURE: VNS+DE
INPUT: Algorithm Parameters and Problem Specification
OUTPUT: Optimal or Near-Optimal Solutions

```
01  BEGIN
02      g ← 0
03      Generate an initial population P_0 with
                    PopSize random solutions
04      REPEAT
05          FOR ( Each Target Vector X_{i,g} of P_g)
06              V_{i,g} ← Mutate(X_{i,g})
07              Z_{i,g} ← Crossover(X_{i,g}, V_{i,g})
08              X_{i,g+1} ← Selection(X_{i,g}, Z_{i,g})
09          END FOR
10          FOR ( Each Top Target Vector X_{i,g+1} )
11              L_{i,g+1} ← VNS(X_{i,g+1})
12          END FOR
13          Construct population P_{g+1}
14          g ← g + 1
15      UNTIL (termination criteria are met)
16      Output The Best Found Solution
17  END
```

Figure 3.15: The framework of the VNS+DE

population which is fixed during the evolution. In each generation, first DE evolves the population and then VNS applies on a top best portion of the evolved population. DE incorporates its mutation, crossover and selection operators to explore the solution space (lines 05 through 09). Then the exploitation of the promising regions (the neighborhood of the top best solutions) is conducted by VNS (lines 10 through 12). The top best solutions are determined using *TopBest* parameter which is fixed for all generations. In our proposed method, in order to find out the top best solutions the whole population is sorted in each generation. After the application of VNS, its results combined with the rest of the population provides the population for the next generation denoted by P_{g+1} (line 13). This routine continues until the criteria are met (line 15). A maximum number of generations is considered as our termination criterion denoted by *MaxGen*.

Table 3.18: Parameters of the Proposed Methods

Parameter	Value
$MaxGen$	200
$PopSize$	1000
$TopBest$	100
$SubPopsNo$	5
$Mutate(X_{i,g})$	Equations 3.14 and 3.10
$Crossover(X_{i,g}, V_{i,g})$	Equation 3.12
$Selection(X_{i,g}, Z_{i,g})$	Equation 3.13 with a PBFF

The second proposed method is a combination of a VNS and a Multi-Population Differential Evolution denoted by VNS+MPDE. Although this method is similar to the proposed VNS+DE, it benefits from the concept of multiple populations. In this method, first the whole population is divided into a number of sub-populations denoted by $SubPopsNo$ and then each sub-population is evolved by a local DE.

As mentioned before, the local DEs incorporate both $DE/current/1$ and $DE/current-to-best/1$ illustrated in Equations 3.14 and 3.10, respectively. The point is that in order to enhance the solution space exploration the random target vectors in both mutation strategies for each sub-population is selected from other sub-populations. This mechanism increases the population diversity and improves the chance of reaching unexplored regions.

3.5.7 Results

Both proposed algorithms are implemented using the *java* programming language version 1.6.0.18 and the experiments are conducted on a system with $Intel(R)$ $Core(TM)2Quad$ $2.50GHz$ CPU and $8.00GB$ RAM. The algorithm parameters presented in Table 3.18 are adjusted through extensive experiments. A population of 1000 individuals with the top 100 best individuals are evolved for 200 generations.

In order to evaluate the performance of the both proposed methods, classical JSSP is considered as the our test bed. The data set introduced by Lawrence [113] is one

of the well-known benchmark for classical JSSP which includes 40 different problems with various sizes. Both proposed methods are applied on all the 40 test problems for 50 independent runs. Both proposed methods find the optimal solution for 28 test problems in every run. The results of the experiments on the rest of the problems which are more challenging are presented in Table 3.19.

Table 3.19: Results on the challenging problems of LA Benchmark

Problem	Method	Best	Average	SD	Median	Worst	Hit
$LA20$	VNS	**902**	906.72	1.13	907.0	907	2
	VNS + GA	**902**	906.46	1.49	907.0	907	3
10×10	VNS + DE	**902**	906.30	1.63	907.0	907	3
902	VNS + MPDE	**902**	905.98	1.95	907.0	907	7
$LA21$	VNS	**1046**	1058.18	7.96	1056.0	1077	4
	VNS + GA	**1046**	1053.28	3.85	1054.0	1060	8
15×10	VNS + DE	**1046**	1049.86	3.49	1049.0	1056	16
1046	VNS + MPDE	**1046**	1049.40	3.11	1048.0	1054	18
$LA24$	VNS	**935**	944.42	5.60	944.0	960	1
	VNS + GA	**935**	943.00	3.52	943.0	949	1
15×10	VNS + DE	**935**	941.16	2.44	941.0	945	2
935	VNS + MPDE	**935**	940.98	2.38	941.0	944	3
$LA25$	VNS	978	984.62	4.38	984.0	1000	2
	VNS + GA	**977**	982.48	2.22	983.0	987	2
15×10	VNS + DE	**977**	981.18	2.21	982.0	984	3
977	VNS + MPDE	**977**	979.50	1.88	979.0	983	7
$LA27$	VNS	1238	1251.44	8.66	1251.0	1265	3
	VNS + GA	1238	1251.60	6.58	1252.0	1263	4
20×10	VNS + DE	1236	1248.58	6.74	1250.0	1262	1
1235	VNS + MPDE	**1235**	1248.12	6.78	1250.0	1256	3

Table 3.19 – Continued on next page

Table 3.19: **(continued from previous page)**

Problem	Method	Best	Average	SD	Median	Worst	Hit
$LA28$	VNS	**1216**	1219.76	6.49	**1216**.0	1234	29
	VNS + GA	**1216**	1218.72	3.61	1217.0	1228	23
20×10	VNS + DE	**1216**	1216.68	1.56	**1216**.0	1223	37
1216	VNS + MPDE	**1216**	1216.26	0.60	**1216**.0	1218	41
$LA29$	VNS	1169	1188.26	10.96	1190.5	1210	1
	VNS + GA	1163	1180.88	7.29	1180.5	1194	1
20×10	VNS + DE	1163	1177.12	6.47	1177.5	1190	2
1152	VNS + MPDE	1163	1176.86	6.69	1178.0	1187	2
$LA36$	VNS	1281	1291.88	5.77	1291.0	1309	7
	VNS + GA	1277	1289.48	6.35	1291.0	1302	4
15×15	VNS + DE	1274	1287.78	6.12	1291.0	1298	1
1268	VNS + MPDE	1271	1284.70	5.94	1286.0	1291	1
$LA37$	VNS	**1397**	1420.22	11.08	1420.0	1442	1
	VNS + GA	**1397**	1403.40	8.07	1401.0	1433	2
15×15	VNS + DE	**1397**	1409.46	6.99	1408.5	1421	5
1397	VNS + MPDE	**1397**	1408.74	6.08	1408.0	1418	5
$LA38$	VNS	1207	1229.34	16.72	1229.5	1265	1
	VNS + GA	1207	1220.52	5.76	1219.0	1237	1
15×15	VNS + DE	1206	1215.88	3.62	1216.0	1225	1
1196	VNS + MPDE	1202	1215.38	3.76	1216.0	1220	1
$LA39$	VNS	1240	1246.76	4.11	1248.0	1256	4
	VNS + GA	1240	1248.40	3.17	1249.0	1253	5
15×15	VNS + DE	1238	1247.52	2.60	1249.0	1249	2
1233	VNS + MPDE	1238	1246.80	3.25	1248.0	1249	3

Table 3.19 – Continued on next page

Table 3.19: **(continued from previous page)**

Problem	Method	Best	Average	SD	Median	Worst	Hit
$LA40$	VNS	1231	1240.98	4.83	1242.0	1249	1
	VNS + GA	1230	1238.94	5.40	1239.0	1252	2
15×15	VNS + DE	1228	1235.62	3.49	1234.5	1242	1
1222	VNS + MPDE	1228	1235.26	3.99	1234.0	1242	2

In order to have a fair comparison with the recently published methods [167], these methods including the simple VNS and the combination of the VNS with a GA are also re-applied on the benchmark problems for 50 independent runs. The results of these experiments are also presented in Table 3.19 which is slightly different than the ones presented in the previously published article. This small difference is due to the fact that the published results are experimented for 10 independent runs while the new ones are averaged over 50 runs.

As presented in Table 3.19, the simple VNS can find the optimal solution for 5 challenging test problems out of 12, while the VNS+GA and the VNS+DE are able to find the optimal solution for the same 6 problems, and the VNS+MPDE is able to offer the optimal solution for 7 test problems. In total, the simple VNS, the VNS+GA, the VNS+DE and the VNS+MPDE find the optimal solutions for 33, 34, 34 and 35 test problems out of 40, respectively.

In addition to offering the optimal solution for more test problems, the VNS+MPDE offers more acceptable solutions compared to the other methods. In other words, not only the VNS+MPDE can find better solutions with a lower makespan, but also the range of its results for 50 independent runs is smaller compared to the result ranges of other methods. The average, median and worst results presented in Table 3.19 which are calculated over the 50 runs proves this

claim. For test problem $la38$, for instance, not only the best solution is found by the VNS+MPDE, but also the range of its obtained solutions in the 50 runs is from 1202 to 1220 (19 time units wide) which is thinner compared to this range for other methods specially the simple VNS which is from 1207 to 1265 (59 time units wide).

In order to compare all the four method over all the 12 challenging test problem, the Error Rate (ER) parameter is incorporated which is defined as follows:

$$ER = \frac{C - LB}{LB} \times 100\%$$

where C is the makespan of a solution found by the methods and LB is the best-known solution. The ER values for the best, average and worst solutions for each methods are calculated and represented in Table 3.20. In this table, the zero errors are emphasized with bold face and the average ERs over all the 12 test problems are presented in the last three rows. Comparing all the ERs illustrates that the proposed VNS+MPDE is the most reliable method to find acceptable solutions. The next reliable method is the proposed VNS+DE, and the least reliable one is the simple VNS. This statement is represented in a graph in Fig. 3.16 more clearly.

This comparison proves our hypothesis mentioning that combining a VNS with more explorative methods offers better solutions. Explorative methods increases the chance of finding the promising regions in a solution space which can be exploited by the VNS. Exploiting more promising regions improves the final results which increases the reliability of finding acceptable solutions.

In order to evaluate the performance of both proposed methods, they are also compared with the state-of-the-art methods in this area . Three recently published methods are considered for this comparison including a hybrid EA proposed by Zobolas *et al.* [225], our published MA [169] and a hybrid GA proposed by Qing-dao-er-ji and Wang [45]. The hybrid EA incorporates VNS as its local search method and uses

Table 3.20: Comparison of the best, average and worst results of the proposed methods.

Problem		VNS	VNS+GA	VNS+DE	VNS+MPDE
	Best	**0.00%**	**0.00%**	**0.00%**	**0.00%**
$LA20$	Average	0.52%	0.49%	0.48%	0.44%
	Worst	0.55%	0.55%	0.55%	0.55%
	Best	**0.00%**	**0.00%**	**0.00%**	**0.00%**
$LA21$	Average	1.16%	0.70%	0.37%	0.33%
	Worst	2.96%	1.34%	0.96%	0.76%
	Best	**0.00%**	**0.00%**	**0.00%**	**0.00%**
$LA24$	Average	1.01%	0.86%	0.66%	0.64%
	Worst	2.67%	1.50%	1.07%	0.96%
	Best	0.10%	**0.00%**	**0.00%**	**0.00%**
$LA25$	Average	0.78%	0.56%	0.43%	0.26%
	Worst	2.35%	1.02%	0.72%	0.61%
	Best	0.24%	0.24%	0.08%	**0.00%**
$LA27$	Average	1.33%	1.34%	1.10%	1.06%
	Worst	2.43%	2.27%	2.19%	1.70%
	Best	**0.00%**	**0.00%**	**0.00%**	**0.00%**
$LA28$	Average	0.31%	0.22%	0.06%	0.02%
	Worst	1.48%	0.99%	0.58%	0.16%
	Best	1.48%	0.95%	0.95%	0.95%
$LA29$	Average	3.15%	2.51%	2.18%	2.16%
	Worst	5.03%	3.65%	3.30%	3.04%
	Best	1.03%	0.71%	0.47%	0.24%
$LA36$	Average	1.88%	1.69%	1.56%	1.32%
	Worst	3.23%	2.68%	2.37%	1.81%
	Best	**0.00%**	**0.00%**	**0.00%**	**0.00%**
$LA37$	Average	1.66%	0.46%	0.89%	0.84%
	Worst	3.22%	2.58%	1.72%	1.50%
	Best	0.92%	0.92%	0.84%	0.50%
$LA38$	Average	2.79%	2.05%	1.66%	1.62%
	Worst	5.77%	3.43%	2.42%	2.01%
	Best	0.57%	0.57%	0.41%	0.41%
$LA39$	Average	1.12%	1.25%	1.18%	1.12%
	Worst	1.87%	1.62%	1.30%	1.30%
	Best	0.74%	0.65%	0.49%	0.49%
$LA40$	Average	1.55%	1.39%	1.11%	1.09%
	Worst	2.21%	2.45%	1.64%	1.64%
	Best	0.42%	0.34%	0.27%	0.22%
Avgerage ER	Average	1.44%	1.13%	0.97%	0.91%
	Worst	2.82%	2.01%	1.57%	1.34%

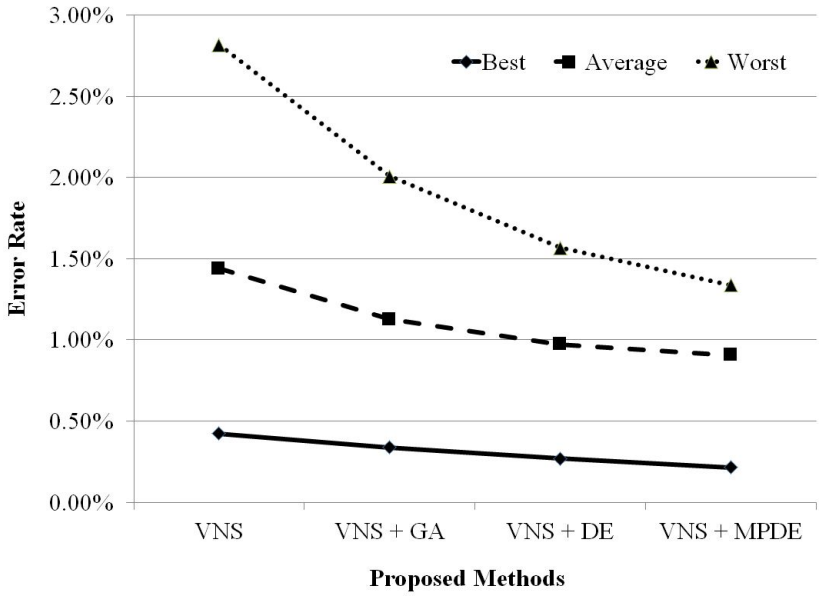


Figure 3.16: Comparison of the best, average and worst results of the proposed methods.

DE to generate a valuable initial population. Our recently published MA incorporates a GA combined with a local search heuristic. The hybrid GA introduces new genetic operators and incorporates a new local search method specifically designed for JSSP.

The results of this comparison are presented in Table 3.21 which illustrates the ER values in brackets, the optimal solutions in bold face, and the best solutions with an asterisk (*). Since the hybrid GA [45] cannot find the optimal solution for test problem $LA22$, this problem should be added to the 12 challenging test problems. Therefore for this comparison, 13 test problems are considered and the last row in the table illustrates the average ERs over the 13 test problems. It should be noted here that in the hybrid EA [225] the result for the test problem $LA20$ is not reported. Therefore, excluding this test problem the average ER equals to 0.27%, while considering that it may find the optimal solution for this test problem decreases its average ER to 0.25%. However, our proposed VNS+MPDE offers the best average ER.

Table 3.21: Comparison with different hybrid EAs proposed recently to solve JSSP

Prob.	BK	hEA (2009) [225]	MA (2012) [169]	hGA (2012) [45]	Proposed VNS+DE	Proposed VNS+MPDE
LA20	**902**	-	907 (0.55%)	907 (0.55%)	**902** (0.00%)	**902** (0.00%)
LA21	**1046**	**1046** (0.00%)	1053 (0.67%)	**1046** (0.00%)	**1046** (0.00%)	**1046** (0.00%)
LA22	**927**	**927** (0.00%)	**927** (0.00%)	935 (0.86%)	**927** (0.00%)	**927** (0.00%)
LA24	**935**	**935** (0.00%)	941 (0.64%)	953 (1.93%)	**935** (0.00%)	**935** (0.00%)
LA25	**977**	**977** (0.00%)	984 (0.72%)	981 (0.41%)	**977** (0.00%)	**977** (0.00%)
LA27	**1235**	1236 (0.08%)	1256 (1.70%)	1236 (0.08%)	1236 (0.08%)	**1235** (0.00%)
LA28	**1216**	1224 (0.66%)	1223 (0.58%)	**1216** (0.00%)	**1216** (0.00%)	**1216** (0.00%)
LA29	**1152**	1160* (0.69%)	1187 (3.04%)	1160* (0.69%)	1163 (0.95%)	1163 (0.95%)
LA36	**1268**	**1268** (0.00%)	1276 (0.63%)	1287 (1.50%)	1274 (0.47%)	1271 (0.24%)
LA37	**1397**	1408 (0.79%)	1401 (0.29%)	1407 (0.72%)	**1397** (0.00%)	**1397** (0.00%)
LA38	**1196**	1202* (0.50%)	1208 (1.00%)	**1196** (0.00%)	1206 (0.84%)	1202* (0.50%)
LA39	**1233**	**1233** (0.00%)	1240 (0.57%)	**1233** (0.00%)	1238 (0.41%)	1238 (0.41%)
LA40	**1222**	1229 (0.57%)	1233 (0.90%)	1229 (0.57%)	1228* (0.49%)	1228* (0.49%)
Avg. ER		0.27% (0.25%)	0.87%	0.56%	0.27%	0.22%

The proposed VNS+MPDE can find the optimal solution for 8 test problems out of 13 which is the highest number compared to the other methods. Furthermore, this method offers the best solution for two more test problems. The results of the proposed VNS+MPDE is not better than the others just for 3 test problems $LA29$, $LA36$ and $LA39$. Among these five methods, only three methods can find solutions with ER less than 1 for all test problems which include the hybrid EA [225] and the both proposed methods. Although there is not a significant difference between the proposed methods and the hybrid EA, our proposed VNS+MPDE slightly outperforms all the mentioned state-of-the-art methods.

3.5.8 Conclusions

VNS is one of the most recent metaheuristics proposed to deal with optimization problems. Although it has been successfully applied in various research areas, its mechanism is highly exploitative and it lacks an efficient search space exploration strategy. Recently, we published an article claiming that VNS is highly a local search metaheuristic and in order to have a great performance as a global search it should

be combined with an explorative method [167]. In the published article, VNS is combined with a GA and results show that this combination outperforms the simple VNS. In this article, two more explorative methods are combined with VNS in order to improve the final results. The first combined method is DE which is more explorative compared to GA. The second one is a more explorative method which is a DE incorporating multiple populations. Dividing the whole population into a number of sub-populations decreases the chance of premature convergence and encourages the method to look into different regions.

Both proposed combinations are evaluated on the same benchmark. The experiments show that the proposed VNS+MPDE outperforms all the other combinations by offering highly acceptable solutions in every run. While the incorporated VNS is the same for all the combinations, as expected more explorative methods offer better solutions. Compared to the single VNS which returns solutions with different qualities, the joined VNS with more explorative methods show higher reliability in terms of finding acceptable solutions. Although the proposed methods are just introduced to show the effects of combining explorative methods with VNS in its performance, they offer competitive solutions compared to the state-of-the-art methods.

Acknowledgment

This work is made possible by a grant from the National Science Foundation and NSERC Discovery No. 327482.

3.6 Conclusions

This chapter presents my published MAs to deal with JSSPs. These MAs include different combinations of population-based global search methods and local search heuristics. The MAs presented in Sections 3.2 and 3.3 are the combinations of GA and local search heuristics, while Sections 3.4 and 3.5 present a combination of a GA and a VNS and a combination of a DE and a VNS, respectively.

In addition to the proposed MAs, a new representation for JSSP is introduce in Section 3.2. The proposed representation which is called Machine Operation Lists (MOL) outperforms the most well-known existing representations. Since this representation is designed only for classical JSSPs, the proposed MA incorporating this representation is applied only on the classical JSSPs.

Furthermore, Section 3.3 introduces a new fitness function in addition to the new local search heuristic for JSSPs. The new fitness function which is called Priority-Based Fitness Function considers different parameters of a solution with different priority levels instead of the only fitness value used in traditional fitness functions. The experiment results show that incorporating the proposed fitness function improves the final solutions by saving the individuals which have better characteristics.

The proposed MA represented in Section 3.3 is a general method applicable on different types of JSSPs. Its successful applications on classical, flexible and multi-objective flexible JSSPs have been reported.

As mentioned before, VNS is one of the most recent metaheuristics which has been successfully applied in various research areas. In spite of its successful applications, it suffers from its inefficient search space exploration. In other words, it is more likely a local search heuristic, not a global search method. Therefore, the combination of VNS with a population-based global search is expected to be a more efficient method. The MA represented in Section 3.4 is a combination of a GA and a VNS which significantly outperforms the VNS as a single method. Moreover, this combination is also improved

by incorporating a more explorative method instead of a GA. The improved method which is described in Section 3.5 is a combination of a DE and the same VNS. This combination is also improve by incorporating a Multi-Population DE (MPDE) which is a more explorative method compared to the traditional DE.

The proposed combinations of VNS are compared with the state-of-the-art evolutionary methods. The comparison results reveal that although there is not a significant difference between the performance of the proposed methods and that of the hybrid EAs, my proposed MAs and specially the combination of VNS and MPDE slightly outperform the state-of-the-art methods.

In order to improve the proposed methods, a number of approaches have been considered as the future directions for this research including:

- As mentioned before, the proposed MOL representation is designed only for classical JSSPs. Therefore, its corresponding MA is only applicable on a restricted problem domain. Generalizing the proposed representation is considered as a future direction in order to make the proposed MA applicable in a wide range of JSSP problem domains including flexible JSSPs and multi-objective JSSPs.

- Evaluating the applications of the general MA described in Section 3.3 on other types of JSSPs, specially Stochastic Flexible Job Shop Scheduling, is also considered as the future work.

- As it has been shown that VNS is more likely a local search heuristic, a future direction is considered as improving its search space exploration by adding some features or manipulating its framework instead of hybridizing with more explorative methods.

Chapter 4

Multi-Population Cultural Algorithm: A Survey on Architectures

4.1 Introduction

A Cultural Algorithm (CA) is an Evolutionary Algorithm (EA) which was developed by Reynolds (1994) [175]. CA is successfully applied in various areas such that there are a lot of research efforts done in using CA. CA extracts knowledge from its population during evolution and then incorporates the extracted knowledge to amend its search mechanism. Incorporating knowledge helps CA to find better solutions as well as to improve its convergence rate.

The architecture of CA is demonstrated in Figure 4.1. CA consists of two different spaces including population space and belief space. The population space contains the individuals and works like all other EAs. The individuals are being evolved in this space to find the optimal solution. Belief space is considered as a storage space for the extracted knowledge. The knowledge is recorded in, updated, and accessed from

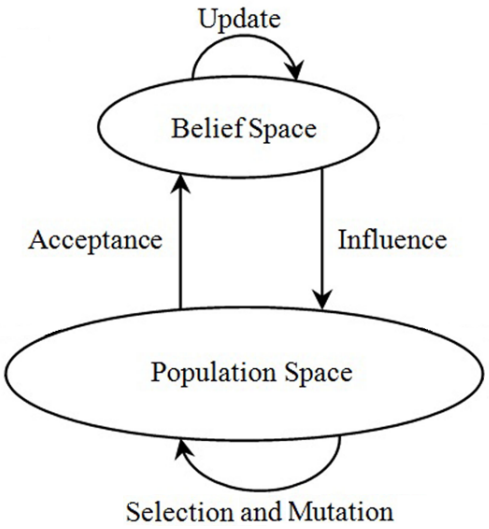


Figure 4.1: CA Architecture

the belief space. The population space communicates with belief space through two links, namely acceptance function and influence function. The population space sends its best individuals to the belief space through the acceptance function. The belief space updates its own knowledge and passes the updated knowledge to population space through the influence function. Then the received knowledge is used to generate offspring from current population. It is possible to say that the CA works like other EAs such as Genetic Algorithm (GA), but in addition it uses the knowledge to control the genetic operators to guide the evolution direction.

Almost all of the existing CAs in the literature use a single population. A CA incorporating multiple populations concept was first introduced by Digalakis and Margaritis (2002) [51] to schedule electrical generators. Afterwards, other researcher from different areas applied this concept in their problem domain. In this survey, a CA which uses multiple populations is called Multi-Population Cultural Algorithm (MP-CA).

A number of strategies are proposed to benefit multi-population concept in a CA. Using a number of population spaces with a shared belief space, for instance, is one of the strategies. Moreover, a CA with multiple populations needs more parameters

to be adjusted such as the number of sub-populations, their communication type, communication rate, and updating knowledge. The number of sub-populations is fixed in some methods, while in others there is a dynamic routine which sets the number of sub-populations.

In this survey, all the published methods incorporating MP-CA are discussed. By the knowledge of the author and based on the extensive research in this area, there is no more published methods to be considered. As mentioned before, the earliest MP-CA was published in 2002, but the majority of the other existing MP-CAs have been published in the last two years. It means that recently there is more interest to use a CA with multiple populations and the area of MP-CA is growing so fast.

The published MP-CAs appeared in various areas, mainly in Optimization Problems. These methods can be classified into the following categories based on their problems domain:

1. Optimization Problems (2002).

2. Supply Chain Management (2006).

3. Multimodal Optimization Problems (2007).

4. Neurofuzzy Inference System Training (2008).

5. Interactive Applications (2009).

The number in the brackets shows the year when the earliest paper in each category was published. For instance, the first MP-CA in the ares of multimodal optimization problems was published in 2007.

The structure of this survey is as follows. The published MP-CAs are described based on their problems domain starting with the area of optimization problems in section 4.2. The method in supply chain management is presented in section 4.3, followed by the methods in multimodal optimization problems in section 4.4. Section

4.5 describes the MP-CAs proposed to train neurofuzzy inference systems and section 4.6 presents the interactive applications of MP-CAs. Finally, the last section presents the concluding comments and future direction.

4.2 Optimization Problems

The field of Optimization Problems (OP) is the first area in which MP-CA was introduced. The optimization problem also known as the constraint optimization problem is a class of problems for which there is an optimization function required to satisfy a number of constraints. There are a number of well-known problems in this area such as Multiple Knapsack Problem (MKP).

Digalakis and Margaritis [51] introduced a MP-CA for the first time. Though the first MP-CA in OP was introduced in 2002, recently there has been more interest in applying MP-CA to optimization problems. The majority of the published papers in this area were in the last two years, which means that this research area attracts more interest recently and it is becoming more mature.

4.2.1 Master-Slave Approach

The electrical generator maintenance scheduling problem which involves a combinatorial optimization is still an open problem. There are a number of methods offering the optimal solution for small-size problems, but there is no method that guarantees finding the optimal solution for all problems. Digalakis and Margaritis [51] proposed a MP-CA to solve electrical generator scheduling.

Digalakis and Margaritis [51] refer to different types of algorithms that have been proposed to deal with resource constrained project scheduling problem, which includes traditional heuristics such as dynamic programming [226, 212], modern heuristics such as Simulated Annealing (SA) [107, 177], and population based search such as GA [67],

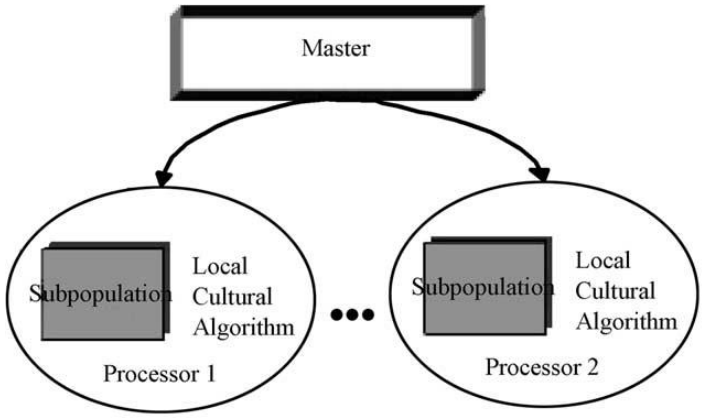


Figure 4.2: MP-CA Architecture of [51]

Hybrid GAs [26, 27] and parallel CAs [131]. However, the authors did not mention any shortcoming of existing methods in the literature.

Digalakis and Margaritis [51] propose a Parallel Co-operating Cultural Algorithm (PARCA) to solve the electrical generator scheduling problem. The proposed method is a master-slave approach, in which the master processor generates the initial population and manages it, while the slave processors execute different CAs on different sub-populations. The architecture of their proposed method is illustrated in Figure. 4.2. Their CAs are GA-based methods incorporating local search heuristics. Three different local search approaches are used in their proposed algorithm including Guided Local Search (GLS), Simulated Annealing (SA) and Tabu Search (TS). For GLS local optimizer, the authors refer to the job done by Moscato and Holstein [140]. Each CA uses one of the mentioned local search heuristics, arbitrarily.

Each sub-population uses different operators and parameter values. For example, some sub-populations use adaptive crossover probability whereas others do not. In their design, sub-populations use different data sets and communicate with each other using Message-Passing Interface (MPI) [46] to migrate good individuals. When there is no improvement for a CA after a given number of generations, it accepts immigrant from a different sub-population. They used three as the number of generations without improvement and 10% for the migration rate.

Digalakis and Margaritis used three problems to evaluate their proposed method. These problems which are described in detail in [180] have been used by other researchers [27] for their comparison. Four different versions of PARCA were tested to find out the best combination. These versions include a combination of PARCA with only TS, its combination with SA, its combination with GLS and its combination with all the three local optimizers. They also investigated the effect of number of processors in the performance of the proposed method by experimenting different number of processors from one to six.

By comparing different types of proposed algorithm, the authors claim that the algorithm which uses all three local search heuristics outperforms the others which use only one type of local search approaches. They also claim that using more distributed processors decreases the number of generations needed to find the optimal solution, which is because of decreasing the probability of facing local optima.

Digalakis and Margaritis [51] state that their proposed method produces better results, while its execution time is the same or slightly more. It should be mentioned that they do not mention any comparison between their work to others'.

4.2.2 Knowledge Migration Strategy

In this section, there is no specific problem to be solved, but overall optimization problems are referred as the main areas where CAs have been successfully applied. However, Guo *et al.* [74] proposed a new idea which is exchanging knowledge among sub-populations instead of migrating individuals.

In the case of single-population CAs, Guo *et al.* [74] refer to the methods which combine CAs with different optimization methods such as work done by Becerra and Coello [16] and Coelho and Mariani [34]. As the existing CAs with multi-population model, the authors refer to the algorithm proposed by Digalakis and Margaritis [51] to deal with electrical generator scheduling optimization (discussed in subsection 4.2.1),

and the proposed method by Alami *et al.* [8] which uses fuzzy clustering to divide population into sub-populations (discussed in subsection 4.4.1).

The CA proposed by Digalakis and Margaritis [51] is a MP-CA in which individuals are migrated between sub-populations as the cultural exchange. [74] state that exchanging individuals instead of implicit knowledge limits the performance of the evolutionary process. They also mention that there are no details about the cultural exchange for another multi-population CA [8].

Guo *et al.* [74] propose a knowledge-migration-based MP-CA for the first time. Their proposed method migrates knowledge instead of the exact individuals. Since the knowledge has more information about the previous generations and the direction of evolution, it would be more effective to be migrated between sub-populations. In their model, population is divided into a number of sub-populations evenly, and then a CA is applied to each of them. The cultural exchange occurs every predefined number of generations.

Each local CA uses the best individuals by the acceptance function to update its own belief sub-space which stores normative knowledge as well as topographic knowledge. Normative knowledge keeps a record of the feasible search space, while topographic knowledge determines the distribution of current best individuals in the feasible search space. It divides the search space into some cells and assigns attributes to the cells determining the possibility of existing the optimal solution in the corresponding cells.

The knowledge in the belief space is used by an influence function to induce the mutation operator for generating offspring. An influence proportion function is proposed to determine how many individuals should be influenced by each of type of knowledge. Moreover, a strategy is proposed to merge the migrated knowledge with the private knowledge for each sub-population. Only the topographic knowledge is considered in this strategy.

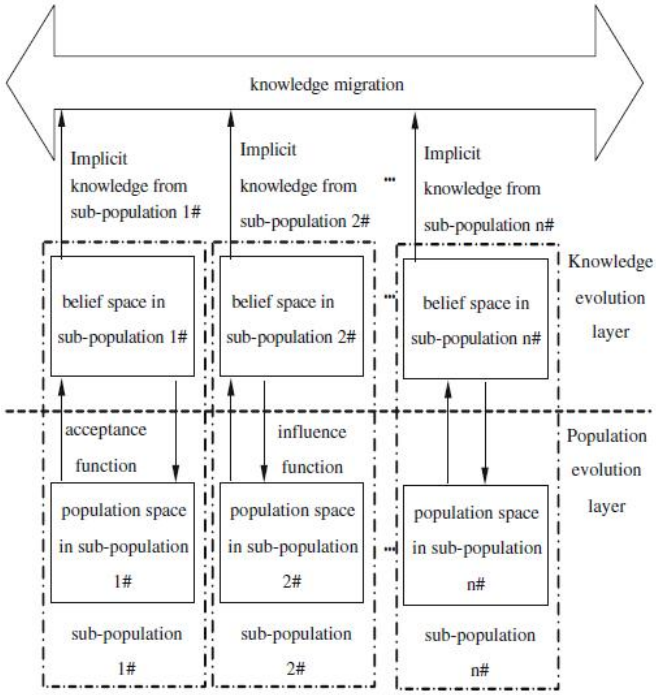


Figure 4.3: MP-CA Architecture of [76]

Guo *et al.* [74] used three benchmark functions to evaluate their proposed method. They present a table including the results of application of a traditional CA, the method introduced by Digalakis and Margaritis [51] and their proposed method. But there is no information saying what the numbers in that table are.

Guo *et al.* [74] claim that their proposed method has the best performance compared with the traditional CA and the method proposed by Digalakis and Margaritis [51] for both efficiency and effectiveness. They state that knowledge migration makes the model more effective, since it has useful information about the evolution process.

An extended version of Guo *et al.* [74] is published by Guo *et al.* [76]. In the new version, they provide a figure to show their proposed architecture which is represented in Figure 4.3.

In addition to Guo *et al.* [74], the authors mention that there are five types of knowledge used by different researchers in implementation of CAs including situational knowledge, normative knowledge, topographic knowledge, domain knowledge

and history knowledge [19]. But only both normative knowledge and topographic knowledge have been considered for the proposed method.

It should be noted that for generating offspring, parent selection and knowledge selection are done by a roulette selection approach [204]. Moreover, the communication topology between sub-populations is considered as ring or mesh [111].

Guo *et al.* [76] carried out experiments on three different high-dimension benchmark functions. They also defined three different performance measures to evaluate the performance of the proposed method which include average optimal solution, average convergence iteration and average iteration when optimal solution appears for the fist time. To show the effect of the cultural exchange frequency, the authors examined their proposed method for four different rates. The proposed method was tested over both communication topologies; ring and mesh.

Guo *et al.* [76] claim that based on the results of experiments on the cultural exchange frequency, the best rate is every three generations. They state that a lower number of generations increases the communication cost, and moreover, it makes the migrated knowledge to be more similar to the private one, and controversially, the larger number decreases the effect of migrated knowledge on the evolutionary process. The authors also claim that the mesh topology has better performance than the ring, while the ring structure has lower cost of communication.

Comparing with the traditional CA and another multi-population CA [51], they claim to have better performance in both solution quality and time complexity.

Guo *et al.* [76] state that by incorporating knowledge migration instead of individual migration in multi-population Cultural Algorithms the convergence speed is increased and better solutions are explored.

4.2.3 Multi-population Cultural Genetic Algorithm (MCGA)

The Multiple Knapsack Problem (MKP) is a well-known NP-complete optimization problem which is applicable in a wide range of research areas. da Silva and de Oliveira [44] proposed a MP-CA to deal with MKP.

da Silva and de Oliveira [44] refer to the multipopulation model described by Tomassini [197] which divides the population into sub-populations called island models. Overall, island models are isolated from each other, but they accept migrated individuals from other sub-populations. They also refer to the work done by Lin *et al.* [121] which investigate the effects of different model parameters such as size of sub-population and migration size and frequency. The authors also refer to the classification of paralleled EAs, mainly GAs [142], which defines four categories including global master-slave, island, cellular, and hierarchical parallel GAs.

da Silva and de Oliveira [44] did not mention any shortcomings of the existing methods in the literature. They just mentioned that they want to investigate the effect of the multi-population model and its parameters on the performance of CAs.

da Silva and de Oliveira [44] propose a MP-CA called Multi-population Cultural Genetic Algorithm (MCGA) to solve the MKP. The architecture of their proposed method is illustrated in Figure 4.4. They divide the population into sub-populations and apply a genetic-based CA to each sub-population. In their model, a communication link has been considered between the belief spaces of different CAs which accept migrations from other populations. CAs communicate with each other every predefined number of generations. In each communication, 20% of the best individuals are migrated to the belief space.

da Silva and de Oliveira [44] carried out a number of experiments using benchmark data sets. They claim that their proposed algorithm can find the best solution for the sample test problems they used. The authors state that combining CA with multi-population model increases search speed as well as convergence rate.

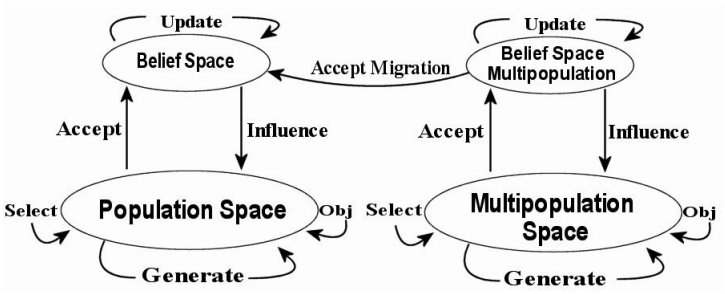


Figure 4.4: MP-CA Architecture of [44]

4.2.4 Competitive Co-evolutionary Cultural Differential Evolution (CCCDE)

The constraint optimization problem is a class of problems for which there is an optimization function required to satisfy a number of constraints. The butane alkylation process is a real-world constraint optimization problem where the optimization goal is to maximize the profit of the process. Xu *et al.* [210] propose a MP-CA to solve constraint optimization problem and they apply it to butane alkylation process as a real problem.

As the application of CAs in different fields, Xu *et al.* [210] refer to the methods proposed by Coello and Becerra (2002, 2003) [35, 36] and Yu and Gu [217]. The authors state that the traditional CAs have a high chance to be trapped into local optimal solutions, which makes it to be inaccurate and premature.

Xu *et al.* [210] propose a MP-CA which uses Differential Evolution (DE) as its evolution process. The proposed algorithm which is called Competitive Co-evolutionary Cultural Differential Evolution (CCCDE) divides the whole population into two sub-populations. To implement competitive co-evolution, CCCDE uses competitive fitness as the fitness function which is proposed by Peter and Jordan [156] and a competitive strategy between different populations proposed by Gu *et al.* [73] as a diversity measure. Considering only two sub-populations, the diversity measure is the difference between each sub-population's competitive fitness. Using the proposed formula

which contains three different threshold parameters, the premature convergence and lack of diversity can be detected and fixed by re-initializing the inactive individuals, which improves algorithm exploration.

Belief space records two types of knowledge including situational knowledge and normative knowledge. In the situational knowledge, the best found solution and the center individual are recorded. The center individual, which is the arithmetic average of all individuals, moves population to the promising region. The authors state that the idea of the center individual is inspired by the center particle introduced by Liu *et al.* [123]. The belief space which is updated by the top 20% of individuals, influence the mutation operation by both situational and normative knowledge, separately. So, two mutated individuals are generated for each individual.

Xu *et al.* [210] carried out experiment using four constrained optimization problems used by Koziel and Michalewicz [110]. To compare the results, each experiment was done 30 times to a maximum of 1000 generations. The authors state that all of the three threshold parameters are adjusted through extensive experiments, and the effect of each is investigated. Moreover, as a real constrained optimization problem, the authors applied their proposed method to the butane alkylation process. The optimizing model for this process is introduced by Bracken and McCormick [21] including 7 variables and 11 constraints.

Xu *et al.* [210] state that regarding the results the proposed method offers better solutions than Homomorphous Mapping (HM) proposed by Koziel and Michalewicz [110], and Stochastic Ranking (SR) proposed by Runarsson and Yao [176]. The authors also compare the results of CCCDE on the alkylation process with α-based Branch and Bound (αBB) proposed by Adjiman *et al.* [3], Constrained Ant Colony System (CACS) introduced by He and Chen [88] with different δ values, and CA with Evolutionary Programming (CAEP) proposed by Huang *et al.* [94]. The authors claim that the comparison shows that the proposed method outperforms CACS for

two δ values of 0 and 5×10^{-6}, while αBB, CACS for $\delta = 5 \times 10^{-4}$, and CAEP outperforms CCCDE. However, the authors state that those algorithms violate constraints, while CCCDE does not.

Xu *et al.* [210] claim that CCCDE has better performance as well as better convergence efficiency, compared to HM and SR methods. Moreover, they also claim that while considering constraints very well, their proposed method offers competitive results.

4.2.5 Multi-population Cultural Differential Evolution (MCDE)

The ammonia synthesis production is another real-world constrained optimization problem where the optimization goal is to maximize the net value of ammonia. Xu *et al.* [211] propose a MP-CA to deal with optimization problem and as a real-world application they apply their method to ammonia synthesis production.

Xu *et al.* [211] refer to some existing Cultural Algorithms (CAs) which use different types of Evolutionary Algorithms (EAs) to evolve their population space, such as Genetic Algorithm (GA) used by Gao *et al.* [61], Particle Swarm Optimization (PSO) proposed by Lin *et al.* [118] and Differential Evolution (DE) used by Becerra and Coello [15]. They also state that Differential Evolution (DE) is a population-based search introduced by Storn and Price [189] which uses a differential formulation to generate the offspring population. The authors state that the traditional CA suffers from trapping in local optimal solutions, which makes it premature and inaccurate.

Xu *et al.* [211] propose a Multi population Cultural Differential Evolution (MCDE) which uses both situational knowledge and normative knowledge for its belief space. In the situational knowledge, the optimal solution and center individual are recorded in each iteration. The center individual is the mean vector of all

individuals for each generation. The influence function of their model generates two new individuals by mutation operation using both knowledge types.

Only two sub-populations are considered in the MCDE model. The cultural exchange between these two sub-population occurs every predefined number of generations, which is set to 10 in the experiments. In the exchange process, 10% of each sub-population is selected randomly, then the better individuals, based on their fitness values, replace the worse ones.

To guarantee population diversity, a procedure is proposed which evaluates culture fusion between sup-populations using two parameters. If evaluation determines that the diversity is fail, two sub-populations are merged to one, and one of the sub-populations which has the shorter radius is re-initialized.

To evaluate proposed method, Xu *et al.* [211] considered 11 constrained optimization problems which have been used by Koziel and Michalewicz [110]. They did the experiments 30 times for each problem with a whole population size of 100 and 1000 generations. As the fitness function, they used the penalty function introduced by Zangwill [219]. The authors mention that the algorithm parameters are tuned based on many experiments. Moreover, the authors applied their proposed method to a real chemical process which is called ammonia synthesis. A constrained optimization model was derived for this problem which is a function of five different parameters.

Xu *et al.* [211] compare their results for the 11 optimization problems with those of Homomorphous Mapping (HM) proposed by Koziel and Michalewicz [110] and Stochastic Ranking (SR) approach introduced by Runarsson and Yao [176], and they state that the proposed method offers better performance than the others by finding the best solution for seven problems and the near optimal solutions for others. Regarding experiments on ammonia synthesis, the authors state that since the optimal solutions are infeasible in the first 98 generations, their corresponding fitness value do

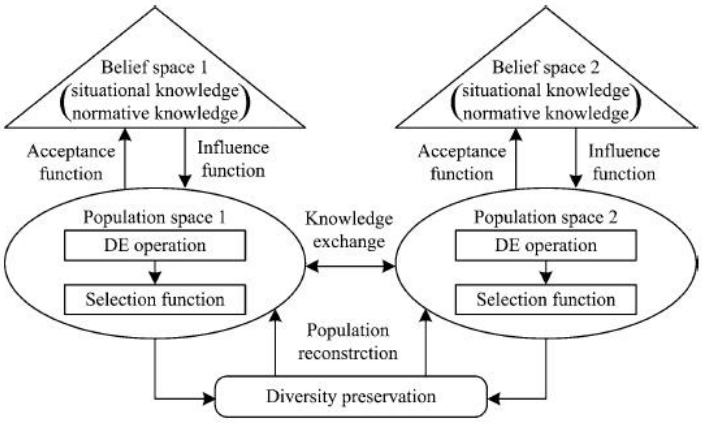


Figure 4.5: MP-CA Architecture of [209]

not make sense and after 98th generation, the solutions move to feasible area, such that the optimal solution is found.

Xu *et al.* [211] claim that their proposed algorithm offers better solution than some of the existing approaches, and moreover, they state that it is capable of solving other constrained optimization problems.

An extended version of Xu *et al.* [211] is published by Xu *et al.* [209]. In addition to the references of Xu *et al.* [211], the authors of the newer version referred to multi-population CAs proposed by Digalakis and Margaritis [51], Alami *et al.* [8], and Guo *et al.* [76].

In the newer version [209] a figure of the proposed method is provided which is represented in Figure 4.5. They also mentioned that when the influence function generates two new populations by mutation operation using each kind of knowledge, then offspring generation is provided using the a tournament strategy.

In addition to the experiments done by Xu *et al.* [211], Xu *et al.* [209] conducted experiments to compare their method with a GA proposed by Janikow and Michalewicz [100] and a PSO presented by Shi and Eberhart [185]. They also investigated finding the optimal values for diversity preservation thresholds.

In addition to the results presented in the former version [211], Xu *et al.* [209] claim that their proposed method uses a lower number of fitness evaluation comparing

to HM [110] and SR [176] which means that it has better computational efficiency. Compared to GA [100] and PSO [185], the authors claim to have better results with the same number of fitness evaluations.

Experimenting the proposed method with different values for the thresholds of diversity preservation, the authors state that the optimal values for three problems can be reached with different values of both thresholds. Moreover, they claim that the proposed method with different threshold values outperforms HM [110] and SR [176]. Through these experiments the authors determined a compromise setting for both thresholds.

Regarding experiments on ammonia synthesis, Xu *et al.* [209] compare the method with DEbest and DErand methods proposed by [190]. They state that the $DErand$ and $DEbest$ reach the optimum at 432^{nd} and 235^{th} generations, respectively, while the proposed method converges to the best solution at 119^{th} generation.

4.2.6 Multi-population Cooperative Particle Swarm Cultural Algorithm (MCPSCA)

Particle Swarm Cultural Algorithm (PSCA) is joint of PSO and CA which is used to deal with optimization problems. Guo and Liu [82] propose a PSCA incorporating multiple populations which is called Multi-Population Cooperative Particle Swarm Cultural Algorithm (MCPSCA).

Guo and Liu [82] refer to Coelho and Mariani [34], Sheng *et al.* [184] and Qiang *et al.* [163] as the three algorithms combined Particle Swarm Optimization with Cultural Algorithm which are called Particle Swarm Cultural Algorithm (PSCA). They also refer to another PSCA proposed by Ma and Ye [125] which uses multiple population in its population space. The PSCA introduced by Lin *et al.* [118] for parameter optimization of RBF fuzzy neural networks is also referred which is discussed in

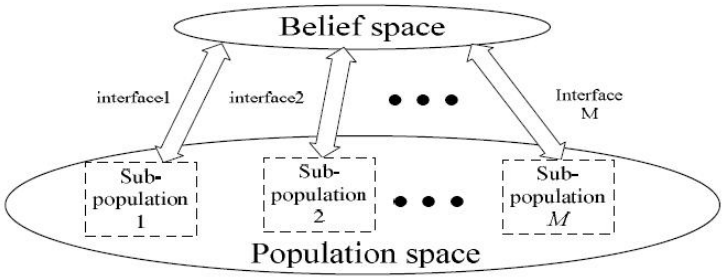


Figure 4.6: MP-CA Architecture of [82]

subsection 4.2.5. Lin *et al.* [118] define the m sub-population by dividing the m-dimension decision variable into m groups.

Guo and Liu [82] mention that Lin *et al.* [118] did not use the information migration among sub-populations on knowledge level, and moreover they optimized the parameters independently.

Guo and Liu [82] proposed a Multi-Population Cooperative Particle Swarm Cultural Algorithm (MCPSCA) which uses a co-evolution PSO in population space. The architecture of their proposed method is illustrated in Figure 4.6. The population is divided into several sub-populations and there is only one belief space shared with local PSOs which are applied on each sub-population independently. Initially the population is divided into the sub-populations with the same size, but the sub-populations size may be changed during the evolutionary process based on the density of each sub-population. The density increment is calculated for each sub-population, then for the sub-populations with positive density increment some new individuals are added and for others the individuals with the worst fitness value are eliminated.

Guo and Liu [82] refer to Guo *et al.* [76] (discussed in subsection 4.2.2) as a MP-CA which uses distinct knowledge migration among the sub-population to show that there is no direct knowledge migration in their proposed method. But the knowledge is exchanged indistinctly in the shared belief space.

The belief space stores two types of knowledge including situational knowledge and normative knowledge. The situational knowledge is incorporated to direct the local

search to the dominant particles and the normative knowledge is used to influence the initial positions of individuals. The belief space is updated by an acceptance function which gets a top best portion of individuals. The knowledge stored in belief space is used to influence the individuals according to the successful ratio of each type of knowledge.

Finally, the authors investigated the effect of knowledge migration intervals to be whether fixed or dynamic.

To evaluate the proposed method, Guo and Liu [82] applied it on 5 benchmark functions. They used three measurements for comparison including average optimal solution, average convergence generation, and convergence times. To investigate the effect of knowledge migration interval, four different experiments are done including three different fixed intervals with 1, 5, and 10 generations and one dynamic intervals. The author also did the experiments to compare their method with PSCA and multi-population PSCA.

Guo and Liu [82] state that the results for dynamic knowledge migration intervals are better than those three fixed intervals. They also claim that their proposed method has better convergence rate, better solutions and better success ratio comparing to PSCA and multi-population PSCA.

Guo and Liu [82] claim that the dynamic knowledge migration intervals outperforms the fixed intervals because it make the intervals shorter if the sub-population is trapped into the local optima. They also claim that the adjust-ability of the proportion of each sub-population to be influenced by each type of knowledge prevents the sub-populations to search unexplored regions and helps them to scape from tapping into local optima.

Table 4.1: Comparison of Proposed MP-CAs in Optimization Problems

ID	Method	Architecture	Knowledge Migration	Evolutionary Approach
1	Master-Slave Approach [51]	Centralized Local CAs	Individuals	GA with LS
2	Knowledge Migration Strategy [74, 76]	Peer Local CAs	Normative and topographic among Belief Spaces	EP
3	Multi-population Cultural Genetic Algorithm (MCGA) [44]	Peer Local CAs	Individuals among Belief Spaces	GA
4	Competitive Co-evolutionary Cultural Differential Evolution (CCCDE) [210]	Peer Local CAs	Situational and Normative	DE
5	Multi-population Cultural Differential Evolution (MCDE) [211, 209]	Peer Local CAs	Individuals among Population Spaces	DE
6	Multi-population Cooperative Particle Swarm Cultural Algorithm (MCPSCA) [82]	Shared Belief Space	Situational and Normative	PSO

4.2.7 Summary

As mentioned before, there are a number of proposed MP-CAs in the field of optimization problems. The proposed methods incorporate various architecture with different specification. They use different EAs including GA, EP and DE. Some methods migrate best individuals, while others use different types of knowledge such as situational, normative and topographic knowledge to be migrated. Table 4.1 gives comparison among proposed MP-CAs in this area.

As presented in Table 4.1, there are three different architectures proposed for MP-CA in the field of optimization problems. These architectures include centralized-local-CAs, peer-local-CAs, and shared-belief-space. In centralized-local-CAs architecture, the local CAs are managed by a server, while peer local CAs communicate and cooperate with each other without using a controller server. There are a number of

MP-CAs with peer-local-CAs architecture proposed in this area which incorporate different knowledge migration strategies. Some methods migrate individuals between population spaces while some others used knowledge migration between belief spaces.

In the third methodology, shared-belief-space architecture, local CAs do not have their own local belief space, and there is only one belief space which is shared among all the local CAs. In this architecture there is no need to a knowledge migration strategy.

4.3 Supply Chain Management

Supply chain management area is the second field in which MP-CA was appeared. By the knowledge of the authors, there is only one proposed MP-CA in this area which is published by Al-Mutawah *et al.* [4].

4.3.1 CA-based Distributed Multi-Objective Genetic Algorithm (CA-DMOGA)

The existing methods and principles for Supply Chain Management (SCM) assume a homogeneous cultural environment for the whole chain, while in reality the environment is dynamic and non-homogeneous. So, a SCM system should have a learning capability to adapt itself with environmental changes. Al-Mutawah *et al.* [4] propose a MP-CA to deal with the dynamic environment of SCM systems.

Al-Mutawah *et al.* [4] refer to GA-based methods that are proposed to deal with SCM systems such as the algorithms proposed by Joines *et al.* [101] and Truong and Azadivar [199]. As another multi-population GA, they refer to Distributed Multi-Objective Genetic Algorithm (DMOGA) proposed by Al-Mutawah *et al.* [5] which uses three different sub-populations including supplier, manufacturer, and retailer. Each sub-population uses a local GA to find its optimal attributes. DMOGA uses a

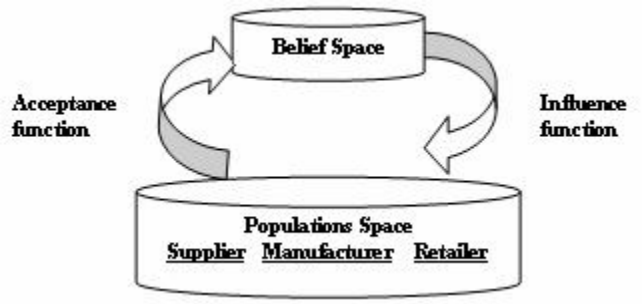


Figure 4.7: MP-CA Architecture of [4]

matchmaker model to generate the individuals containing three segments from different sub-populations. Finally, a global GA is used to generate a better configuration.

Al-Mutawah *et al.* [4] state that the main shortcoming of DMOGA and similar multi-objective GAs is the inheritance issue, such that new generations get experiences only from their parents and influences of other external sources are ignored.

Al-Mutawah *et al.* [4] propose a CA-based DMOGA to find the best sub-chain configuration for a SCM system. The proposed CA contains one belief space and one population space which is divided into three sub-populations including supplier, manufacturer and retailer. Figure 4.7 illustrates the architecture of their proposed method. A Decision Tree (DT) is also used for acceptance function which generates rules with confidence values. The confidence value is calculated using Dempter-Shafer (DS) theory [182].

Al-Mutawah *et al.* [4] carried out experiments on simulated data of a supply chain consisting of three sub-chains. To test the effect of cultural concepts, two test cases were considered one with a belief space and one without any cultural knowledge.

Al-Mutawah *et al.* [4] claim that the results of incorporating cultural concepts show an improvement in the convergence rate compared to the results of the method without using them. Moreover, they state that using global knowledge generates individuals with almost the same fitness value, while the GA by itself generates individuals with a clear variation in their fitness values. They state that cultural concepts have

a good influence on SCM systems, as the CA-based version of the DMOGA makes it faster.

Table 4.2: Comparison of Proposed MP-CAs in SCM

ID	Method	Architecture	Knowledge Migration	Evolutionary Approach
1	CA-based Distributed Multi-Objective Genetic Algorithm (CA-DMOGA) [4]	Shared Belief Space and Heterogeneous Sub-Populations	Rules	GA

4.3.2 Summary

Since there is only one published method in this area, there is no comparison. However, Table 4.2 represents characteristics of the proposed method [4].

However, the MP-CA architecture proposed by Al-Mutawah *et al.* [4] is different from others in case of their population space which consists of a number of heterogeneous sub-populations. They used a shared belief space to extract, store and update knowledge. In terms of knowledge type, they used a kind of rules which is defined by themselves.

4.4 Multimodal Optimization Problems

The field of multimodal optimization problems is another research area where MP-CAs have been introduced. Multimodal optimization problem is a subclass of optimization problems which needs to find a set of optimal or local optimal solutions instead of the only optimal one. Some multimodal optimization problems need to find all the optimal solutions, both local and global ones. There are a number of MP-CAs published in this area to find the optimal solutions of various multimodal optimization problems. The majority of these papers are published in 2007.

4.4.1 Fuzzy Clustering based Parallel Cultural Algorithm (FC-PACA)

The problem considered in this subsection is the multimodal optimization problem. While CAs perform well in optimization problems, they are not efficient for multimodal optimization where it is required to find multi global optima or maybe the local ones. Other methods such as niching approaches and immune system algorithms are proposed to deal with multimodal optimization. Alami *et al.* [6] propose a MP-CA incorporating fuzzy clustering to deal with multimodal optimization problems.

Many algorithms have been proposed to solve multimodal optimization problems. Among them, Alami *et al.* [6] refer to some niching techniques that are proposed in this field such as the GA proposed by Michalewicz [134] which uses niching to diversify its population, and the sharing method proposed by Goldberg and Richardson [69]. They also refer to other types of algorithms introduced to deal with multimodal optimization such as an Artificial Immune Algorithm (AIA) and a Particle Swarm Optimization (PSO) proposed by Castro and Timmis [48] and Li [115], respectively.

Alami *et al.* [6] state that the main shortcoming of the previous work is their dependency on some parameters that are required to be estimated based on each problem. As the authors mention, the parameter estimation influences the algorithm's performance. The performance of niching methods, for instance, depends on the niche radius and its spatial disposition [13, 179]. There is the same problem for the PSO proposed by Li [115] which needs an Euclidean radius parameter.

Alami *et al.* [6] propose Parallel Cultural Algorithms which uses fuzzy clustering to divide the population into clusters. Their proposed algorithm, which is called Fuzzy Clustering based Parallel Cultural Algorithm (FC-PACA) consists of three layers including a CAEP layer , a fuzzy clustering layer and a Spatial Separation (SS) layer. At the first cycle, FC-PACA generates the whole population, then it classifies some clusters of individuals, also called nations. SS is used to generate the

nations based on the results of fuzzy classifier, and separating the nations makes it possible to have parallel algorithms. Afterwards, a local CA is applied to each nation, so each nation has its own sub-population and belief sub-space. The nation's elite corresponds to the best individual in each nation. The belief space of each nation keep both situational knowledge and normative knowledge.

In the CAEP layer, the new population is generated from the parent population using only a mutation operator, such that each parent generates one child by random mutation. So, the belief space is used to influence the mutation operator.

Alami *et al.* [6] state that they select fuzzy clustering, since it can work as an unsupervised learning method to deal with the individuals without any prior knowledge about the data distribution. Moreover, it can handle overlapping clusters efficiently. The result of clustering is evaluated using a partition entropy parameter, such that it terminates the evolution cycles, when it reaches its corresponding threshold.

Parallel CAs communicate with each other to exchange their culture. Each nation communicates with its neighbors which are determined based on the distances between centers of nations. In the communication, nations exchange their normative knowledge. The exchange rate decreases over the iterations.

To evaluate the proposed algorithm, Alami *et al.* [6] used the multimodal functions incorporated by Beasley *et al.* [13] and Li [115] which are seven different functions with a different number of optima and different data distributions.

Alami *et al.* [6] claim that their proposed methods find all the optima and converge in each nation to its corresponding maximum. Comparing their method to the fitness sharing method proposed by Beasley *et al.* [13], the authors claim that they have identified the optima with more quality which is determined by the maximum peaks ratio criterion, while both methods locate all the optima. Moreover, the authors state that the proposed method is faster than the sharing method, which has been shown by comparing the number of fitness function evaluations.

They also compared their method to the AIA method proposed by Castro and Timmis [48] using two measurements defined by Castro and Timmis which are finding the global optimum and number of located optima. They state that both methods find the global maximum, but their proposed method finds 83 maxima by the 7th iteration while AIA method finds only 61 optima at the 451st iteration.

Moreover, they claim that the success rate of their method is better or at least the same as the approaches proposed by Beasley *et al.* [13] and Li [115] while their method uses a smaller number of fitness evaluations.

Alami *et al.* [6] state that incorporating fuzzy clustering improves the CA performance in the multimodal optimization problems. They claim to have better search performance both qualitatively and quantitatively, without increasing the time complexity. Since the main time consuming part of many optimization problems is their fitness evaluation, they state that the time complexity of other parts of their proposed method is relatively small. Finally, the authors mention that their proposed method is more efficient than the other existing methods in such problems that there is no prior knowledge about them.

An extended version of Alami *et al.* [6] is published by Alami *et al.* [8]. The newer method is the same as the former one with some slight changes. The exchange rate of nations' culture depends on the degree of their neighborhood, such that exchange probability for neighboring nations are more than that of remote nations. Moreover, in the extended version, the authors provide an image to show the architecture of their proposed method which is illustrated in Figure 4.8.

To evaluate the effectiveness and efficiency of the proposed method, Alami *et al.* [8] defined three parameter including:

1. Maximum Peak Ratio (MPR) which evaluates both quality and quantity of the results. It is calculated by dividing sum of the fitness of identified optimum by sum of the fitness of actual ones.

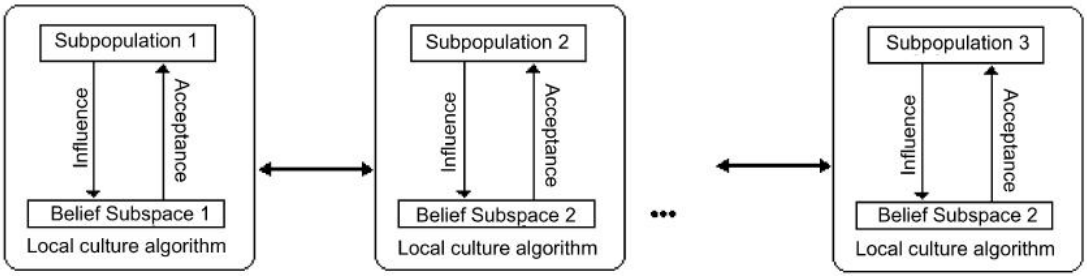


Figure 4.8: MP-CA Architecture of [8]

2. Effective number of Peaks Maintained (EPM).

3. Number of Fitness unction Evaluations (NFE).

As in Alami *et al.* [6], different multimodal functions have been used to evaluate the newer version. Alami *et al.* [8] used six different functions, four of which are the same as used in Alami *et al.* [6].

In addition to the claims of Alami *et al.* [6], Alami *et al.* [8] state that the inefficient results of the sharing method is caused by not estimating niche radius appropriately.

An extended version of Alami *et al.* [8] is also published by Alami and Imrani [7]. The method presented is the same as that mentioned in Alami *et al.* [6] and Alami *et al.* [8] with some slight changes. The important change is that they alter their proposed architecture a bit, which is presented in Figure 4.9. Moreover, Alami and Imrani [7] mention that they tested several validity criteria to validate the clustering method including a partition coefficient, the Xie and Beni index, the Fukuyama and Sugeno index, the separation / compactness index, the Bensaid index and the entropy criterion [152]. They mention that they chose the entropy criterion in their method because experiments show that it is more reliable than others. Unlike Alami *et al.* [8], the Maximum Peak Ratio (MPR) measure is not used to compare the methods.

Like the two earlier versions [6, 8], different multimodal functions were used to evaluate the proposed method. Alami and Imrani [7] used four different functions, two

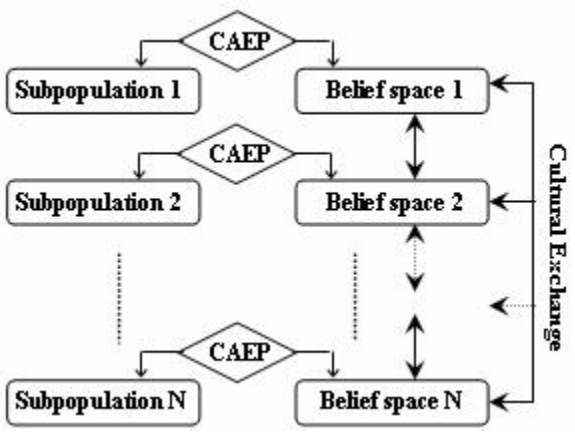


Figure 4.9: MP-CA Architecture of [7]

of which are similar to two functions they used in previous studies, one is exactly the same as one used before, and the last one is used only in Alami *et al.* [6]. In addition to these multimodal functions, the authors applied their proposed method on an electromagnetic benchmark for dielectric composite multimodal optimization problem to show the performance of their method on a real-world optimization problem. For this problem, any optimization technique can find the optimal solution, but the goal is to identify all the optima.

Alami and Imrani [7] claim the same results as Alami *et al.* [6, 8] for the multimodal functions. For the dielectric problem, the method converged after 4 generations using the population size 100.

In addition to the claims of Alami *et al.* [6, 8], Alami and Imrani [7] state that their method can be perfectly applied to real-world optimization problems since it does not require any prior knowledge of the input data.

4.4.2 Multi-population Multi-Objective Cultural Algorithm (MMOCA)

There are some multi-objective optimization problems which need to find a set of optimal solutions instead of only the optimal one. Guo *et al.* [75] propose a MP-CA to solve such problems.

Guo *et al.* [75] refer to different algorithms as the existing methods dealing with multi-objective problems including a strength pareto EA [224], a GA [49], a PSO [198], and a CA [36].

Guo *et al.* [75] mention that Zitzler *et al.* [224], Deb *et al.* [49], and Tripathi *et al.* [198] do not extract the evolution information completely, and the method proposed by Coello and Becerra [36] which is using the implicit knowledge is a single-population CA.

Guo *et al.* [75] propose a MP-CA to deal with multimodal optimization problem. First of all, a population is initialized with a number of individuals, then it is divided to different sub-population randomly. The local CAs are applied on each sub-population using EP to evolve population space and extract normative knowledge and topographic knowledge to update belief space. The knowledge migration is performed every predefined number of generation using the mesh topology.

Guo *et al.* [75] considered three different benchmark functions to evaluate their proposed method. The experiments were carried out 20 times using population size 250 on 4 sub-populations. Two measures are considered for comparison including SP-metric and GD-metric. SP-metric measures the average distribution of the non-dominated solutions, and the GD-metric measures the distance between non-dominated solutions and Pareto front.

Guo *et al.* [75] claim to have the better results for both measurement compared to the EA [224], GA [49], and PSO [198]. The authors state that knowledge migration

avoids blind selections and keeps the distribution and diversity of individuals which results in better solutions.

Table 4.3: Comparison of Proposed MP-CAs in Multimodal Optimization Problems

ID	Method	Architecture	Knowledge Migration	Evolutionary Approach
1	Fuzzy Clustering based Parallel Cultural Algorithm (FC-PACA) [6, 8, 7]	Peer Local CAs	Situational and Normative among Belief Spaces	EP
2	Multi-population Multi-Objective Cultural Algorithm (MMOCA) [75]	Peer Local CAs	Normative and Topographic	EP

4.4.3 Summary

A number of published MP-CAs were presented in this section which are proposed to deal with multimodal optimization problems. These methods are compared in Table 4.3 based on their architectures, knowledge migration strategies and incorporated EAs.

As it can be seen in the table, all the MP-CAs in this area incorporate peer-local-CAs architecture and also all of them use EP as their evolutionary approach, while they benefit from different types of knowledge.

4.5 Neurofuzzy Inference System Training

MP-CAs are introduced in Neurofuzzy Inference System (NFIS) as well. NFIS is a fuzzy system based on Neural Network approaches. The NFIS need a training procedure to optimize its parameters as well as to find the optimal solution. So the problem is which training approach should be used for the NFIS. There are a number of published MP-CAs in this research area, the majority of which were published in 2008.

4.5.1 Cultural Cooperative Particle Swarm Optimization (CCPSO)

Similar MP-CAs are proposed by Lin *et al.* [117], Chen *et al.* [29] and Lin *et al.* [118, 120] to train Neurofuzzy Inference Systems (NFISs). The most complete version of these methods is published by Lin *et al.* [117]. Lin *et al.* [118] considered prediction as the main problem for which some algorithms based on neural fuzzy networks have been proposed.

Neurofuzzy Inference Systems (NFISs) are fuzzy systems based on Neural Network approaches. NFISs contain a training procedure to optimized their parameters. Lin *et al.* [117] refer to Genetic Fuzzy Systems, as fuzzy systems which use EAs for their learning approach [39, 38, 89]. There are various strategies to incorporate EAs for NFISs, among which the authors refer to the three main strategies which are as follows:

1. Pittsburg-type: It uses fuzzy systems as individuals [188].
2. Michigan-type: It uses each fuzzy rule as an individual [155, 98].
3. Iterative rule learning: It searches for the best rule set in each generation [72, 37].

Lin *et al.* [117] also refer to an adaptive neuro-evolution method called symbiotic which is proposed by Moriarty and Miikkulainen [138]. In the symbiotic evolution, a GA is applied on a population of neurons. The individuals are the fuzzy rules which are combined together to form a fuzzy system. To decrease the effect of inappropriate fuzzy rule in a fuzzy system, a crossover operator is applied. In the symbiotic evolution, unlike other EAs, the best solution is found in different, unconverted populations. In other words, the symbiotic evolution does not converge to the optimal or near optimal solution [138, 102].

Lin *et al.* [117] refer to PSO developed by Kennedy and Eberhart [106, 52] as an EA which is used on different optimization problems [216, 1, 60]. They also refer to Cooperative Particle Swarm Optimization (CPSO) proposed by Bergh and Engelbrecht [17] which is a modification of the traditional PSO. CPSO uses p swarms of 1-dimensional vectors instead of one swarm of p-dimensional vectors. In CPSO, each swarm aims at optimizing one component of the vector.

Lin *et al.* [117] state that PSO does not work efficiently for multi-dimensional problems. The higher the dimensionality, the lower the PSO performance. They also state that the improved PSO, CPSO, still uses the same formula as the traditional one, so its particle diversity is still not sufficient which results in high chance to tarp into local optimal solution. In the other words, CPSO suffers from its inefficient exploration procedure [187, 128].

Lin *et al.* [117] propose an algorithm called Self-Evolving Evolutionary Learning Algorithm (SEELA) to deal with Neurofuzzy Inference Systems (NFIS). The architecture of their proposed method is illustrated in Figure 4.10. There are two types of learning in this algorithm, structure learning which sets the number of fuzzy rules and parameter learning which tunes the parameters for a NFIS.

The structure learning approach incorporates Subgroup Symbiotic Evolution (SSE) and Elite-based Structure Strategy (ESS) and uses the probability values to find the best number of fuzzy rules. Parameter learning is based on the combination of strategies of a CA and a CPSO. The combination strategy is called Cultural Cooperative Particle Swarm Optimization (CCPSO).

In the proposed method, a local CA is applied to each swarm such that each swarm plays as the population space and has a local belief space. The authors do not mention exactly what types of knowledge they are using, but they keep records of the local and global best position which can be considered as the situational knowledge and moreover they use an interval as the belief space which is the normative knowledge.

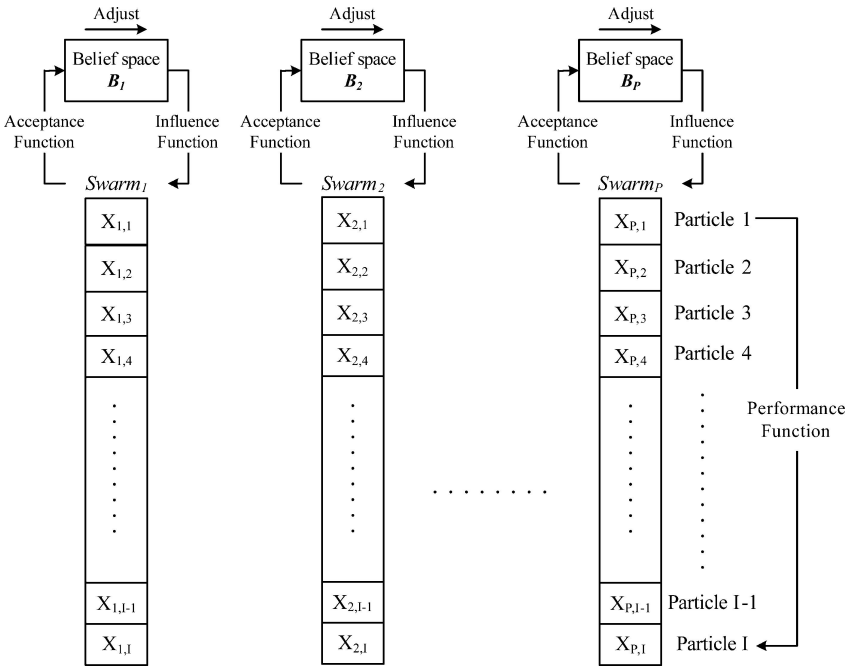


Figure 4.10: MP-CA Architecture of [117, 29, 118, 120]

To update the belief space, the best particles are selected by an acceptance function. The number of selected particles decreases during the evolution process.

The authors used normative knowledge to update the position of each particle, and used the situational knowledge to update the velocity to generate the new particle. There is no explicit knowledge migration between belief spaces. However, since each swarm is optimizing one variable and the final vector includes all the variables, the knowledge is exchanged implicitly.

Lin *et al.* [117] state that all parameters of the algorithm, excluding the acceptance function parameter are determined using extensive experiments. The parameter of acceptance function which is the number of the best particles to be selected to update the belief space is set to 20% which is suggested by Saleem [178].

Lin *et al.* [117] considered three examples to evaluate the proposed algorithm, including chaotic time series prediction [41], forecasting the number of sunspots [122], and backing up the truck controller [143]. For the first example, three cross-validation sets is considered to train and test the algorithm. In the second example, the number

of sunspot forecasting [122], which is a difficult prediction of the number of sunspots in non-stationary and non-Gaussian cycles from 1700 to 2004, the data of the first 151 years is used for training, while the whole data is used to test the proposed model. The main goal of the last example is to control the truck to move backward to a desire destination.

The experiments were carried out for ten runs, and for 1000 generations for the first two examples and 500 generations for the truck example.

Regarding chaotic time series prediction, Lin *et al.* [117] claim that the results show that nine is the best number of rules for the first cross-validation set. The authors compared the performance of their proposed method with those of PSO [106], CPSO [17], and GA-based neural fuzzy system [116]. They state that for all three cross-validations the proposed method has the smallest RMS error. They also claim that comparing the generalization capability of the proposed method with other methods presented by Jang [99], Kasabov and Song [105], Juang *et al.* [102] and Cowder [41] shows that the proposed method has the smallest RMS error to predict 500 points immediately after training session.

Lin *et al.* [117] also state that as with the first example, the best number of rules for the second example, the number of sunspots forecasting, is nine too, and comparing with PSO [106], CPSO [17], and GA-based neural fuzzy system [116], the proposed model offers better prediction with less RMS error.

Lin *et al.* [117] also state that the best number of rules for the third example is again nine, like both the first two examples, and comparing with other existing methods including Lin and Lin [119], Juang *et al.* [102] and Nomura *et al.* [145], the proposed method outperforms other methods by offering smaller RMS error.

Lin *et al.* [117] claim that their proposed method is a novel method which uses an EA to determine the number of fuzzy rules and adjusts the model parameters.

They also state that their proposed model outperforms the traditional PSO as well as CPSO generally.

Lin *et al.* [120] also claim that using the belief space increases the global search capacity. They also state that the proposed method avoids trapping into local optimal solutions.

Table 4.4: Comparison of Proposed MP-CAs in NFIS Training

ID	Method	Architecture	Knowledge Migration	Evolutionary Approach
1	Cultural Cooperative Particle Swarm Optimization (CCPSO) [117, 29, 118, 120]	Parallel Local CAs (each working on one variable)	Situational and Normative (Implicit Knowledge Migration)	PSO

4.5.2 Summary

All of the proposed MP-CAs to be used in NFIS training are a hybrid of CA and CPSO. So it can be mentioned that the EA for all the methods is PSO. Moreover, all of them used parallel local CAs which are the same, but working on different variables. Each local CA is applied to a local swarm which is designed to optimize one variable. So, each belief space has information related to its corresponding variable. Therefore, there is no exact knowledge migration among belief spaces, but the knowledge is exchanged implicitly. Table 4.4 represents specification of proposed MP-CAs in NFIS training.

4.6 Interactive Applications

MP-CAs are also used in interactive applications where some optimization problems need to be solved interactively. Interactive optimization problems such as music compositions and production design are categorized in the area of optimization field.

Since there is no explicit fitness function for these problems, they have to be evaluated by a human.

There are a number of MP-CAs proposed in interactive applications. They were recently published in this field such that the earliest one was published in 2009.

4.6.1 Cooperative Interactive Cultural Algorithms (CICA)

In some optimization problems for which there is no explicit fitness function, each individual should be evaluated by a human. Since the evaluation is performed by a human, human fatigue can influence the evaluation performance. So it limits the population size as well as the number of generations. Therefore, finding the optimal solution using a small-size population in a few generations would be a difficult issue. Moreover, human cognition and preference is another issue which is included in the evaluation as well. A number of MP-CAs are proposed by Guo *et al.* [78], Guo *et al.* [80], and Guo *et al.* [83] to deal with interactive optimization problems.

Guo *et al.* [78] refer to two Interactive Genetic Algorithms (IGAs) proposed by Gong *et al.* [71] and Guo *et al.* [79] as the related work. Actually in this article, the authors referred to those IGAs as Interactive Cultural Algorithms (ICAs).

Guo *et al.* [78] mention that some of previous work cannot be applied to different computer nodes. In other words, it is not possible for different users to participate in the evolutionary process. Moreover, they state that most of the parallel ICAs migrate only the best individuals, while migrating the knowledge extracted from the evolutionary process may have much more useful information than just best individuals, such as human cognition and preference.

Guo *et al.* [78] propose a Cooperative Interactive Cultural Algorithm (CICA) which has two evolutionary layers including population evolution layer and knowledge evolution layer. The architecture of proposed CICA is presented in Figure 4.11. The population layer interacts with users in computer nodes separately, while in the

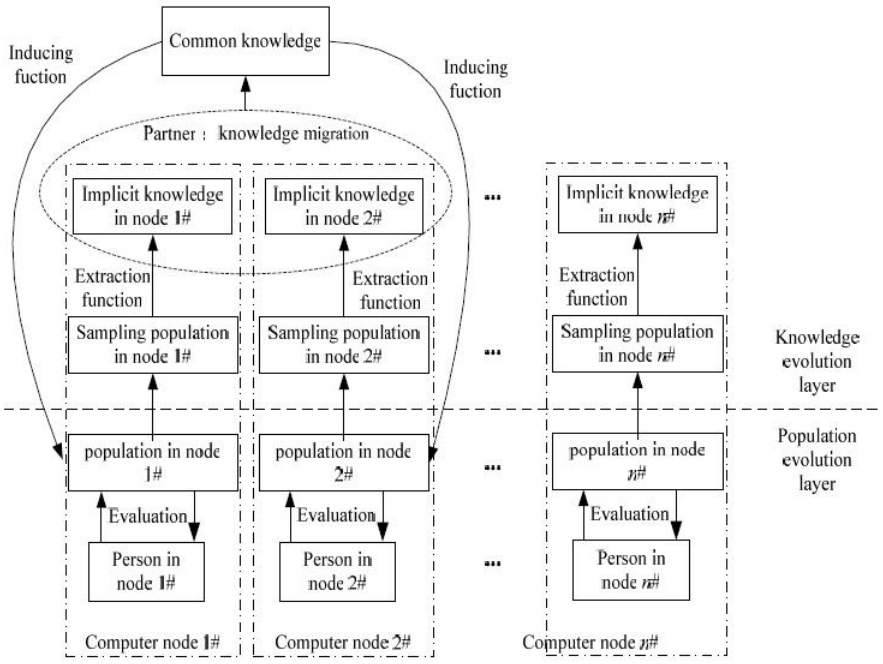


Figure 4.11: MP-CA Architecture of [78]

knowledge layer, the implicit knowledge is extracted from each node and migrated among others. The implicit knowledge which reflects the persons' cognition and preference is used to decrease human fatigue and improve convergence rate by sharing the knowledge and shrinking the search space.

The traditional IGA is used in the population layer and the implicit knowledge introduced by Guo *et al.* [77] is used in the knowledge layer. Since human fatigue depends on the number of individuals evaluated by the user and the similarity between those individuals, the authors defined the reliability of extracted knowledge as a function of population similarity and proportion of population evaluated by user. Moreover, they defined the evolution status as a measure to judge trapping in local optima or not. The evolution status is the difference between the average fitness of two consecutive generations. If the evolution status determines trapping in local optima, the knowledge migration is performed to scape from this situation.

If two users have similar preferences, their extracted human cognition knowledge from their computer nodes would be the same. So, exchanging the knowledge between

these two nodes is not useful. Therefore, the computer nodes with dissimilar implicit knowledge are the best candidates for knowledge migration. The authors categorize computer nodes in different knowledge tribes such that each knowledge tribe includes the computer nodes with dissimilar implicit knowledge. For knowledge migration, the computer nodes with the largest difference in the same tribe are selected. To update the private knowledge using migrated knowledge, a decision-making function is introduced which provides the common knowledge to be used to generate next population.

Guo *et al.* [78] considered cooperative fashion evolutionary design system to evaluate their proposed method. There are 16 persons divided into 4 groups using 4 computer nodes to interact with algorithm. The authors also compared two different migration strategies by experimenting the implicit knowledge migration as well as migrating the best individuals.

Guo *et al.* [78] claim that their proposed method converges faster than the IGA proposed by Guo *et al.* [79], and moreover its corresponding human fatigue is much less which is because of decreasing the number of individuals evaluated by users. The authors also claim that the knowledge migration shows better convergence rate as well as lower human fatigue than the individual migration strategy.

Guo *et al.* [78] state that migrating the knowledge of other users' cognition and preference improves the diversity of the population which will helps the population to escape from trapping into the local optima, and finally it improves the convergence rate and decrease the human fatigue. They also state that the implicit knowledge has much more useful information than the best individuals, so the propagation of the knowledge in the individual migrated strategy would be slower which results in lower convergence rate than using the knowledge migration strategy.

An extended version of Guo *et al.* [78] was published by Guo *et al.* [80]. Guo *et al.* [80] refer to an IGA proposed by Guo *et al.* [81]. As the shortcomings of existing

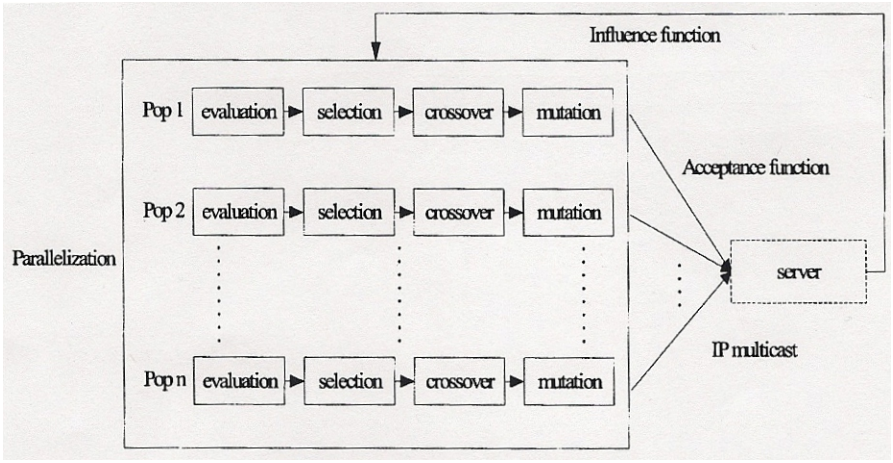


Figure 4.12: MP-CA Architecture of [80]

algorithms, Guo *et al.* [80] mention that since they are using the crossover operator on two individuals from different processors, the communication cost of the system is massive which makes the system inefficient.

The differences between Guo *et al.* [78] and Guo *et al.* [80] are as follows. Guo *et al.* [80] considered a server to keep and update the knowledge. Figure 4.12 illustrates the architecture of their proposed CICA. In each iteration, the acceptance function sends the best individuals of all of the populations to the server to update the knowledge. To find the similar users, the users' preferences are extracted using IP multicast transfer protocol instead of traditional TCP/IP protocol. Then the similar users join the same group automatically. Finally, the preferences from different users is used to influence the evolution.

Like Guo *et al.* [78], Guo *et al.* [80] considered a fashion evolutionary design system to evaluate their proposed method. In this system, a dress is expressed by four characteristics including coat pattern and color, dress pattern and color. There are 5 users to interact with the algorithm which is using the roulette wheel selection and elitism strategy on a population with 8 individuals. They also compared they method with the traditional IGA.

Guo *et al.* [80] claim that their proposed method converges fast while the users do not feel tired. Comparing to the traditional IGA, they claim that their method offers lower convergence time, lower evaluation by user, less user's fatigue, and better solutions. They state that their proposed method offers satisfied results by finding the optimal solution and converging fast.

Another MP-CA is published by Guo *et al.* [83] which is an extended version of Guo *et al.* [80]. Guo *et al.* [83] refer to an IGA published by Guo *et al.* [81] which uses frequent pattern mining to extract users' preference. They also refer to a distributed collaborative IGA published by Sun *et al.* [192], and an asynchronous collaborative IGA presented by Miki *et al.* [135]. The idea of using collaboration in an online shopping navigation system proposed by Hiroyasu and Yokouchi [90] is also referred by authors. They refer to the CICA proposed by Guo *et al.* [80] as a method which uses the extracted users' preferences to influence the evolution of other users with dissimilar preferences.

The authors state that the CICA proposed by Guo *et al.* [80] suffers from its network communication cost.

The differences between Guo *et al.* [80] and Guo *et al.* [83] are as follows. Guo *et al.* [83] used K-means clustering algorithm to classify the users into K different groups called knowledge alliances. Each alliance consists of users with the same preference. The classification is performed every generation, so the users may be switched between alliances. The common knowledge of each alliance is migrated among alliances which will be used by alliances to notice their own individuals using IP broadcast. In other words, the authors used a hierarchical exchange to decrease the communication cost of the network such that the alliance-oriented knowledge is migrated among alliances while the user-oriented knowledge is exchanged within the alliance. The authors state that for N users the communication complexity without hierarchical structure

is $N(N-1)$, while hierarchical structure makes it $N+K(K-1)$ for K knowledge alliance.

Since, in the interactive algorithms, the times spending by users are different, the steps of algorithm are asynchronous. The proposed method waits for %80 percent of the users to start knowledge migration. To fuse the migrated knowledge with self knowledge, the authors considered an add operation because both types of knowledge are in binary format.

Since human's cognitive is variable, the preference of users may be changed during the evolution. So it is possible for the users to be switched between alliances. Therefore, authors used K-mean clustering method which does not need any prior knowledge about data. In each classification, the users are classified into K knowledge alliances. The numbers of users in alliances are different and may be changed during the evolution.

Like Guo *et al.* [80], Guo *et al.* [83] used a fashion evolutionary design system. They also used roulette selection as well as an elitism strategy. In their experiments, they used $K=3$ and population size 8 evaluated by 10 users.

Guo *et al.* [83] claim that their proposed method outperforms the CICA and the IGA by offering lower number of user evaluation, smaller generations, more satisfying results as well as better convergence rate.

Guo *et al.* [83] state that knowledge migration among alliances speeds up the convergence and improves the population diversity.

4.6.2 Summary

In this section, the proposed MP-CAs in the field of interactive optimization problems were discussed. The specifications of these methods are presented in Table 4.5. As presented in this table, all the methods used implicit knowledge and incorporated IGA.

Table 4.5: Comparison of Proposed MP-CAs in Interactive Applications

ID	Method	Architecture	Knowledge Migration	Evolutionary Approach
1	Cooperative Interactive Cultural Algorithms (CICA) [78]	Peer Local CAs	Implicit Knowledge (Migrated within a Knowledge Tribe)	IGA
2	Cooperative Interactive Cultural Algorithms (CICA) [80, 83]	Centralized Local CAs with a shared Belief Space in a Server	Implicit Knowledge (Migrated among Knowledge Alliances	IGA

While the authors used the same name for the methods, their proposed architectures are different. The former method used peer-local-CAs architecture to extract implicit knowledge, while other methods incorporated a centralized architecture with a shared belief space. The former method extracts knowledge to construct knowledge tribes. Then within each knowledge tribe, the extracted knowledge is migrated among individuals. The recent methods use a server to extract, store and update the knowledge. The extracted knowledge is used by the server to construct knowledge alliances. Then the extracted knowledge is migrated among the individuals from different alliances.

4.7 Conclusions

In this survey, published CAs which use multiple populations have been discussed. A CA incorporating multi-population concept is called Multi-Population Cultural Algorithm (MP-CA) in this article. Based on extensive research and the knowledge of the authors in this area, all the published MP-CAs in the literature are surveyed in this article. Since these methods are published in various areas, we categorize them based on their problem domain into five classes including optimization problems,

supply chain management, multimodal optimization problems, neurofuzzy inference system training, and interactive applications.

The main category is optimization problems class which contains greater number of published methods. Digalakis and Margaritis [51] introduced a MP-CA for the first time. They used MP-CA to solve an optimization problem, namely electrical generator scheduling problem. Afterwards, Guo *et al.* [74, 76] introduced knowledge migration for the first time. They exchanged the extracted knowledge instead of individuals, since the knowledge has more useful information regarding previous generations and evolutionary direction. A MP-CA was also published by da Silva and de Oliveira [44] to solve another optimization problem called Multiple Knapsack Problem. Then other methods were introduced to solve real optimization problems namely butane alkylation process and ammonia synthesis process Xu *et al.* [210, 211, 209]. The application of a hybrid of PSO and CA using multiple populations in optimization problems was proposed by Guo and Liu [82].

MP-CA was also introduced in supply chain management systems by Al-Mutawah *et al.* [4]. They proposed a MP-CA to manage the complexity of dynamic environment which exists in the real supply chain systems.

The application of MP-CAs in multimodal optimization problems was introduced by Alami *et al.* [6]. They used fuzzy clustering in their method to generate a number of sub-populations. The extended versions of this work were published by Alami *et al.* [8] and Alami and Imrani [7]. The most recent method in this class was published by Guo *et al.* [75] to solve multi-objective problems.

A number of MP-CAs are also used a training approach for neurofuzzy inference systems [117, 29, 118, 120]. The most recent paper in this category was published by Lin *et al.* [118] to represent the prediction application of the training method.

The last category where MP-CAs are applied is interactive applications. The first one was published by Guo *et al.* [78] which introduced Knowledge Migration in this

category, followed by using user's preference aggregation published by Guo *et al.* [80]. Finally, Guo *et al.* [83] incorporated the new concept of dynamic knowledge alliance in MP-CA to use in interactive applications.

While the idea of all the proposed MP-CAs are similar, researchers determined various architectures for their proposed methods. Moreover, they used different characteristics for their methods such as different evolutionary approaches, knowledge types, exchange policies, and knowledge migration rates. It can be said that they tuned their methods based on their problem domains.

Table 4.6: Comparison of All Proposed MP-CAs based on their Architectures

Architecture	Methods
Centralized Local CAs	Master-Slave Approach [51] Cooperative Interactive Cultural Algorithms (CICA) [80, 83]
Shared Belief Space	CA-based Distributed Multi-Objective Genetic Algorithm (CA-DMOGA) [4] Cooperative Interactive Cultural Algorithms (CICA) [80, 83] Multi-population Cooperative Particle Swarm Cultural Algorithm (MCPSCA) [82]
Heterogeneous Sub-Populations	CA-based Distributed Multi-Objective Genetic Algorithm (CA-DMOGA) [4]
Peer Local CAs	Fuzzy Clustering based Parallel Cultural Algorithm (FC-PACA) [6, 8, 7] Cooperative Interactive Cultural Algorithms (CICA) [78] Knowledge Migration Strategy [74, 76] Multi-population Cultural Genetic Algorithm (MCGA) [44] Competitive Co-evolutionary Cultural Differential Evolution (CCCDE) [210] Multi-population Multi-Objective Cultural Algorithm (MMOCA) [75] Multi-population Cultural Differential Evolution (MCDE) [211, 209]
Parallel Local CAs	Cultural Cooperative Particle Swarm Optimization (CCPSO) [117, 29, 118, 120]

However, the proposed architectures could be classified in a number of groups which are as follows:

1. Centralized Local CAs (2002): There are a number of local CAs which cooperate to find the optimal solution such that all the local CAs are controlled by a server.

2. Shared Belief Space (2006): There is only one belief space for the whole system which is shared among the local CAs.

3. Heterogeneous Sub-Populations (2006): In this architecture, the whole population consists of a number of heterogeneous sub-populations.

4. Peer Local CAs (2007): There are a number of local CAs which work the same as each other to find the optimal solution.

5. Parallel Local CAs (2008): Like peer local CAs, there are a number of local CAs which cooperate to find the optimal solution, but in parallel architecture, local CAs do not work the same such that each local CA is designed to optimize one specific parameter.

The numbers in the brackets indicate the year when the architectures were introduced.

The main group of architectures is peer-local-CAs one which is used with different methodologies by a number of researchers in various areas [6, 8, 7, 74, 44, 210, 211, 75, 76, 209]. Table 4.6 represents the classification of all discussed methods based on their architectures.

As the future work, there are a number of jobs suggested by some researchers such as a theoretical research on the convergence behavior and a study on knowledge migration strategies. Besides, other researchers suggest do apply MP-CAs on greater number of real-world problems.

However, we do recommend a comparative study on different architectures which represents the strengthes and weaknesses of each architecture relative to others. It will clarify that whether or not it is possible to determine the best architecture for each problem domain.

Furthermore, we suggest to study the effects of different parameters on the performance of each method. Then it would be easy to tune a method just based on its architecture.

Acknowledgments

Special thanks to Dr. Richard Frost for his great help and useful comments for preparation of this survey.

Chapter 5

Multi-Population Cultural Algorithm

5.1 Introduction

As fully described in Chapter 4, Multi-Population Cultural Algorithm (MP-CA) is an improved version of the traditional CA. MP-CA was first introduced by Digalakis and Margaritis [51] to schedule electrical generators.

MP-CAs have more parameters to be optimized compared to the traditional CA such as the number of local CAs, their communication topology and frequency, and knowledge fusion. Therefore, many works have been conducted by various researchers to improve MP-CAs. One of the main improvement is to incorporate knowledge migration instead of individual migration between local CAs which is introduced by Guo *et al.* [74].

The successful applications of MP-CA have been recently reported in various areas such as constraint optimization problem [210], multimodal optimization problems [75], and interactive optimization problems [80, 83].

This chapter presents my proposed MP-CAs to deal with JSSPs which are published as research papers. These papers include:

- M. R. Raeesi N. and Z. Kobti, "A knowledge-migration-based multi-population cultural algorithm to solve job shop scheduling," in The 25th Florida Artificial Intelligence Research Society Conference (FLAIRS-25), Marco Island, FL, USA, May 23-25 2012, pp. 68-73 [168]:

  In this research paper, a MP-CA is proposed to deal with JSSPs. All of the proposed CAs in JSSP by the time use a single population in their population space. Therefore, the proposed method is the first attempt to use a CA with multiple populations to solve JSSPs. The proposed MP-CA incorporates a number of homogeneous local CAs communicating with each other by exchanging their extracted knowledge. In the proposed method, two different types of knowledge are considered to guide the search direction including normative and topographic knowledge. The details of the knowledge extraction and update, knowledge migration topology and frequency are presented in Section 5.2.

- M. R. Raeesi N. and Z. Kobti, "A multiagent system to solve JSSP using a multi-population cultural algorithm," in The 25th Canadian Conference on Artificial Intelligence (Canadian AI), no. 7310, Toronto, ON, Canada, May 28-30 2012, pp. 362-367 [170]:

  This research paper introduces a new knowledge for JSSPs which is called structured belief. Structured belief is incorporated within a MP-CA which uses knowledge migration as the communication strategy between its local CAs. Section 5.3 further describes the new knowledge and its incorporation mechanism.

5.2 A Knowledge-Migration-Based Multi-Population Cultural Algorithm to Solve Job Shop Scheduling

Abstract. In this article, a multipopulation Cultural Algorithm (MP-CA) is proposed to solve Job Shop Scheduling Problems (JSSP). The idea of using multiple populations in a Cultural Algorithm is implemented for the first time in JSSP. The proposed method divides the whole population into a number of sub-populations. On each sub-population, a local CA is applied which includes its own population space as well as belief space. The local CAs use Evolutionary Programming (EP) to evolve their populations, and moreover they incorporate a local search approach to speed up their convergence rates. The local CAs communicate with each other using knowledge migration which is a novel concept in CA. The proposed method extracts two types of knowledge including normative and topographic knowledge and uses the extracted knowledge to guide the evolutionary process to generate better solutions. The MP-CA is evaluated using a well-known benchmark. The results show that the MP-CA outperforms some of the existing methods by offering better solutions as well as better convergence rates, and produces competitive solutions when compared to the state-of-the-art methods used to deal with JSSPs.

5.2.1 Introduction

Job Shop Scheduling Problem (JSSP) is a combinatorial optimization problem which is well-known in different areas, specially manufacturing systems. JSSP is the task of scheduling different operations to be processed on different machines. The main

goal of this type of problems is minimizing the maximum completion time of all the operations. The maximum completion time of a schedule is also called *makespan*. JSSP is still an open problem. It is proved that the job shop scheduling systems with more than two machines are NP-complete [65], which means that there is no method capable to find the best solution for all the scheduling problems in an acceptable time.

There are various types of algorithms proposed to deal with JSSPs including heuristic approaches, meta-heuristic methods and Evolutionary Algorithms (EAs). In the area of Evolutionary Computation, there are various algorithms with different versions proposed to solve JSSPs including Genetic Algorithm (GA), hybrid GA, Ant Colony Optimization (ACO), Memetic Algorithm (MA), and Cultural Algorithm (CA). Each method has its own strengths and weaknesses. However, combinations of different types of algorithms work better.

In this article, a new CA is proposed to solve JSSPs. The proposed method incorporate a multipopulation design which is called multipopulation CA (MP-CA). In this design, there are a number of sub-populations incorporating local CAs to cooperate with each other to generate better solutions. The sub-populations communicate with each other by exchanging their extracted knowledge every predefined number of generations.

The structure of this article is as follows. Subsection 5.2.2 present the existing Evolutionary Algorithm introduced in the area of Job Shop Scheduling, which is followed by the definition of the classical JSSPs in Subsection 5.2.3. Subsection 5.2.4 describes the proposed MP-CA in details, and Subsection 5.2.5 shows the results of evaluating the MP-CA. Finally, the conclusions are represented in Subsection 5.2.6.

5.2.2 Related Work

The application of EAs in JSSPs is first introduced by Lawrence [112] as a GA. Hasan *et al.* [87] combined a GA with different priority rules including Partial Reordering,

Gap Reduction, and Restricted Swapping. An ACO method is proposed by Wang *et al.* [202] and recently we proposed a MA to solve JSSPs [164].

Becerra and Coello [14] introduced the application of CA in JSSP for the first time. CA, developed by Reynolds [175], is an EA which extracts knowledge to improve its search mechanism as well as its convergence rate. CA consists of population space and belief space. Population space contains individuals which are evolving to generate the optimal solution. The knowledge is extracted from the best individuals every generation. The extracted knowledge which is also called belief is recorded in the belief space to be used in the next generations to direct the evolutionary process. The link from population space to belief space which sends the best individuals in order to update the extracted knowledge is called acceptance function, and the link from belief space to population space which sends the updated knowledge to guide the evolution is called influence function.

The CA proposed by Becerra and Coello [14] incorporates Evolutionary Programming (EP) for population space evolution which only uses the mutation operator to generate the offspring population. The belief space of their proposed method only records the best individual as the situational knowledge. An improved version of the first CA [14] in JSSP is proposed by Cortes *et al.* [40]. The newer method is similar to the first version with the main differences being in the implementation of the mutation operator and influence function.

Ho and Tay [91] proposed a CA to solve Flexible Job Shop Scheduling which is called GENACE. Like Becerra and Coello [14], they used EP and situational knowledge. The next version of GENACE is also proposed by Ho and Tay [92] which is called LEGA.

All of the proposed CAs in JSSP use a single population in their population space. Our proposed method would be the first attempt to use a CA with multiple populations which is called MP-CA. However, there are a number of CAs in different fields

which are using multiple populations. Digalakis and Margaritis [51] introduced the multipopulation concept in CA for the first time by proposing a master-slave design. The master processor generates the initial population and manages it, while the slave processors execute different CAs on different sub-populations. The communication among sub-populations is implemented using Message-Passing Interface (MPI). The latter also used the situational knowledge.

One of the main issues in MP-CAs is the communication among the sub-populations. Most of the existing methods consider exchanging the best individuals as the communication process. But there is a more powerful strategy which exchanges the extracted knowledge instead of the best individuals. The new strategy is called knowledge migration which was first proposed by Guo *et al.* [74]. The extracted knowledge can incorporate more useful information about the previous generations to be used to direct the evolutionary process. Consequently, it would be more effective to use knowledge migration as the sub-populations communication mechanism.

5.2.3 Problem Definition

Classical JSSP is defined as a process of assigning different jobs to be processed on different machines [11]. There are M machines denoted by m_k and N jobs denoted by J_i, where k is the machine index and i is the job index. Each job is defined by a fixed sequence of operations. Each operation is denoted by O_{ij} where i is the job index and j is the operation index in that job. Each operation can be processed on only one machine in a known processing time. In other words, the route of operations of each job through the given machines is predefined.

In classical JSSP, there are some assumptions which are as follows: All the jobs are available at the starting point which are independent from each other; the machine set up time and part movement time between machines are negligible; each job is

Table 5.1: A sample classical job shop scheduling problem

Operation Index	1	2	3
J_1	$m_2, 1$	$m_1, 2$	$m_3, 3$
J_2	$m_1, 2$	$m_2, 1$	$m_3, 2$
J_3	$m_1, 2$	$m_3, 4$	$m_2, 1$

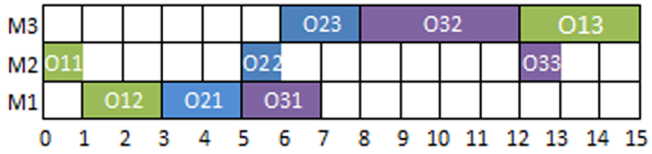


Figure 5.1: Sample Schedule

processed only one time on each machine; the machines can process only one operation at a time which cannot be interrupted; and there is no due date for the jobs.

Table 5.1 presents a sample classical JSSP which contains 3 jobs to be processed on 3 machines. This table shows the applicable machine with the corresponding processing time for each operation. The third operation of the second job O_{23} , for example, is applicable on the third machine, and it needs 2 time units to be processed. A sample schedule for this example is shown in Figure 5.1.

It should be mentioned that we use the active schedule concept defined by Croce *et al.* [43], and used by Hasan *et al.* [87] as Gap Reduction rule and by Becerra and Coello [14] as permissible left shift.

5.2.4 Proposed Cultural Algorithm

In this article, a MP-CA is proposed to deal with JSSPs. The proposed method is the first attempt to incorporate multiple populations in JSSP. In this method, the whole population is divided into a number of sub-populations. Each sub-population incorporates a local CA which contains its own belief space. The local CAs are cooperating to find the optimal solutions. They are communicating with each other by exchanging their own extracted knowledge which is called knowledge migration.

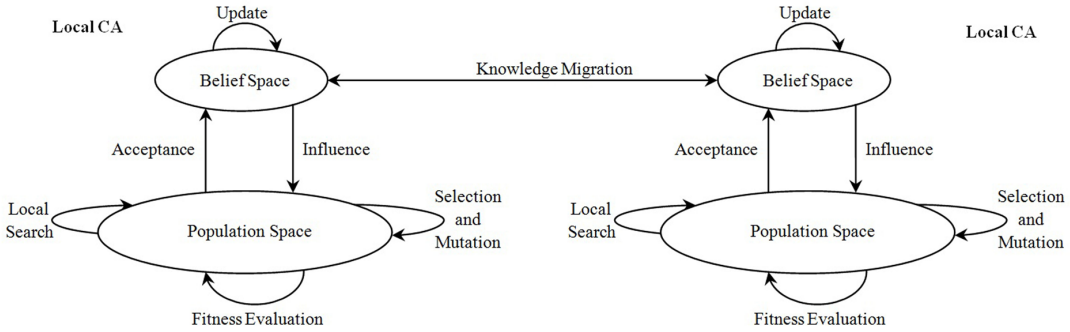


Figure 5.2: MP-CA Architecture

The architecture of the MP-CA is represented in Figure 5.2. As it is illustrated in the figure, like other CAs, each local CA has its own population space and belief space [175].

Moreover, the overall framework of the proposed MP-CA is represented in Figure 5.3. The number of sub-populations, the number of iterations which is the termination criterion, and the migration frequency are denoted by the $SubPopulationsNo$, $IterationNo$, and $ExchangeRate$ parameters, respectively. The parameters values which are used in our experiments are presented in Table 5.2.

5.2.4.1 Population Space

Population space is a set of individuals which are evolving using EP. Our EP uses the selection and mutation as regular EP, and incorporates a local search heuristic to speed up the convergence rate.

Chromosome Representation. Chromosome representation is one of the main characteristics of an EA. In the literature, there are various chromosome representations, with nine of them described by Cheng *et al.* [32]. Recently, we introduced Machine Operation Lists (MOL) representation [164]. MOL is an extended version of preference list-based representation such that it adds the concept of fixed list, the operation sequence of which cannot be changed unless due to the permissible left

PROCEDURE: MP-CA Framework
INPUT: Test Problems and Algorithm Parameters
OUTPUT: Optimal or Near-Optimal Schedules

Generate $SubPopulationsNo$ sub-populations.
FOR ($IterationNo$)
  FOR (*each subpopulation*)
    Evaluate all individuals and sort them.
    Apply mutation and local search method
      to generate offspring population.
    Update belief space.
  END
  IF ($IterationNo \bmod ExchangeRate = 0$)
    Exchange knowledge.
  END
END
Output the best found individual so far.

Figure 5.3: MP-CA Framework

shift. In [164], we showed that MOL representation outperforms preference list-based representation by yielding better solutions.

In our proposed algorithm, we incorporate the MOL representation. MOL considers a list of operations for each machine determining the sequence of operations to be processed on that machine. Because each machine only processes one operation of each job, the operations in the list can be denoted by their job indices. For example, the sample schedule illustrated in Figure 5.1 is represented as follows.

$$\{(1,2,3),(1,2,3),(2,3,1)\}$$

Evolutionary Programming. The proposed method uses EP to evolve the population space. The EP incorporates only the mutation operator as a genetic operator. The mutation operator uses the knowledge recorded in belief space to influence the direction of evolution. The EP applies the mutation operator on all the individuals to generate new ones. In our proposed MP-CA, there are two mutation operators which will be described in details in subsection 5.2.4.2.

Local Search. After generating the offspring population, a number of best individuals will be selected to be investigated by a local search heuristic. We use the same local search method as we used before [164] which is compatible with MOL representation. The search method reassigns all the operations of a randomly selected job to decrease the makespan. It has been shown by the authors that the time complexity of the local search method is negligible compared to the number of fitness evaluations in each generation.

5.2.4.2 Belief Space

Each local CA has its own belief space which space gets updated every generation using the acceptance function. The acceptance function passes a number of best individuals from the sub-population to the belief space. We consider the top 20 percent of the individuals in our implementation. The belief space extracts both normative and topographic knowledge from the individuals, and updates its own belief by the extracted knowledge. The knowledge stored in the belief space is incorporated to direct the mutation operator using the influence function.

The local CAs migrate their knowledge to each other to improve the search exploration in different sub-populations. The knowledge migration occurs every predefined number of generations. In knowledge migration, the sub-population which finds the best individual so far sends its own knowledge to others. The sub-populations which receive the migrated knowledge replace their own knowledge with migrated one.

Normative Knowledge. We use normative knowledge to improve the search exploration of our proposed method. Since normative knowledge records the feasible search space, it is used to explore all the feasible search space uniformly. In our method, we consider the position of each operation in its corresponding machine list as the search variable. So we have $M \times N$ variables for a system with M machines and N jobs. For each operation O_{ij}, there are a lower position $L_{O_{ij}}$ and an upper one $U_{O_{ij}}$. The

normative knowledge is initialized using the best individual as follows:

$$\begin{aligned} L_{O_{ij}} = &\quad P_{O_{ij}} - 0.5 \\ U_{O_{ij}} = &\quad P_{O_{ij}} + 0.5 \end{aligned}$$

where $P_{O_{ij}}$ denotes the position of operation O_{ij} in its corresponding machine list in the best individual at the first iteration. The 0.5 is used to provide a range of 1 for each position. The normative knowledge gets updated every generation by each individual passed by acceptance function using:

$$L_{O_{ij}} = min\left(L_{O_{ij}}, P_{O_{ij}} - 0.5\right)$$

$$U_{O_{ij}} = max\left(U_{O_{ij}}, P_{O_{ij}} + 0.5\right)$$

where $P_{O_{ij}}$ denotes the position of operation O_{ij} in its corresponding machine list in that individual.

Topographic Knowledge. We use the topographic knowledge to exploit certain regions which have more individuals. The topographic knowledge for each operation keeps the record of all positions used by all the individuals passed through the acceptance function. For each operation, there is a list of N available positions where N is the number of jobs. The topographic knowledge counts the number of occurrences for each position. Since the greater number of occurrences determines the existence of more good individuals in that region, the topographic knowledge is used to exploit those regions. Like the normative knowledge, topographic knowledge is also updated every generation, and it has its own mutation operator which influences the evolution to exploit certain regions.

Influence Function. Each type of knowledge has its own mutation operator, one mutation for normative knowledge and another one based on topographic knowledge. To generate a new individual, each machine list of an existing individual is mutated

using one of both mutation operators which is selected randomly with the same chance.

The normative-knowledge-based mutation operator calculates a position for each operation using the following formula, and then it sequences the operations in their operation lists based on their calculated position values.

$$P_{O_{ij}} = \begin{cases} P_{O_{ij}} + R \times \left(U_{O_{ij}} - L_{O_{ij}}\right) & P_{O_{ij}} < L_{O_{ij}}, \\ P_{O_{ij}} - R \times \left(U_{O_{ij}} - L_{O_{ij}}\right) & P_{O_{ij}} > U_{O_{ij}}, \\ P_{O_{ij}} + G & otherwise, \end{cases}$$

where R is a random number uniformly distributed between 0 and 1, and G is a random number in a normal distribution with mean 0 and variance 1.

In topographic knowledge, there is a list of position occurrences for each operation such that we need to use the position with greater occurrence to generate new individuals. Here we use roulette-wheel selection strategy to choose the new position for each operation.

5.2.5 Results

The proposed algorithm is implemented and evaluated using the java programming language version 1.6.0.18 on a system with Intel(R) Core(TM)2Quad 2.50GHz CPU and 8.00GB RAM. Table 5.2 presents the parameters used in our experiments which are adjusted using extensive experiments. In our experiments, we generate 7 sub-populations with 142 individuals, approximately 1000 individuals in total. We run the method for 200 iterations such that the knowledge migration occurs every 20 iterations.

In our experiments, we consider a well-known benchmark [113] for classical JSSP. The benchmark consists of 40 different problems with different size. All the experiments are done 10 times independently.

Table 5.2: Parameters of the proposed algorithm

Parameters	Value
$SubPopulationsNo$	7
$SubPopSize$	142
$IterationNo$	200
$ExchangeRate$	20

Table 5.3: Sample Results on LA Benchmark

Problem	Algorithm	Best	Median	Worst
la02	CA [40]	**655**	660.5	667
655	MP-CA	**655**	655.0	655
la20	CA [40]	907	912.6	924
902	MP-CA	**902**	907.0	907
la40	CA [40]	1256	1277.4	1328
1222	MP-CA	1228	1234.0	1245

The experiment results show that the proposed MP-CA finds the optimal solution for 28 test problems out of 40. To show the performance of the proposed method, we compare our method with another CA recently published by Cortes *et al.* [40]. The authors claim that their method finds the optimal solution for 26 test problems. However, our MP-CA outperforms their method by finding the best solution for 2 more test problems as well as by offering better statistical results (lower median and worst solutions) for those 26 test problems which means that the convergence rate of our proposed method is better than theirs. Moreover, for the remaining test problems, our method offers better solutions comparing to theirs, for all the best, median and worst solutions. Table 5.3 shows three sample results of our proposed MP-CA compared to the result of Cortes *et al.* [40][1]. The results show that incorporating multi-population and using knowledge exchange offers better solution and improves the convergence rate.

To have a fair comparison, we compare our results with their results for 200,000 fitness evaluations, not for more than 2,000,000 evaluations [40]. The maximum

[1]Please refer to Subsection 5.2.7 to see the complete results. These results are also accessible online at http://cs.uwindsor.ca/∼raeesim/Flairs-25/Statictical-Analysis.pdf.

Table 5.4: Comparison among Different Evolutionary Algorithms proposed recently to solve JSSP on LA Benchmark

Problem(Size)	Best Known	Hybrid GA [70]	MA [164]	CA [40]	Proposed MP-CA
LA20 (10×10)	**902**	907 (0.55%)	907 (0.55%)	907 (0.55%)	**902** (0.00%)
LA21 (15×10)	**1046**	1046* (0.00%)	1057 (1.05%)	1059 (1.24%)	1048 (0.19%)
LA22 (15×10)	**927**	935 (0.86%)	935 (0.86%)	947 (2.16%)	932* (0.54%)
LA24 (15×10)	**935**	953 (1.93%)	944 (0.96%)	950 (1.60%)	943* (0.86%)
LA25 (15×10)	**977**	986 (0.92%)	983* (0.61%)	998 (2.15%)	983* (0.61%)
LA26 (20×10)	**1218**	**1218** (0.00%)	**1218** (0.00%)	1219 (0.08%)	**1218** (0.00%)
LA27 (20×10)	**1235**	1256* (1.70%)	1269 (2.75%)	1279 (3.56%)	1264 (2.35%)
LA28 (20×10)	**1216**	1232 (1.32%)	1223 (0.58%)	1236 (1.64%)	1219* (0.25%)
LA29 (20×10)	**1157**	1196 (3.37%)	1191 (2.94%)	1219 (5.36%)	1182* (2.16%)
LA36 (15×15)	**1268**	1279 (0.87%)	1281 (1.03%)	1296 (2.21%)	1274* (0.47%)
LA37 (15×15)	**1397**	1408* (0.79%)	1429 (2.29%)	1416 (1.36%)	1415 (1.29%)
LA38 (15×15)	**1196**	1219 (1.92%)	1208 (1.00%)	1231 (2.93%)	1202* (0.50%)
LA39 (15×15)	**1233**	1246 (1.05%)	1248 (1.22%)	1269 (2.92%)	1240* (0.57%)
LA40 (15×15)	**1222**	1241 (1.55%)	1234 (0.98%)	1256 (2.78%)	1228* (0.49%)
Average ER		1.20%	1.20%	2.18%	0.73%
Average Ranking		2.39	2.46	3.79	1.36

number of evaluations in our method is 200,000 which is 1000 evaluations in 200 iterations.

Furthermore, different Evolutionary Algorithms are considered to be compared with our MP-CA including a hybrid GA [70], our published MA [164], and the CA [40]. All the four algorithms find the optimal solution for certain 26 test problems including la01-la19, la23, and la30-la35. Table 5.4 represents the results for the remaining 14 test problems. The Error Rate (ER) is considered to be able to compare the results over all the 14 problems which is defined as follows.

$$ER = \frac{C - LB}{LB} \times 100\%$$

where LB is the best-known solution, and C is the best solution found by the algorithms. The ER values are represented in brackets in Table 5.4.

To compare the 4 algorithms on the 14 problems, we used the non-parametric procedure incorporated by Garcia *et al.* [64] with the same levels of significance as theirs which are $\alpha = 0.05$ and $\alpha = 0.10$. To do so, first the algorithms are ranked using Friedman's test. The p-value for this test is less than 0.0001 which is also less than both significance levels; this means there are significant differences between the compared algorithms. Then, the algorithm with the minimum average ranking is selected as the control algorithm which is our MP-CA. Afterwards, the critical differences (CD) for control algorithm MP-CA are calculated using Bonferroni-Dunns test which are:

$$CD = 1.17 \quad \text{for} \qquad \alpha = 0.05$$
$$CD = 1.09 \quad \text{for} \qquad \alpha = 0.10$$

Now, the algorithms with the average rankings greater than the summation of the control algorithm ranking and CDs are considered as the algorithms with worse performance than the control algorithm. In other words, the summation of the average ranking of the MP-CA and the CD for each significance level is a threshold to find the worse algorithms. So the thresholds are as follows.

$$Threshold = 2.53 \quad \text{for} \qquad \alpha = 0.05$$
$$Threshold = 2.45 \quad \text{for} \qquad \alpha = 0.10$$

Figure 5.4 represents this concept graphically. The thresholds for $\alpha = 0.05$ and $\alpha = 0.10$ are illustrated using a solid line and a dashed line, respectively. This figure shows that the control algorithm MP-CA outperforms the algorithms whose bar exceeds the threshold line. So based on the Friedman's test and Bonferroni-Dunn's approach, we can claim that our proposed MP-CA outperforms the CA proposed by Cortes *et al.* [40] with significance level $\alpha = 0.05$ and outperforms our published MA [164] with level $\alpha = 0.10$. However, while the average ER and average ranking of the proposed MP-CA are better than the hybrid GA [70], these statistical tests show

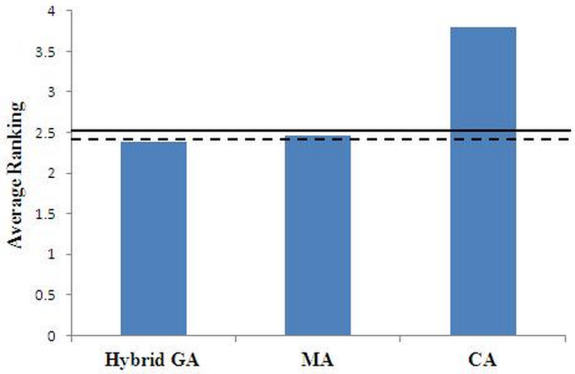


Figure 5.4: The Graphical Representation of Statistical Result of Friedman's test and Bonferroni-Dunn's method

Table 5.5: Overall Mutual Comparison

Proposed MP-CA	Win	Tie	Lose
HGA [70]	10	27	3
MA [164]	12	28	0
CA [40]	14	26	0

that the differences are not significant enough to say their performance is worse than our method's performance.

Finally, we compare our algorithm with others mutually over all the 40 problems. The comparison results illustrated in Table 5.5 shows that, for example, the MP-CA works similar to the hybrid GA [70] for 27 problems, outperforms it for 10 problems, and works worse for the remaining 3 problems.

5.2.6 Conclusions

In this article, we propose a MP-CA to solve JSSP which is a combinatorial optimization problem proved to be NP-Complete. This article introduces application of the MP-CA in JSSPs for the first time. Using multipopulation we improve the search exploration. The proposed method incorporates local CAs on the sub-populations cooperating to find the optimal solution. In addition, it uses knowledge migration

among sub-populations as the communication link. In the proposed method, we consider the extraction of normative and topographic knowledge.

The experiments show that the proposed MP-CA offers better solutions. Moreover, it improves the convergence rate. The comparison of the MP-CA with the traditional CA shows that using multiple populations as well as incorporating knowledge migration enhance the search space exploration and help the algorithm to avoid trapping into local optimal solutions.

The statistical comparisons reveal that the proposed method outperforms some existing methods very well and offers competitive solutions compared to other state-of-the-art methods. Furthermore, while the statistical comparisons do not show the significant differences between the proposed MP-CA and those methods, the results show that it outperforms others by offering lower error rates.

The proposed method is also applicable for other types of JSSPs, but since it uses the MOL representation and this representation does not cover other JSSP types, we need to use another representation or provide an extension for MOL, in future work.

5.2.7 Complete Results

Table 5.6 represents the complete results of our proposed MP-CA on a well-known benchmark introduced by Lawrence [113]. The tables also illustrates the results of another CA proposed by Cortes *et al.* [40].

Table 5.6: Results on the Lawrence [113] benchmark (la01-la40)

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
la01	10×5	**666**	CA [40]	**666**			666.5	668
			MP-CA	**666**	666.00	0.00	666.0	666

Table 5.6 – Continued on next page

Table 5.6: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
la02	10×5	**655**	CA [40]	**655**			660.5	667
			MP-CA	**655**	655.00	0.00	655.0	655
la03	10×5	**597**	CA [40]	**597**			610.2	623
			MP-CA	**597**	597.00	0.00	597.0	597
la04	10×5	**590**	CA [40]	**590**			593.0	599
			MP-CA	**590**	590.00	0.00	590.0	590
la05	10×5	**593**	CA [40]	**593**			593.5	595
			MP-CA	**593**	593.00	0.00	593.0	593
la06	15×5	**926**	CA [40]	**926**			926.0	926
			MP-CA	**926**	926.00	0.00	926.0	926
la07	15×5	**890**	CA [40]	**890**			890.0	890
			MP-CA	**890**	890.00	0.00	890.0	890
la08	15×5	**863**	CA [40]	**863**			863.4	865
			MP-CA	**863**	863.00	0.00	863.0	863
la09	15×5	**951**	CA [40]	**951**			951.3	953
			MP-CA	**951**	951.00	0.00	951.0	951
la10	15×5	**958**	CA [40]	**958**			958.1	959
			MP-CA	**958**	958.00	0.00	958.0	958
la11	20×5	**1222**	CA [40]	**1222**			1222.3	1224
			MP-CA	**1222**	1222.00	0.00	1222.0	1222
la12	20×5	**1039**	CA [40]	**1039**			1039.3	1041
			MP-CA	**1039**	1039.00	0.00	1039.0	1039
la13	20×5	**1150**	CA [40]	**1150**			1150.4	1152
			MP-CA	**1150**	1150.00	0.00	1150.0	1150

Table 5.6 – Continued on next page

Table 5.6: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
la14	20×5	**1292**	CA [40]	**1292**			1292.4	1294
			MP-CA	**1292**	1292.00	0.00	1292.0	1292
la15	20×5	**1207**	CA [40]	**1207**			1207.8	1209
			MP-CA	**1207**	1207.00	0.00	1207.0	1207
la16	10×10	**945**	CA [40]	**945**			962.8	990
			MP-CA	**945**	945.00	0.00	945.0	945
la17	10×10	**784**	CA [40]	**784**			793.4	811
			MP-CA	**784**	784.00	0.00	784.0	784
la18	10×10	**848**	CA [40]	**848**			857.5	863
			MP-CA	**848**	848.00	0.00	848.0	848
la19	10×10	**842**	CA [40]	**842**			859.0	872
			MP-CA	**842**	842.40	1.26	842.0	846
la20	10×10	**902**	CA [40]	907			912.6	924
			MP-CA	**902**	906.50	1.58	907.0	907
la21	15×10	**1046**	CA [40]	1059			1093.0	1114
			MP-CA	1048	1057.20	4.78	1059.0	1063
la22	15×10	**927**	CA [40]	947			964.3	985
			MP-CA	932	936.20	3.49	935.0	945
la23	15×10	**1032**	CA [40]	**1032**			1035.5	1045
			MP-CA	**1032**	1032.00	0.00	1032.0	1032
la24	15×10	**935**	CA [40]	950			976.6	997
			MP-CA	943	949.30	3.65	948.5	956
la25	15×10	**977**	CA [40]	998			1010.7	1034
			MP-CA	983	986.30	2.00	987.0	988

Table 5.6 – Continued on next page

Table 5.6: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
la26	20×10	**1218**	CA [40]	1219			1234.4	1260
			MP-CA	**1218**	1218.50	1.58	1218.0	1223
la27	20×10	**1235**	CA [40]	1279			1300.0	1324
			MP-CA	1264	1268.20	1.75	1269.0	1269
la28	20×10	**1216**	CA [40]	1236			1260.7	1291
			MP-CA	1219	1232.20	6.63	1234.0	1242
la29	20×10	**1157**	CA [40]	1219			1238.8	1271
			MP-CA	1182	1207.20	13.51	1204.5	1229
la30	20×10	**1355**	CA [40]	**1355**			1357.5	1369
			MP-CA	**1355**	1355.00	0.00	1355.0	1355
la31	30×10	**1784**	CA [40]	**1784**			1784.0	1784
			MP-CA	**1784**	1784.00	0.00	1784.0	1784
la32	30×10	**1850**	CA [40]	**1850**			1850.1	1851
			MP-CA	**1850**	1850.00	0.00	1850.0	1850
la33	30×10	**1719**	CA [40]	**1719**			1719.0	1719
			MP-CA	**1719**	1719.00	0.00	1719.0	1719
la34	30×10	**1721**	CA [40]	**1721**			1721.0	1721
			MP-CA	**1721**	1721.00	0.00	1721.0	1721
la35	30×10	**1888**	CA [40]	**1888**			1888.0	1888
			MP-CA	**1888**	1888.00	0.00	1888.0	1888
la36	15×15	**1268**	CA [40]	1296			1316.7	1351
			MP-CA	1274	1284.20	6.32	1282.0	1291
la37	15×15	**1397**	CA [40]	1416			1464.9	1514
			MP-CA	1415	1420.80	5.05	1419.5	1432

Table 5.6 – Continued on next page

Table 5.6: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
la38	15×15	**1196**	CA [40]	1231			1265.3	1276
			MP-CA	1202	1215.40	8.85	1216.0	1235
la39	15×15	**1233**	CA [40]	1269			1294.6	1327
			MP-CA	1240	1244.20	3.29	1243.5	1250
la40	15×15	**1222**	CA [40]	1256			1277.4	1328
			MP-CA	1228	1235.80	5.85	1234.0	1245

Acknowledgments

This work is made possible by a grant from the National Science Foundation and NSERC Discovery No. 327482.

5.3 A Multiagent System to Solve JSSP Using a Multi-Population Cultural Algorithm

Abstract. In this article, a multiagent system is proposed to solve Job Shop Scheduling Problems. In the proposed system, a number of autonomous agents cooperate in a Multi-Population Cultural Algorithm (MP-CA) framework. The proposed multiagent system consists of a number of groups of agents called sub-populations. The agents in each sub-population are co-evolving using a local CA. The local CAs are working in parallel and communicating to each other to exchange their extracted knowledge. The knowledge is migrated in the form of structured belief which is defined as a statistical records of an agent or a group of agents. Experiments show that our method outperforms some existing methods by offering better solutions as well as a better convergence rate.

5.3.1 Introduction

Job Shop Scheduling Problem (JSSP) is a well-known optimization problem in different areas, specially manufacturing systems. JSSP is the process of assigning various operations to different machines to be processed in predefined time while meeting some criteria. The optimization function of JSSP is to minimize the maximum completion time of all the jobs called *makespan*. There are lots of algorithms proposed to solve JSSPs which can be categorized into different types of algorithms such as heuristics, meta-heuristics and Evolutionary Algorithms (EAs). Moreover, there are combinations of different types which work better.

In this article, a multiagent system is designed and introduced to solve JSSP by incorporating a Multi-Population Cultural Algorithm (MP-CA) which is one of the first MP-CAs proposed in this area.

Operation Index	1	2	3
J_1	m_2,1	m_1,2	m_3,3
J_2	m_1,2	m_2,1	m_3,2
J_3	m_1,2	m_3,4	m_2,1

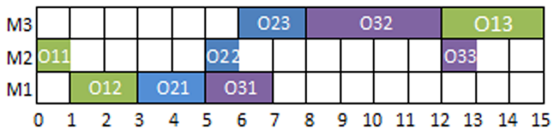


Figure 5.5: Sample Schedule

5.3.2 Related Work

There are various EAs proposed to solve JSSP. Lawrence [112] introduced an application of EAs in JSSP by proposing a Genetic Algorithm (GA). Cultural Algorithm (CA) is introduced in JSSP for the first time by Becerra and Coello Coello [14]. All of the existing CAs in JSSP use a single population. This article is one of the first attempts to use multi-population concept of CAs in JSSP. The idea of MP-CA is first introduced by Digalakis and Margaritis [51] to schedule electrical generators. Almost all the existing MP-CAs migrate the best individuals. Guo *et al.* [74] introduced a CA with knowledge migration for the first time. Since knowledge has more information about previous generations and evolution direction, it would be more effective to migrate knowledge among sub-populations.

Classical JSSP is a process of assigning different jobs to different machines. Each job is defined by a fixed sequence of operations. Each operation can be processed on only one machine in a known processing time. The machine set up time and part movement time between machines are negligible. Each job is processed only one time on each machine. The machines can process only one operation at a time which cannot be interrupted. Fig. 5.5 presents a sample classical JSSP containing 3 jobs to be processed on 3 machines.

5.3.3 Proposed Multi-Population Cultural Algorithm

The proposed multiagent system incorporates a Multi-Population Cultural Algorithm (MP-CA). Figure 5.6 illustrates the architecture of the proposed MP-CA. In this sys-

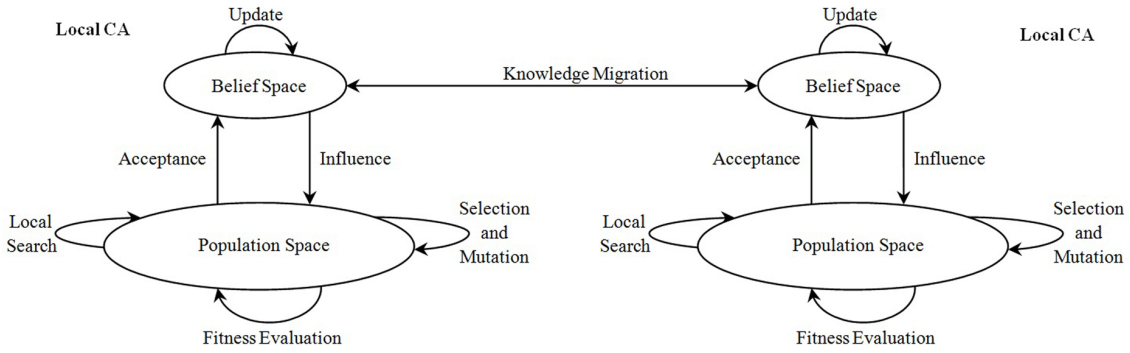


Figure 5.6: MP-CA Architecture which is an extended version of the traditional CA

tem, a number of autonomous agents are generated, which are randomly divided to some groups called sub-populations. Each agent has a chromosome which is an encoded schedule. The agents themselves use mutation operators to improve their chromosomes. There are two mutation operators, the random one and the knowledge-based one. A local CA is applied on each sub-population, such that each sub-population has its own belief space. The local CAs communicate with each other by exchanging their belief also called knowledge migration which is occurred every predefined number of generations.

Like other CAs, the proposed local CAs consist of both population space and belief space. Our population space includes a set of autonomous agents with their corresponding chromosomes which are represented using MOL representation [164]. The agents are autonomous and they mutate their own chromosomes. The agents are capable to incorporate a local search heuristic as well. We use the local search method presented in our previous publication [164].

The belief space is used to store the extracted knowledge which is represented as a structured belief. The structured belief is introduced for the first time in JSSP, which is an array of statistical data of operation positions in their corresponding machine's operation list. The length of the array equals to the total number of operations. The structured belief for one chromosome is an array of their operation positions. For

instance, the structured belief for $\{(1,2,3),(1,2,3),(2,3,1)\}$ as a sample chromosome is as follows:

O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
1	1	3	2	2	1	3	2	3

The belief of a sub-population is calculated by an average function of the belief of the chromosome of its best agents. Consider a sub-population with 100 agents for the sample problem such that the five best agents of which have the chromosomes as follows: $\{(1,2,3),(1,2,3),(1,2,3)\}$, $\{(1,3,2),(3,1,2),(2,1,3)\}$, $\{(2,1,3),(1,3,2),(1,2,3)\}$, $\{(3,2,1),(3,2,1),(2,1,3)\}$, and $\{(2,3,1),(2,3,1),(2,3,1)\}$. So the structured belief for this population of its top 5% agents would be the same as the following array.

O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
2	2	1.8	1.8	2.2	1.4	2.2	2.8	1.8

Each belief can be converted to an individual by sorting the operations in their corresponding machine's operation lists based on their position values represented in the belief. Using this definition, the resulted belief of the whole population would be converted to the chromosome $\{(2,1,3),(3,1,2),(2,1,3)\}$.

To find the new position for each operation the same routine is done. It finds out the position of the values in the structured belief regarding the values of other operations in the same operation list. For instance, if the position value for an operation is the lowest value, that operation should be reassigned to the first position of the list. Consider operation O_{21} in the belief mentioned above. Its value is 1.8 which is not the lowest value in this belief, but it is the lowest value compared to the operations having to be processed on machine m_1. So, its new position would be the first position on machine m_1.

Each local CA has its own belief space which gets updated every generation. The population space sends the top best agents to the belief space using an acceptance

function. The belief space generates a new belief using those agents. It updates its own belief as the average of its old belief and the new one. Then the belief space sends its updated belief to the population space by an influence function. Finally, the mutation operator uses the belief to generate offsprings.

In each generation the agents decide how to mutate their chromosomes. If there is no appropriate belief, the random mutation operator work as follows. First it finds the critical operations of the schedule and select two consecutive critical operations on the same machine randomly. Then, it swaps the positions of those operations. The second mutation operator uses the structured belief and works as follows. It finds all the critical operations and reassigns them to the positions whose information is embedded in the structured belief.

The overall framework of the proposed MP-CA is as follows. First, it generates $PopulationSize$ autonomous agents, and divides them into $SubPopulationsNo$ sub-populations. The number of iterations is predefined by the $IterationNo$ parameter. In the first iteration, each agent generates a chromosome randomly and afterwards, it uses the mutation and local search methods to improve its own chromosome. The fitness for each agent is the makespan of its chromosome, so the agents are sorted based on that ascendingly; the smaller, the better. After sorting the agents, the belief space is updated using the top best agents determined by the $TopBest$ parameter. Finally, the knowledge migration is occurred every predefined number of generations denoted by $ExchangeRate$.

5.3.4 Results

Parameters are adjusted using extensive experiments. We used the whole population of 1000 agents divided into 7 sub-populations. The top best agents includes the two best agents. The algorithm runs for 100 iterations and knowledge migration occurs every 20 generations. We used the Lawrence's benchmark [113] and carried out the

Table 5.7: Sample Results on LA Benchmark

Problem	Algorithm	Best	Average	Median	Worst
la24	CA[40]	950		976.6	997
15×10	SP-CA	948	958.40	956.5	970
935	MP-CA	941	951.90	952.0	962
la40	CA[40]	1256		1277.4	1328
15×15	SP-CA	1240	1251.40	1252.5	1262
1222	MP-CA	1234	1247.50	1249.5	1259

experiments for 10 independent runs for each problem. The MP-CA is also evaluated using only one sub-population which is called Single-Population CA (SP-CA). Some sample results [2] are represented in Table 5.7.

The results show that both SP-CA and MP-CA find the optimal solutions for 28 test problems out of 40, but in these problems the statistical results shows lower average, standard deviation and median values for MP-CA. It means that the MP-CA offers better convergence rate than SP-CA. Moreover for the rest of the test problems, the MP-CA outperforms SP-CA by offering better solutions. So, it is possible to say that MP-CA works better than SP-CA in all of the 40 test problems. In other words, incorporating multi-population and using knowledge exchange offers better solution and improves the convergence rate.

In order to show the performance of the proposed method, another CA recently published by Cortes *et al.* [40] is considered for the comparison. Their results are also shown in Table 5.7. Their method finds the optimal solution for 26 test problems, while our MP-CA finds the best solution for 2 more problems. For all other problems, the MP-CA offers better solution comparing to their CA. Furthermore, the proposed method is compared with different EAs proposed recently including a hybrid GA proposed by Goncalves et al. [70], our recently published Memetic Algorithm (MA) [164], and the CA proposed by Cortes *et al.* [40]. For certain 26 test problems of LA benchmark , all the five algorithms can find the optimal solution. The results of the

[2]Please refer to Subsection 5.3.6 to see the complete results. These results are also accessible online at http://cs.uwindsor.ca/∼raeesim/CanadianAI2012/Statictical.pdf.

Table 5.8: Comparison among Different EAs proposed recently to solve JSSP

Prob.	BK	hGA [70]	MA [164]	CA [40]	SP-CA	MP-CA
LA20	**902**	907 (0.55%)	907 (0.55%)	907 (0.55%)	**902** (0.00%)	**902** (0.00%)
LA21	**1046**	1046* (0.00%)	1057 (1.05%)	1059 (1.24%)	1059 (1.24%)	1057 (1.05%)
LA22	**927**	935 (0.86%)	935 (0.86%)	947 (2.16%)	935 (0.86%)	934* (0.76%)
LA24	**935**	953 (1.93%)	944 (0.96%)	950 (1.60%)	948 (1.39%)	941* (0.64%)
LA25	**977**	986 (0.92%)	983 (0.61%)	998 (2.15%)	989 (1.23%)	980* (0.31%)
LA26	**1218**	**1218** (0.00%)	**1218** (0.00%)	1219 (0.08%)	**1218** (0.00%)	**1218** (0.00%)
LA27	**1235**	1256* (1.70%)	1269 (2.75%)	1279 (3.56%)	1269 (2.75%)	1269 (2.75%)
LA28	**1216**	1232 (1.32%)	1223* (0.58%)	1236 (1.64%)	1234 (1.48%)	1225 (0.74%)
LA29	**1157**	1196 (3.37%)	1191* (2.94%)	1219 (5.36%)	1205 (4.15%)	1197 (3.46%)
LA36	**1268**	1279* (0.87%)	1281 (1.03%)	1296 (2.21%)	1294 (2.05%)	1281 (1.03%)
LA37	**1397**	1408* (0.79%)	1429 (2.29%)	1416 (1.36%)	1414 (1.22%)	1410 (0.93%)
LA38	**1196**	1219 (1.92%)	1208* (1.00%)	1231 (2.93%)	1215 (1.59%)	1208* (1.00%)
LA39	**1233**	1246 (1.05%)	1248 (1.22%)	1269 (2.92%)	1248 (1.22%)	1243* (0.81%)
LA40	**1222**	1241 (1.55%)	1234* (0.98%)	1256 (2.78%)	1240 (1.47%)	1234* (0.98%)
Avg. ER		1.20%	1.20%	2.18%	1.48%	1.03%
Avg. Ranking		2.61	2.50	4.75	3.29	1.86

algorithms as well as their error rates (ER) on the remaining 14 test problems are represented in Table 5.8.

To compare all the 5 algorithms, the non-parametric procedure incorporated by Garcia et al. [64] is used which includes Friedman's ranking and Bonferroni-Dunn's tests. We used the same levels of significance as they did which are $\alpha = 0.05$ and $\alpha = 0.10$. These tests inform us that the proposed MP-CA outperforms the CA [40] with significance level $\alpha = 0.05$ and outperforms the proposed SP-CA with $\alpha = 0.10$. While the average ER and average ranking of the proposed MP-CA is better than the hybrid GA [70] and the MA [164], but these statistical tests show that the differences are not significant enough.

5.3.5 Conclusions

In this article, a multiagent system is proposed in which agents use knowledge-based evolution to improve their fitness. The proposed method uses multi-population and knowledge migration to improve search space exploration as well as to prevent trap-

ping into local optima. The experiments show the proposed method outperforms the traditional CA by offering better solutions. Moreover, the comparison of MP-CA with another CA shows that the proposed method finds better solutions while it improves the convergence rate. Comparing to other types of algorithms, the proposed method offers competitive solutions. However, it is possible to say that the proposed MP-CA outperforms the existing methods, while it is not significant enough.

5.3.6 Complete Results

Table 5.9 represents the complete results of our proposed SP-CA and MP-CA on a well-known benchmark introduced by Lawrence [113]. The tables also illustrates the results of another CA proposed by Cortes *et al.* [40].

Table 5.9: Results on the Lawrence [113] benchmark (la01-la40)

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
la01	10×5	**666**	CA [40]	**666**			666.5	668
			SP-CA	**666**	666.00	0.00	666.0	666
			MP-CA	**666**	666.00	0.00	666.0	666
la02	10×5	**655**	CA [40]	**655**			660.5	667
			SP-CA	**655**	655.00	0.00	655.0	655
			MP-CA	**655**	655.00	0.00	655.0	655
la03	10×5	**597**	CA [40]	**597**			610.2	623
			SP-CA	**597**	597.60	1.90	597.0	603
			MP-CA	**597**	597.20	0.63	597.0	599
la04	10×5	**590**	CA [40]	**590**			593.0	599
			SP-CA	**590**	590.00	0.00	590.0	590
			MP-CA	**590**	590.00	0.00	590.0	590

Table 5.9 – Continued on next page

Table 5.9: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
			CA [40]	**593**			593.5	595
la05	10×5	**593**	SP-CA	**593**	593.00	0.00	593.0	593
			MP-CA	**593**	593.00	0.00	593.0	593
			CA [40]	**926**			926.0	926
la06	15×5	**926**	SP-CA	**926**	926.00	0.00	926.0	926
			MP-CA	**926**	926.00	0.00	926.0	926
			CA [40]	**890**			890.0	890
la07	15×5	**890**	SP-CA	**890**	890.00	0.00	890.0	890
			MP-CA	**890**	890.00	0.00	890.0	890
			CA [40]	**863**			863.4	865
la08	15×5	**863**	SP-CA	**863**	863.00	0.00	863.0	863
			MP-CA	**863**	863.00	0.00	863.0	863
			CA [40]	**951**			951.3	953
la09	15×5	**951**	SP-CA	**951**	951.00	0.00	951.0	951
			MP-CA	**951**	951.00	0.00	951.0	951
			CA [40]	**958**			958.1	959
la10	15×5	**958**	SP-CA	**958**	958.00	0.00	958.0	958
			MP-CA	**958**	958.00	0.00	958.0	958
			CA [40]	**1222**			1222.3	1224
la11	20×5	**1222**	SP-CA	**1222**	1222.00	0.00	1222.0	1222
			MP-CA	**1222**	1222.00	0.00	1222.0	1222
			CA [40]	**1039**			1039.3	1041
la12	20×5	**1039**	SP-CA	**1039**	1039.00	0.00	1039.0	1039
			MP-CA	**1039**	1039.00	0.00	1039.0	1039

Table 5.9 – Continued on next page

Table 5.9: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
			CA [40]	**1150**			1150.4	1152
la13	20×5	**1150**	SP-CA	**1150**	1150.00	0.00	1150.0	1150
			MP-CA	**1150**	1150.00	0.00	1150.0	1150
			CA [40]	**1292**			1292.4	1294
la14	20×5	**1292**	SP-CA	**1292**	1292.00	0.00	1292.0	1292
			MP-CA	**1292**	1292.00	0.00	1292.0	1292
			CA [40]	**1207**			1207.8	1209
la15	20×5	**1207**	SP-CA	**1207**	1207.00	0.00	1207.0	1207
			MP-CA	**1207**	1207.00	0.00	1207.0	1207
			CA [40]	**945**			962.8	990
la16	10×10	**945**	SP-CA	**945**	945.60	0.70	945.5	947
			MP-CA	**945**	945.20	0.42	945.0	946
			CA [40]	**784**			793.4	811
la17	10×10	**784**	SP-CA	**784**	784.90	1.20	784.5	787
			MP-CA	**784**	784.20	0.42	784.0	785
			CA [40]	**848**			857.5	863
la18	10×10	**848**	SP-CA	**848**	848.00	0.00	848.0	848
			MP-CA	**848**	848.00	0.00	848.0	848
			CA [40]	**842**			859.0	872
la19	10×10	**842**	SP-CA	**842**	846.60	2.84	847.0	850
			MP-CA	**842**	845.50	3.92	844.5	852
			CA [40]	907			912.6	924
la20	10×10	**902**	SP-CA	**902**	907.70	2.75	907.0	911
			MP-CA	**902**	904.00	2.58	902.0	907

Table 5.9 – Continued on next page

Table 5.9: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
			CA [40]	1059			1093.0	1114
la21	15×10	**1046**	SP-CA	1059	1067.00	6.20	1066.0	1078
			MP-CA	1057	1067.70	8.33	1065.0	1085
			CA [40]	947			964.3	985
la22	15×10	**927**	SP-CA	935	941.10	8.27	938.0	962
			MP-CA	934	942.30	6.15	941.0	954
			CA [40]	**1032**			1035.5	1045
la23	15×10	**1032**	SP-CA	**1032**	1032.00	0.00	1032.0	1032
			MP-CA	**1032**	1032.00	0.00	1032.0	1032
			CA [40]	950			976.6	997
la24	15×10	**935**	SP-CA	948	958.40	7.89	956.5	970
			MP-CA	941	951.90	5.76	952.0	962
			CA [40]	998			1010.7	1034
la25	15×10	**977**	SP-CA	989	995.10	5.26	997.5	1002
			MP-CA	980	987.00	5.06	988.5	994
			CA [40]	1219			1234.4	1260
la26	20×10	**1218**	SP-CA	**1218**	1226.50	9.13	1226.0	1244
			MP-CA	**1218**	1228.70	6.07	1229.0	1236
			CA [40]	1279			1300.0	1324
la27	20×10	**1235**	SP-CA	1269	1280.40	11.64	1278.5	1304
			MP-CA	1269	1280.30	7.62	1282.5	1293
			CA [40]	1236			1260.7	1291
la28	20×10	**1216**	SP-CA	1234	1254.70	12.49	1256.5	1273
			MP-CA	1225	1239.40	10.07	1241.0	1258

Table 5.9 – Continued on next page

Table 5.9: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
			CA [40]	1219			1238.8	1271
la29	20×10	**1157**	SP-CA	1205	1217.40	9.83	1215.5	1238
			MP-CA	1197	1231.00	14.04	1233.0	1249
			CA [40]	**1355**			1357.5	1369
la30	20×10	**1355**	SP-CA	**1355**	1355.00	0.00	1355.0	1355
			MP-CA	**1355**	1355.00	0.00	1355.0	1355
			CA [40]	**1784**			1784.0	1784
la31	30×10	**1784**	SP-CA	**1784**	1784.00	0.00	1784.0	1784
			MP-CA	**1784**	1784.00	0.00	1784.0	1784
			CA [40]	**1850**			1850.1	1851
la32	30×10	**1850**	SP-CA	**1850**	1850.00	0.00	1850.0	1850
			MP-CA	**1850**	1850.00	0.00	1850.0	1850
			CA [40]	**1719**			1719.0	1719
la33	30×10	**1719**	SP-CA	**1719**	1719.00	0.00	1719.0	1719
			MP-CA	**1719**	1719.00	0.00	1719.0	1719
			CA [40]	**1721**			1721.0	1721
la34	30×10	**1721**	SP-CA	**1721**	1721.00	0.00	1721.0	1721
			MP-CA	**1721**	1721.00	0.00	1721.0	1721
			CA [40]	**1888**			1888.0	1888
la35	30×10	**1888**	SP-CA	**1888**	1888.00	0.00	1888.0	1888
			MP-CA	**1888**	1888.00	0.00	1888.0	1888
			CA [40]	1296			1316.7	1351
la36	15×15	**1268**	SP-CA	1294	1300.70	5.60	1301.0	1312
			MP-CA	1281	1289.80	7.11	1291.0	1301

Table 5.9 – Continued on next page

Table 5.9: **(continued from previous page)**

Problem	Size	BK	Algorithm	Best	Average	SD	Median	Worst
			CA [40]	1416			1464.9	1514
la37	15×15	**1397**	SP-CA	1414	1435.60	13.03	1440.0	1449
			MP-CA	1410	1426.60	12.26	1425.0	1447
			CA [40]	1231			1265.3	1276
la38	15×15	**1196**	SP-CA	1215	1235.90	12.74	1235.5	1254
			MP-CA	1208	1226.10	10.47	1227.5	1240
			CA [40]	1269			1294.6	1327
la39	15×15	**1233**	SP-CA	1248	1256.40	9.48	1253.5	1275
			MP-CA	1243	1254.30	9.90	1251.0	1275
			CA [40]	1256			1277.4	1328
la40	15×15	**1222**	SP-CA	1240	1251.40	6.72	1252.5	1262
			MP-CA	1234	1247.50	8.11	1249.5	1259

Acknowledgments

This work is made possible by a grant from the National Science Foundation and NSERC Discovery No. 327482.

5.4 Conclusions

This chapter represents my proposed MP-CAs which are designed to solve classical JSSPs. It should be noted here that the MP-CA illustrated in Section 5.2 introduces the application of MP-CAs in JSSPs for the first time. Moreover, Section 5.2 represents a statistical analysis procedure to evaluate whether the performance improvement is significant or not.

Furthermore, the incorporated knowledge in these proposed MP-CAs are different such that the first MP-CA illustrated in Section 5.2 incorporates two types of knowledge including normative and topographic knowledge, while a new knowledge is introduced in the second MP-CA characterized in Section 5.3.

The proposed MP-CAs are evaluated over a well-known JSSP benchmark. The evaluation experiments reveal that the proposed methods outperform well to deal with classical JSSPs such that they offer competitive solutions compared to the state-of-the-art methods.

Since the proposed MP-CAs use the MOL representation and this representation does not cover other types of JSSPs, the proposed methods have only applied on the classical JSSPs. Therefore, incorporating another representation or providing an extension for MOL is considered as future directions for this research.

Chapter 6

Heterogeneous Multi-Population Cultural Algorithm

6.1 Introduction

Different architectures have been proposed to implement MP-CAs. Each architecture has its own strengthes and weaknesses. Homogeneous local CAs is one of the common architectures in which the local CAs have their own local belief spaces and works exactly the same [211, 76, 168].

Heterogeneous local CAs is another architecture for MP-CAs in which the local CAs are designed to work on different sub-problems instead of the whole problem. This architecture is introduced by Lin *et al.* [117, 118, 120] in their proposed Cultural Cooperative Particle Swarm Optimization (CCPSO) to train a Neurofuzzy Inference System. In CCPSO, each local CA is designed to work on only one dimension by incorporating Particle Swarm Optimization (PSO) to evolve its population. The local CAs in CCPSO have their own local belief spaces to record and update the extracted knowledge.

The most recent architecture to implement MP-CAs is my proposed Heterogenous Multi-Population Cultural Algorithm (HMP-CA) [171] which belongs to the category of heterogeneous local CAs architectures. HMP-CA and its improved versions are the key contribution of this dissertation which are published or accepted to be published as research papers.

This chapter represents the proposed HMP-CA and its improved versions in the following order:

- M. R. Raeesi N. and Z. Kobti, "Heterogeneous multi-population cultural algorithm," in IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, June 20-23 2013, pp. 292-299 [171]:

  This research paper introduces HMP-CA which includes a number of heterogeneous local CAs designed to optimize different subsets of the problem dimensions. HMP-CA incorporates only one belief space which is shared among local CAs instead of one local belief space for each local CA. The performance of the proposed HMP-CA is evaluated by its application on numerical optimization problems. Section 6.2 further characterized the proposed architecture and the results of its performance evaluation.

- M. R. Raeesi N., J. Chittle, and Z. Kobti, "A new dimension division scheme for heterogenous multi-population cultural algorithm," in The 27th Florida Artificial Intelligence Research Society Conference (FLAIRS-27), Pensacola Beach, FL, USA, May 21-23 2014 [165]:

  In order to incorporate HMP-CA, first a problem decomposition technique should be incorporated to divide the problems dimensions into a number of groups. This research paper investigates the effect of different static decomposition techniques on the performance of the proposed HMP-CA. The investigated static techniques include sequential, jumping, sequential with overlap,

logarithmic and customized logarithmic approaches. The details of the investigated techniques and their effects on the algorithm performance is presented in Section 6.3.

- M. R. Raeesi N. and Z. Kobti, "Heterogeneous multi-population cultural algorithm with a dynamic dimension decomposition strategy," in The 27th Canadian Conference on Artificial Intelligence (Canadian AI), Montreal, QC, Canada, May 6-9 2014 [174]:

  In this research paper, the proposed HMP-CA is improved such that it is able to deal with dynamic decomposition strategies. In addition to this improvement, two new dynamic decomposition techniques are introduced which are called top-down and bottom-up strategies. The idea of the former approach is to start with a local CA designed to optimize all the problem dimensions and split it if it cannot find a better solution for a number of generations. Conversely, the idea of the latter approach is to start with a number of local CAs designed to optimize only one dimension and merge them if they cannot find a better solution for a number of generations. The proposed dynamic approaches are fully described in Section 6.4.

- M. R. Raeesi N. and Z. Kobti, "Adaptive heterogenous multi-population cultural algorithm for large scale global optimization," in The 21st European Conference on Artificial Intelligence (ECAI), Praque, Czech Republic, August 18-22 2014 [173]:

  This research paper suggests to incorporate a more systematic composition instead of the random merging used in the bottom-up approach presented in the previous research article. In order to do so, a technique is introduced in this research paper to detect additively dimension interactions. The proposed technique which is called Variable Additively Interdependence Learning (VAIL) is

incorporated to design an adaptive dimension decomposition technique which merges the interacting dimensions into one local CA and preserves the non-interacting dimensions within different local CAs. The proposed VAIL technique and its effect on HMP-CA performance is presented in Section 6.5.

6.2 Heterogeneous Multi-Population Cultural Algorithm

Abstract. In this article, a new architecture for Cultural Algorithms is proposed. The new architecture incorporates a number of sub-populations such that each sub-population is designed to optimize different parameters. According to the assigned parameters, each sub-population is a set of partial solutions which are managed by a local CA. Local CAs do not communicate with each other directly. In this architecture, a shared belief space is considered to record the best parameters. Local CAs send their best partial solutions to the belief space every generation. The belief space then updates its record of best parameters which will be used later by local CAs to evaluate their partial solutions. Due to incorporating a number of heterogeneous sub-populations, the proposed architecture is called Heterogeneous Multi-Population Cultural Algorithm (HMP-CA). Additionally, a local search heuristic is proposed to speed up the convergence of HMP-CA. The proposed HMP-CA is evaluated using a number of numerical optimization benchmark functions. The results show that the HMP-CA without the local search offers competitive results compared to the state-of-the-art methods and incorporating the proposed local search heuristic makes the proposed HMP-CA more efficient such that it outperforms all the state-of-the-art methods.

6.2.1 Introduction

Optimization problems are a set of problems where the goal is to find a set of input parameters to make a system as effective as possible. The input parameters could be either discrete or continuous. A problem with continuous parameters is called a global

optimization problem, while one with discrete parameters is called a combinatorial optimization problem. The focus of this paper is on numerical optimization where the input is a D-dimensional vector of continuous parameters.

There are various types of algorithms proposed to solve optimization problems. Evolutionary Algorithms (EAs) is a subset of those methods successfully applied to deal with optimization problems. EAs are population-based approaches incorporating the concept of evolution inspired from what is seen in nature. An EA starts with a population of randomly generated individuals which is called an initial population. Incorporating evolutionary operators including mutation and crossover, it generates new population of individuals from the current population. Afterwards a selection method finalizes the population for the next generation. This method is inspired by the natural selection mechanism which states that the individuals which survive for the next generation are more likely the ones which show to be fitter to the environment. This routine stops when termination criteria (*e.g.* CPU time, predefined number of generations) are met.

Various types of EAs with different specifications are introduced in the literature. Genetic Algorithm (GA) and Genetic Programming (GP) are two popular EAs proposed by Holland [93] and Koza [109], respectively. Another instance of EA which is popular due to its efficient search space exploration is Differential Evolution (DE). DE is the most recent evolutionary method introduced in literature which was designed by Storn and Price [191] to solve global optimization problems. Cultural Algorithm (CA) is another EA developed by Reynolds [175] incorporating knowledge to improve the search mechanism. CA extracts the knowledge during the process of evolution and uses the extracted knowledge to direct the search process. The knowledge is stored and updated within another space different than the population space, called the belief space.

Although EAs are successfully applied on various types of optimization problems, they suffer from immature convergence. This is due to the fact that they cannot preserve the population diversity over generations. There are a number of strategies introduced to do so such as rejecting duplicate solutions and high mutation rate. Recently, researchers have been interested in incorporating multiple populations to keep the population diversity and scape the local optimal regions. The multiple populations approach divides the whole population into a number of sub-populations. Each sub-population then is evolved by a local EA. During the process of evolution, sub-populations communicate to each other. It is also possible to say that the local EAs are communicating. In fact parallel EAs could be considered as multi-population methods.

In this article, a new architecture is proposed for Multi-Population CA such that the sub-populations are not exactly the same. The optimization parameters are divided among sub-populations and each sub-population is a set of partial solutions which is responsible to optimize its own parameters. Each sub-population is evolved by a local CA which does not communicate to other local CAs directly. A local CA transforms its best partial solution to a shared belief space which will be access by other local CAs for evaluating their partial solutions. As a matter of fact, the local CAs are cooperating to find the best combination for the input parameters. Because the proposed architecture incorporates a number of sub-population with different set of parameters, it is called Heterogeneous Multi-Population Cultural Algorithm (HMP-CA). The proposed architecture incorporates DE as its evolutionary approach due to its ability to highly explore the search space, especially in global optimization problems.

The remainder of this article is organized as follows. Subsection 6.2.2 concisely describes DE and CA, and Subsection 6.2.3 mathematically introduces the definition of numerical optimization. The proposed HMP-CA is described in detail in Subsection

6.2.4, followed by representing the experiments designed to evaluate the proposed method, the discussion on the results and the comparison of the results with the state-of-the-art methods in Subsection 6.2.5. Finally, Subsection 6.2.6 represents conclusion remarks. It should be noted that the optimization functions incorporated for experiments are illustrated in Appendix 6.2.7.

6.2.2 Related Work

6.2.2.1 Differential Evolution

Differential Evolution (DE) is an EA designed by Storn and Price [191] to solve global optimization problems. Although the canonical DE was designed to deal with continuous domains, it also shows great performance on combinatorial optimization problems. However, it is not possible to directly apply the traditional DE on continuous domains. Overall DE shows remarkable performance on both types of optimization problems such as space trajectory optimization [201] and multi-area economic dispatch [183] as global optimization problems and a number of permutation problems as combinatorial ones [151, 162, 223].

DE is a popular EA due to its strong search space exploration. It incorporates a differential formulation mechanism to generate offspring from current individuals. Its mechanism includes both mutation and crossover operators which will be applied on all individuals of all generations. Each individual which is also considered as a solution is represented by a D-dimensional vector of real-value numbers. This vector is so-called target vector denoted by $X_{i,g}$, the i^{th} target vector of generation g:

$$X_{i,g} = [x_{1,i,g}, \quad x_{2,i,g}, \quad ..., \quad x_{D,i,g}] \tag{6.1}$$

where $x_{j,i,g}$ denotes the value of target vector $X_{i,g}$ for dimension j ranging from 1 to D.

In each generation g the mutation operator is applied on each target vector $X_{i,g}$ to generate a mutant vector $V_{i,g}$. Although there are a number of different mutation strategies introduced for DE, the general strategy is to incorporate Equation 6.2:

$$V_{i,g} = X_{r1,g} + F \times (X_{r2,g} - X_{r3,g}) \tag{6.2}$$

where $X_{r1,g}$, $X_{r2,g}$, and $X_{r3,g}$ are three different target vectors which are randomly selected within the same generation g. $X_{r1,g}$ is called the base vector and the other two are called perturbation vectors. F is a scale factor to determine how much the base vector $X_{r1,g}$ should be perturbed.

Clearly, in this equation the mutant vector $V_{i,g}$ is calculated without considering any values of the target vector $X_{i,g}$, but modern equations may use the target vector values or some of its information such as its locality. Each mutation strategy includes a base vector and a number of perturbation levels. Since the general strategy incorporates a randomly selected base vector and one level of perturbation it is called $DE/rand/1$. Four different strategies are introduced by Price *et al.* [161], and one more is proposed by Wisittipanich and Kachitvichyanukul [207].

A comparative study is provided by Mezura-Montes *et al.* [133] to compare eight DEs incorporating different mutation and crossover operators. The authors claim that $DE/best/1/bin$ is the most competitive DE regardless of the problem domain. It should be noted that *bin* refers to the crossover operator which is described as follows and mutation operator $DE/best/1$ is presented in Equation 6.3:

$$V_{i,g} = X_{best,g} + F \times (X_{r1,g} - X_{r2,g}) \tag{6.3}$$

where $X_{r1,g}$ and $X_{r2,g}$ are randomly selected target vectors within generation g, $X_{best,g}$ represents the best solution within the same generation, and F is a scale factor.

After applying the mutation operator and generating mutant vector $V_{i,g}$, crossover operator is applied on both target vector $X_{i,g}$ and mutant vector $V_{i,g}$ to generate a trial vector $Z_{i,g}$. The most popular crossover strategy for DE is binomial crossover operator which is represented in Equation 6.4:

$$z_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } r_j \leq Cr \text{ or } j = j_{rand} \\ x_{j,i,g} & \text{otherwise} \end{cases} \tag{6.4}$$

where r_j is a random number uniformly distributed in interval $[0, 1)$ for j^{th} dimension, Cr is the crossover probability which could be fixed for all generations or changing over the generations, and j_{rand} is a randomly selected index to ensure that the trial vector $Z_{i,g}$ differs from target vector $X_{i,g}$ at least in one component.

Like other evolutionary methods, the final step of each generation is applying a selection mechanism. The selection function selects the better solution between target vector $X_{i,g}$ and trial vector $Z_{i,g}$ by comparing their objective values. The selected solution is considered as a target vector for the next generation denoted by $X_{i,g+1}$.

$$X_{i,g+1} = \begin{cases} Z_{i,g} & \text{if } f(Z_{i,g}) \leq f(X_{i,g}) \\ X_{i,g} & \text{otherwise} \end{cases} \tag{6.5}$$

6.2.2.2 Cultural Algorithm

Cultural Algorithm (CA) is an EA incorporating knowledge to direct the search process [175]. A large number of successful applications of CA shows the performance of a knowledge-based EA. The knowledge is extracted and then incorporated by a CA to amend its search mechanism. The extracted knowledge helps the CA to find solutions with better quality and moreover it improves the convergence rate.

Figure 6.1 illustrates the architecture of a CA. As depicted in the figure, CA has a population space like other EAs where the individuals are being evolved. This space is managed by an EA such as a GA or a DE. CA has one more space which is

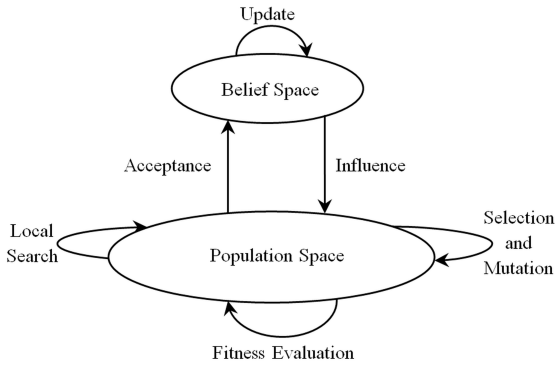


Figure 6.1: CA Architecture

called the belief space. Belief space is incorporated to save and update the extracted knowledge over all generations. In each generation, both spaces communicate to each other using two communication links, namely the acceptance and influence functions. The knowledge circulation is described as follows.

1. The belief space receives the top best individuals within generation g from the population space using acceptance function.

2. The belief space updates its own knowledge.

3. In the next generation $g + 1$, the belief space sends the updated knowledge through the influence function to the population space.

4. The population space incorporates the knowledge to generate offspring from generation g and produce next generation $g + 1$.

5. The top individuals within generation $g+1$ are sent to the belief space to update its knowledge.

This routine continues until the CA stops. It would seem that the population space of a CA works like other EAs, but it uses knowledge-based evolutionary operators instead of random ones.

It should be noted here that the local search presented in Figure 6.1 is not a specific component of a CA but it is usually incorporated by CAs in order to improve the convergence rate.

As mentioned before, EAs suffer from their immature convergence and one possible solution is to incorporate multiple populations. There are a number of Multi-Population Cultural Algorithms (MP-CAs) introduced in the literature, the first MP-CA being introduced by Digalakis and Margaritis [51] to schedule electrical generators. Recently, there has been more effort made by researchers to benefit the multi-population concept and a number of strategies have been proposed to design MP-CAs. Peer local CAs [211, 76, 168], for instance, is one of these strategies in which each sub-population is managed by a local CA having its own local belief space.

There are more parameters which can be adjusted for a MP-CA compared to a single-population CA. For instance, the number of sub-populations could be either fixed or dynamic, the local CAs could be homogeneous or heterogeneous, and the communication topology could be either a ring, a mesh or something else. Furthermore, the information which is migrated could be either the best found solution within each sub-population, the refined algorithm parameters or the knowledge extracted during the previous generations.

6.2.3 Numerical Optimization

Global optimization problems is a class of optimization problem where the input parameters are continuous and the goal is to minimize or maximize an optimization function. Since both minimization and maximization problems work the same, we only consider the minimization approach which is mathematically defined in Equation 6.6:

$$
\begin{aligned}
Minimize \quad & f(X) \\
& X = \{x_1, x_2, ..., x_D\} \\
& f : \mathbf{R}^D \rightarrow \mathbf{R} \\
Subject\ to \quad & g_i(X) = 0, i = 1, ..., m \\
& h_j(X) \leq 0, j = 1, ..., n
\end{aligned}
\tag{6.6}
$$

where X is a D-dimensional vector of real-value parameters and x_i denotes the value for i^{th} dimension. $g_i(X)$ and $h_j(X)$ are a number of equality and inequality constraints, correspondingly.

Minimization is defined as finding vector x^* which has the lowest objective value among all possible vectors in the solution space S.

$$
X^* \in S \qquad and \qquad f(X^*) \leq f(X), \forall X \in S
$$

Numerical optimization is a global optimization problem where the optimization function is a mathematical equation dependent to a vector of input parameters. Sphere model presented in Equation 6.7 is a sample of numerical optimization functions where the goal is to find the minimum value for this model and both upper and lower bounds are considered for each dimension.

$$
f_1(X) = \sum_{i=1}^{D} {x_i}^2 \tag{6.7}
$$

$$
-100 \leq x_i \leq 100
$$

$$
min(f_1) = f_1(0, ..., 0) = 0
$$

6.2.4 Proposed Method

A new architecture is proposed for MP-CA incorporating a number of heterogeneous sub-populations. The optimization parameters are divided among these sub-populations such that each sub-population which is directed by a local CA optimizes its own parameters. Based on the parameters assigned to each sub-population, it defines a partial solution structure. Therefore, each sub-population is a set of partial solutions instead of complete solutions. The proposed architecture called Heterogeneous Multi-Population Cultural Algorithm (HMP-CA) incorporates a shared belief space to be contacted by all local CAs.

Figure 6.2 illustrates the proposed architecture which is composed of a number of heterogeneous local CAs and a shared belief space. The shared belief space in our proposed architecture is a very simple space which records only best parameter for each dimension with its corresponding objective value. Moreover, the proposed method in this architecture incorporates the simplest DE ($DE/rand/1$ presented in Equation 6.2 with binomial crossover illustrated in Equation 6.4) as the evolutionary approach for local CAs. The crossover probability of the simplest DE is set to 0.5 for all generations, and the scale factor is selected randomly from intervals $[0.5, 2.5]$ for each generation. These parameters are adjusted based on an extensive experiments.

Incorporating heterogeneous local CAs is introduced in Cultural Cooperative Particle Swarm Optimization (CCPSO) which is published by Lin *et al.* [117, 118, 120]. CCPSO is designed to train a Neurofuzzy Inference System (NFIS). Each local CA in CCPSO is a PSO-based method with its local belief space working to optimize one variable. Shared belief space is also used in Cooperative Interactive Cultural Algorithms (CICA) [80, 83] to deal with interactive optimization problems. Therefore, the proposed architecture is a new one which benefits both a shared belief space and heterogeneous characteristic of local CAs.

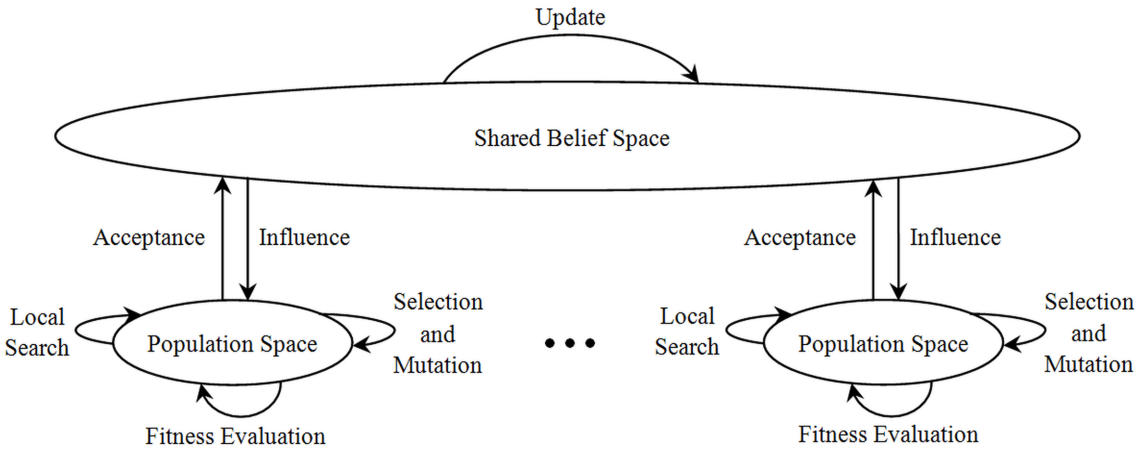


Figure 6.2: HMP-CA Architecture

The details of dimensions division strategy, population space's definition and the structure of the belief space are described in the following sub-subsections, and this subsection is finalized with a precise description of the proposed HMP-CA's framework.

6.2.4.1 Dimensions Division

The first step of our proposed HMP-CA is parameter division in which the optimization parameters are divided among the sub-populations. The division is designed such that the difference of the number of dimensions between each pair of local CAs is one at maximum. The proposed division strategy assigns the dimensions one by one to different local CAs sequentially and when it reaches the last local CA, it starts from the first one. The following example shows the division of 10 dimension among 4 local CAs. In this example the first two local CAs get three dimensions each and others get two dimensions each.

Local CAs	1	2	3	4
Dimensions	1	2	3	4
	5	6	7	8
	9	10		

6.2.4.2 Population Space

The population space of our proposed methods consists of a number of sub-populations. The individuals of a sub-population include only the dimensions that are assigned to the corresponding sub-population. Therefore these individuals are called partial solutions. Considering the example presented in Sub-subsection 6.2.4.1, the partial solutions of the second sub-population only have the parameters for dimensions 2, 6 and 10.

In order to evaluate a partial solution, it is completed by its complement parameters coming from the belief space. The complete individual then is evaluated based on a numerical optimization function. Incorporating this strategy makes parameters comparison fair since the complements of all partial solutions within a population at a given generation are exactly the same.

Furthermore, since only the partial solutions are recorded within a sub-population, the proposed HMP-CA is an efficient method in terms of both CPU time and memory usage. Considering the instance illustrated in Sub-subsection 6.2.4.1, a partial solution in the first two sub-populations is a 3-dimensional vector and for the other sub-populations it is a 2-dimensional vector which are much shorter than a complete solution with a 10-dimensional vector. Moreover, to apply the mutation and crossover operators only the partial solutions are taken into account. It means that for the same example only 2 or 3 dimensions are considered to be used in Equations 6.2 and 6.4 instead of 10 dimensions. It implies that the CPU time required for calculating these equations is approximately saved by 70%.

6.2.4.3 Belief Space

The HMP-CA's belief space has a very simple structure. It only has a vector of the best found parameters and a vector of their corresponding objective value. Figure 6.3 illustrates a sample belief space for a 10-dimensional problem. Actually, this

belief space could be the belief space of applying our proposed HMP-CA to solve a 10-dimensional Sphere Model (presented in Equation 6.7) when the best solution is found.

Dimensions	1	2	3	4	5	6	7	8	9	10
Parameters	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Values	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 6.3: A sample belief space.

When belief space receives a best partial solution from a local CA, it updates its parameters of the corresponding dimensions if the objective value of that partial solution is better than that of those dimensions. Although each sub-population is improving during the process of evolution and they never sends worse partial solutions than the ones they sent before, the idea of comparing the objective value is due to the fact that the belief space may incorporate a local search method to optimize all the dimensions at the same time.

6.2.4.4 Local Search

In order to speed up the convergence of the proposed method, a simple local search heuristic is incorporated which is looking into the neighborhood of each dimension to find out the local optimum parameter. The pseudo-code of the proposed local search is presented in Figure 6.4. The local search method can be called either by each sub-population or by belief space. The belief space calls it by sending a complete solution while sub-populations call it by sending a partial solution. The local search heuristic defines the $factor$ parameter and incorporating addition, subtraction and cut operators, it looks for better parameters for the dimensions presented in the given solution. Cut is considered as removing the tenth and higher decimal digits. Then if a better solution is obtained, the local search continues with the best found solution.

```
PROCEDURE: Local Search
INPUT: Partial Solution or Complete Solution
OUTPUT: Local Optimal Solution
BEGIN
    factor ← 10
    FOR (IterationNo)
        FOR (each dimension i)
            minusClone[i] ← input[i] × (1 + factor)
            plusClone[i] ← input[i] × (1 − factor)
            cuttedClone[i] ← Cut(input[i])
        END
        Evaluate all three new solutions.
        Assign the best one to bestClone.
        IF (bestClone is better than input)
            input ← bestClone
        ELSE
            factor ← 0.1 × factor
        END
    END
    Output input.
END
```

Figure 6.4: The pseudo-code for local search.

Otherwise, $factor$ parameter is decreased. This routine continues for $IterationNo$ iterations.

The $factor$ parameter is initialized by 10, and if no improvement is made, it continues to be decreased by 90%. Therefore the value of the $factor$ parameter in a given iteration is one of the following values.

$$\left[10, 1, 0.1, 0.01, ..., 10^{(2-IterationNo)}\right]$$

Since the local search heuristic is an additional component for the proposed HMP-CA, the HMP-CA which incorporates the local search method is called HMP-CA + LS.

PROCEDURE: HMP-CA Framework
INPUT: Test Problems and Algorithm Parameters
OUTPUT: Optimal or Near-Optimal Solution

BEGIN
    Initialize *LocalCAsNo* Local CAs.
    Divide D Dimensions among local CAs.
    Generate initial sub-populations.
    FOR ($MaxGen$)
        FOR (*each subpopulation*)
            Apply DE's operators to generate offspring.
            Evaluate all partial solutions and sort them.
            Call local search method.
            Update belief space.
        END
    END
    Output the best found individual so far.
END

Figure 6.5: The framework of proposed HMP-CA

6.2.4.5 Framework

As depicted in Figure 6.5, the framework of the proposed HMP-CA starts by initializing $LocalCAsNo$ local CAs. The D dimensions are then divided among local CAs using the approach presented in Sub-subsection 6.2.4.1. Then each local CA initializes its own sub-population with a number of randomly generated partial solutions. It should be noted here that the partial solutions from different sub-populations are different. The local CAs run for $MaxGen$ generations and then returns their best found solution.

In each generation, local CAs evolve their own sub-population of partial solutions using DE's operators to generate offspring population. The generation step is followed by the evaluation of the new partial solutions. After sorting the new partial solutions, each local CA calls local search method by sending its own best found solution. Finally, the local CAs sends their own best found solutions through the acceptance function to update belief space. It should be noted here again that the influence function is used when a partial solution is evaluated.

As mentioned before, the local search method is an additional component and it is not be used for the proposed HMP-CA. When it is incorporated, the proposed method is called HMP-CA + LS.

6.2.5 Experiments and Results

The proposed HMP-CA is implemented by using the java programming language version 1.6.0.18 and evaluated on a system with $Intel(R)\ Core(TM)2Quad\ 2.50GHz$ CPU and $8.00GB$ RAM. The algorithm parameters used in our experiments are listed in Table 6.1. In our experiments, we consider the whole population size to be 1000 individuals to be divided into a number of sub-populations evenly. In case of having 30 sub-populations, the size of each one is 33. The method runs for the maximum of 10000 generations and the local search runs for exactly 10 iterations.

Table 6.1: Parameters of the proposed algorithm

Parameters	Value	Parameters	Value
$WholePopSize$	1000	$MaxGen$	10000
$LocalCAsNo$	30	$IterationNo$	10
$SubPopSize$	33	D	30

In our experiments, we consider a number of well-known numerical optimization functions which are incorporated by different researchers [218, 133]. Appendix 6.2.7 presents a list of all incorporated functions with their corresponding optimal solutions. For our experiment, we used the common number of dimension which is 30. For each function, the experiments are done 10 times independently.

The results of applying the proposed HMP-CA shows that it is able to find the minimum value for seven numerical optimization functions out of 8. It cannot find the minimum value for function f_2 in an acceptable time because of the dependency between different dimensions. Actually, the HMP-CA does not converge for function f_2 in an acceptable time and it continues looking for smaller value. When the proposed

Table 6.2: The number of generations required to find the best solution for both proposed methods with different number of sub-populations

Problem	Min	HMP-CA				HMP-CA + LS			
		5	10	15	30	5	10	15	30
f_1	**0.00E+00**	7834.9	4122.7	3061.1	1794.7	3.0	2.0	2.0	2.0
f_2	**0.00E+00**	-	-	-	-	2.0	2.0	2.0	2.0
f_3	**-12569.49**	463.7	184.6	118.0	61.7	118.2	55.5	42.9	23.5
f_4	**0.00E+00**	7998.6	4301.0	2933.8	1582.9	127.0	39.5	20.2	4.1
f_5	**0.00E+00**	10000+	4376.5	3038.8	1782.9	3.0	2.0	2.0	2.0
f_6	**0.00E+00**	7361.8	4525.5	2764.3	1663.0	3.0	2.0	2.0	2.0
f_7	**1.57E-32**	1132.5	539.8	382.5	210.5	2.0	2.0	2.0	2.0
f_8	**1.35E-32**	1103.7	581.0	375.8	220.3	2.0	2.0	2.0	2.0

method incorporates the local search heuristic, it is able to find the optimum value for all experimented functions very quickly. As presented in Table 6.2, the proposed HMP-CA + LS is able to find the minimum value for all test functions in less than 25 generations.

The effect of the number of local CAs is also evaluated for the proposed HMP-CA. Four different numbers of local CAs are considered for this comparison including 5, 10, 15, and 30. When the number of local CAs is increased, the number of dimensions to be optimized in one local CAs is decreased. For instance, when there are 30 local CAs for a 30-dimensional problem, each local CA is supposed to optimize only one dimension, while with 10 local CAs each one gets 3 dimensions to optimize. Therefore, the higher number of local CAs results in a faster convergence. The results of this evaluation are presented in Table 6.2. In this table, each value is the average number of generations when an optimal value is reached for each test function with each number of local CAs. The results confirm that when the number of local CAs increases, the generations required to find the minimum value decreases.

It should be mentioned here that the HMP-CA is not able to find the minimum value for function f_5 just in 10000 generations when incorporating 5 sub-populations, but it is able to do so if it gets the permission to use higher number of generations.

Therefore, the number of generations required to find the minimum value for this case is represented as 10000+.

Furthermore, Table 6.2 illustrates that the proposed HMP-CA + LS is able to find the optimal solution for all the test problems within 200 generations. In other words, the proposed HMP-CA + LS can find the optimal solution for all test problems using less than 200,000 fitness evaluation.

In order to compare our proposed method with the state-of-the-art methods, a number of recently published methods are considered. It should be noted here that since there is no MP-CA proposed to deal with numerical optimization and because our local CAs are DE-based CAs, a number of multi-population DEs are considered for comparison. These algorithms include jDE [23], JADE [222], PLADE [220], and MPDE [218]. These methods incorporate different strategies to optimize the control parameter of a DE. jDE incorporates an aging mechanism to deal with dynamic optimization problems and JADE incorporates historical knowledge to adapt DE's parameters. PLADE benefits a learning strategy inspired by Particle Swarm Optimization and MPDE uses an adaptive parameter control based on parameters of the most successful local DE within a generation.

The results of the comparison are presented in Table 6.3. Each values in this table is the average of the best values obtained in 10 independent runs and the minimum values are emphasized in bold face. Although the minimum value for functions f_7 and f_8 is mathematically zero, it is not possible for a computer program to reach to that minimum and this is due to the fact that the results of functions *sin* and *cos* are calculated approximately.

The results presented in Table 6.3 show the the only algorithm which is able to offer the minimum values for all the eight numerical optimization functions is the proposed HMP-CA + LS. Furthermore, the second best algorithm is HMP-CA which is able to find the minimum value for seven functions out of eight.

Table 6.3: Comparison among Different Multi-Population Differential Evolution

Problem	jDE [23]	JADE [222]	PLADE [220]	MPDE [218]	HMP-CA	HMP-CA + LS
f_1	3.05E-31	2.09E-58	1.75E-29	8.62E-43	**0.00E+00**	**0.00E+00**
f_2	2.39E-01	7.97E-01	7.97E-02	3.76E-30	8.59E-16	**0.00E+00**
f_3	**-12569.49**	-12498.40	**-12569.49**	-12567.12	**-12569.49**	**-12569.49**
f_4	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_5	4.28E-15	6.34E-15	4.71E-15	4.14E-15	**0.00E+00**	**0.00E+00**
f_6	**0.00E+00**	1.53E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_7	3.93E-32	7.60E-30	3.06E-30	**1.57E-32**	**1.57E-32**	**1.57E-32**
f_8	2.84E-31	**1.35E-32**	4.69E-30	**1.35E-32**	**1.35E-32**	**1.35E-32**

6.2.6 Conclusions

In this article, a new architecture for MP-CA is proposed incorporating heterogeneous local CAs. The local CAs are supposed to optimize different sets of parameters. The proposed method which is called Heterogeneous Multi-Population Cultural Algorithm (HMP-CA) defines a belief space shared among all the local CAs. The local CAs send their best combinations to the shared belief space to update its record. The proposed HMP-CA which is an efficient method in terms of both CPU time and memory usage is effective such that it presents competitive results compared to the state-of-the-art methods.

Furthermore, a local search heuristic is proposed which is a very powerful mechanism to accelerate the HMP-CA's convergence. The results show that the HMP-CA + LS is able to find the minimum values for all the experimented functions very quickly. It is possible to claim that the proposed HMP-CA outperforms all the state-of-the-art methods.

Although the HMP-CA offers a remarkable performance over all the eight test functions, it should be evaluated by more complex functions with higher dimensionality which is considered as a future direction. Another direction is to optimize the DE's parameter instead of using random values.

Acknowledgment

This work is made possible by a grant from the National Science Foundation and NSERC Discovery No. 327482.

6.2.7 Optimization Functions

The details of the optimization functions used in our experiments are presented as follows.

f_1 - Sphere Model:

$$f_1(X) = \sum_{i=1}^{D} {x_i}^2 \tag{6.8}$$

$$-100 \leq x_i \leq 100$$

$$min(f_1) = f_1(0, ..., 0) = 0$$

f_2 - Generalized Rosenbrock's Function:

$$f_2(X) = \sum_{i=1}^{D-1} \left| 100 \left(x_{i+1} - {x_i}^2 \right)^2 + (x_i - 1)^2 \right| \tag{6.9}$$

$$-30 \leq x_i \leq 30$$

$$min(f_2) = f_2(1, ..., 1) = 0$$

f_3 - Generalized Schwefel's Problem 2.26:

$$f_3(X) = \sum_{i=1}^{D} \left(-x_i sin\left(\sqrt{|x_i|} \right) \right) \tag{6.10}$$

$$-500 \leq x_i \leq 500$$

$$min(f_3) = f_3(420.9687, ..., 420.9687) \simeq -12569.5$$

f_4 - Generalized Rastrigin's Function:

$$f_4(X) = \sum_{i=1}^{D} \left[{x_i}^2 - 10cos\left(2\pi x_i\right) + 10 \right] \tag{6.11}$$

$$-5.12 \leq x_i \leq 5.12$$

$$min(f_4) = f_4(0, ..., 0) = 0$$

f_5 - Ackley's Function:

$$f_5(X) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} {x_i}^2} \right) - \exp\left(\frac{1}{D} \sum_{i=1}^{D} cos\left(2\pi x_i\right) \right) + 20 + e \tag{6.12}$$

$$-32 \leq x_i \leq 32$$

$$min(f_5) = f_5(0, ..., 0) = 0$$

f_6 - Generalized Griewank's Function:

$$f_6(X) = \frac{1}{4000} \sum_{i=1}^{D} {x_i}^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1 \tag{6.13}$$

$$-600 \leq x_i \leq 600$$

$$min(f_6) = f_6(0, ..., 0) = 0$$

f_7 and f_8 - Generalized Penalized Functions:

$$f_7(X) = \frac{\pi}{D}\Bigg\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_D-1)^2\Bigg\} + \sum_{i=1}^{D} u(x_i,10,100,4) \quad (6.14)$$

$$-50 \le x_i \le 50$$

$$min(f_7) = f_7(-1,...,-1) = 0$$

$$f_8(X) = 0.1\Bigg\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right] + (x_D-1)^2\left[1+\sin^2(2\pi x_D)\right]\Bigg\} + \sum_{i=1}^{D} u(x_i,10,100,4) \quad (6.15)$$

$$-50 \le x_i \le 50$$

$$min(f_8) = f_8(1,...,1) = 0$$

where

$$u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m & \text{if } x_i > a \\ 0 & \text{if } -a \le x_i \le a \\ k(-x_i-a)^m & \text{if } x_i < -a \end{cases}$$

$$y_i = 1 + \frac{1}{4}(x_i+1)$$

6.3 A New Dimension Division Scheme for Heterogeneous Multi-Population Cultural Algorithm

Abstract. Heterogeneous Multi-Population Cultural Algorithm (HMP-CA) is a new class of Multi-Population Cultural Algorithms which incorporates a number of local Cultural Algorithms (CAs) designed to optimize different subsets of the dimensions of a given problem. In this article, various dimension decomposition techniques for HMP-CAs are proposed and compared. The concept of using a dimension decomposition scheme which does not result in populations having the same number of dimensions is implemented, and named imbalanced dimension division. All the techniques are evaluated using a number of well-known benchmark optimization functions and two measures are defined in order to compare them including success rate and convergence ratio. The results show that imbalanced dimension division schema works better with a higher number of local CAs, and outperforms all the other evaluated techniques in both measures.

6.3.1 Introduction

Evolutionary Algorithms (EAs) are subset of optimization methods successfully applied to solve problems in various research areas. EAs are population-based approaches incorporating the concept of evolution inspired by natural selection; that individuals which survive for the next generation are more likely to be fitter for their environment. An EA starts with a population of randomly generated individuals called the initial population. New populations are generated by applying evolutionary

operators on the individuals from the previous population, followed by incorporating a selection method afterward for finalization. This routine halts when a predefined termination criterion (*e.g.* CPU time, predefined number of generations) has been met.

A Cultural Algorithm (CA) is an EA incorporating knowledge to improve the search mechanism. In a CA, knowledge is extracted, stored, and updated in a space separate form the population, called the belief space. The recorded knowledge in the belief space is then used to direct the search process during evolution.

Although CAs are successfully applied to various types of optimization problems, they suffer from immature convergence. There are a number of strategies introduced to resolve converging to local optima, however researchers have recently been interested in incorporating multiple populations to increase diversity and escape to the global optimum. The multiple populations approach divides the whole population into a number of sub-populations, with each sub-population being evolved using a local CA independently while the local CAs communicate with each other between generations.

In addition to incorporating multiple populations in CAs, we recently showed that incorporating heterogeneous local CAs offers better solutions compared to the homogeneous ones. We proposed a Heterogeneous Multi-Population Cultural Algorithm (HMP-CA) to deal with numerical optimization problems where the goal is to optimize mathematical functions with a D-dimensional vector of continuous parameters [171]. The results showed that dividing the dimensions among the local CAs improves the convergence rate. Furthermore, by incorporating partial solutions our published method saves both memory and CPU usage while handling the same total number of individuals in each generation.

There are various techniques to divide the dimensions among local CAs in a HMP-CA. In this article, an analysis is conducted to study the performance of these tech-

niques. Our published HMP-CA [171] is modified such that it is capable of dealing with a number of dimension division strategies and any number of local CAs. The modified HMP-CA includes a number of heterogeneous sub-populations with a shared belief space. Each sub-population includes only partial solutions corresponding to its assigned dimensions and sends its best partial solutions to the belief space every generation. For each dimension, the shared belief space keeps the best parameters with their corresponding objective value. In order to evaluate the strategies, a number of benchmark numerical optimization functions are incorporated. Each strategy is investigated with variable number of sub-populations.

The structure of this article is as follows. Subsection 6.3.2 briefly describes CA and its multi-population version, followed by a concise description of HMP-CA in Subsection 6.3.3. Subsection 6.3.4 illustrates the proposed HMP-CA in detail, followed by representing the investigated dimension decomposition strategies in Subsection 6.3.5. Subsection 6.3.6 represents the experiments and the discussion on the results, followed by representing concluding remarks and future directions in Subsection 6.3.7.

6.3.2 Cultural Algorithm

Cultural Algorithm (CA) as developed by Reynolds [175] is an EA which extracts knowledge during the evolutionary process in order to redirect the search process. CA incorporates two spaces, namely a population space and a belief space. The population space evolves individuals over generations while the belief space is responsible for storing and updating the extracted knowledge. CAs incorporate a two-way communication between the spaces, namely an acceptance function which transfers the best individuals from the population space to the belief space and an influence function which carries the extracted knowledge from the belief space to influence the operations in the population space. The evolution process in population space can

be managed by any evolutionary algorithm such as a Genetic Algorithm (GA) or a Differential Evolution (DE).

In spite of their successful application in various fields, CAs suffer from immature convergence. The main reason is the fact that they do not preserve population diversity over generations. There are a number of strategies introduced to overcome this limitation such as rejecting duplicate solutions or having a high mutation rate. Incorporating multiple populations is another such strategy which represents a good performance in order to do so. The first Multi-Population Cultural Algorithm (MP-CA) was introduced by Digalakis and Margaritis [51] which was incorporated to schedule electrical generators. The main characteristic of a MP-CA is its architecture which determines the knowledge propagation within the local CAs. There are different architectures proposed to implement a MP-CA. The most common one is homogeneous local CAs [211, 76, 168] in which there are a number of homogeneous local CAs with their own local belief spaces cooperating to find the best solution.

Heterogeneous local CAs is another class of architectures for MP-CA where the sub-populations are heterogeneous such that each sub-populations is working on different dimensions. Lin *et al.* [117, 118, 120] introduced heterogeneous local CAs by proposing their Cultural Cooperative Particle Swarm Optimization (CCPSO) to train a Neurofuzzy Inference System (NFIS). In CCPSO, each local CA has its own local belief space and incorporates a Particle Swarm Optimization (PSO) to evolve its corresponding sub-population. In this framework each local CA is responsible for optimizing only one variable.

In addition to the architecture, incorporating multiple populations brings more algorithm parameters to be adjusted in order to get a good performance. The number of sub-populations, the communication topology, and the type of migrated knowledge are three instances of MP-CAs parameters.

6.3.3 Heterogeneous Multi-Population Cultural Algorithm

Heterogeneous Multi-Population Cultural Algorithm (HMP-CA) is an architecture in the class of Heterogeneous local CAs in which there is only one belief space shared among all local CAs [171]. In this architecture, there are a number of local CAs working on optimization of different subsets of the all dimensions. The only one shared belief space is responsible to keep a track of the best parameters found for each dimension. The focus of this paper is on this architecture where there are various strategies to divide dimensions among local CAs.

In HMP-CA, each local CA is designed to handle only its assigned dimensions. Therefore instead of complete solutions, each sub-population provides a set of partial solutions. For instance, if a local CA is responsible to optimize the first three dimensions of a 30-dimensional optimization problem, it only works with the values for the first three dimensions of a solution meaning only a 3-dimensional vector is used instead of a complete 30-dimensional solution. In other words, in this example a sub-population including 100 partial solutions deals with only 300 parameters in each generation while a sub-population with the same number of complete solutions incorporates 3,000 parameters. Due to this huge deduction in the number of parameters for each generation, the HMP-CA is an efficient method in terms of both CPU time and memory usage.

It should be mentioned here that in order to evaluate a partial solution, it is completed by the complement of its parameters coming from the belief space. This mechanism provides a fair comparison platform for partial solutions such that all the partial solutions are completed with the same parameters complement.

Heterogeneous local CAs incorporating either a shared belief space or local belief spaces are initialized by dividing the dimensions. Lin *et al.* [117, 118, 120] assigned each dimension to a local CA, and in our published HMP-CA [171] a jumping strategy was incorporated for dimension decomposition. There are more dimension division

strategies, the most common ones are investigated in this study which are represented in Subsection *Dimension Decomposition Strategies*.

6.3.4 Proposed Method

The architecture of the proposed method is similar to our recently published HMP-CA [171]. A number of sub-populations including only partial solutions are incorporated alongside a shared belief space. In the published HMP-CA, sub-populations cooperate with each other by transferring only their best partial solution to the belief space, and the belief space updates its only one parameter for each dimension with its corresponding objective value. In contrast, the architecture of the proposed method provides the flexibility to set the number of best partial solutions to be transferred to the belief space as well as to set the number of parameters to be stored in the belief space for each dimension.

Furthermore, the published HMP-CA incorporates only one static dimension division strategy, while the proposed method is capable to deal with any dimension decomposition technique such as sequential, logarithmic, and even customized ones.

Like the published HMP-CA, in order to evolve the sub-population a simple DE is incorporated which benefits from $DE/rand/1$ mutation represented in Equation 6.16, binomial crossover illustrated in Equation 6.17, and a selection mechanism depicted in Equation 6.18. Due to the extensive experiments conducted in the published HMP-CA, the scale factor for Equation 6.16 is selected randomly from intervals $[0.5, 2.5]$ for each generation and the crossover probability for Equation 6.17 is set to 0.5 for all generations.

$$V_{i,g} = X_{r1,g} + F \times (X_{r2,g} - X_{r3,g}) \tag{6.16}$$

Table 6.4: A sample belief space with size 3.

Dimensions		1	2	3	4	5
Set 1	Parameters	0.0	0.0	-0.88	-0.23	-0.43
	Objective	1.01	1.01	1.01	1.01	1.01
Set 2	Parameters	0.01	-0.26	1.06	-0.24	-0.44
	Objective	1.02	1.09	1.37	1.01	1.01
Set 3	Parameters	0.39	-0.31	-0.73	-0.62	0.79
	Objective	1.19	1.12	1.49	1.35	1.79

where $X_{r1,g}$, $X_{r2,g}$, and $X_{r3,g}$ are three randomly selected target vectors within the same generation g. F is a scale factor to determine how much the base vector $X_{r1,g}$ should be perturbed by the difference of the other two.

$$z_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } r_j \leq Cr \text{ or } j = j_{rand} \\ x_{j,i,g} & \text{otherwise} \end{cases} \tag{6.17}$$

where r_j is a real number randomly selected within the interval $[0, 1)$ for j^{th} dimension, and Cr is the crossover probability. In order to ensure that the trial vector $Z_{i,g}$ differs from target vector $X_{i,g}$ at least in one component, j_{rand} is randomly selected as the index of the different dimension.

$$X_{i,g+1} = \begin{cases} Z_{i,g} & \text{if } f(Z_{i,g}) \leq f(X_{i,g}) \\ X_{i,g} & \text{otherwise} \end{cases} \tag{6.18}$$

where $X_{i,g}$ and $Z_{i,g}$ denote a target vector and its corresponding trial in generation g, respectively and $X_{i,g+1}$ represents the selected target vector for the next generation.

A shared belief space is incorporated in the proposed HMP-CA which tracks the best found parameters with their corresponding objective values for each dimension. A sample belief space of size 3 is illustrated in Table 6.4 which is incorporated to solve a 5-dimensional sphere model.

The framework of the belief space starts with receiving a number of best partial solutions from a local CA. For each parameter of each partial solution, the corresponding records are reviewed in order to avoid pushing duplicate parameters. Therefore, for the existing parameters only the objective value is updated and the new parameters are inserted into the belief space if they have a better objective value.

The belief space influences the search direction only in fitness evaluations. When a local CA requires a partial solution to be evaluated the belief space is queried for the complement of its set of parameters, the belief space returns the complement parameters which are randomly selected from its recorded parameters.

The number of local CAs in the proposed HMP-CA and their corresponding assigned dimensions are dependent to its dimension division strategy. Therefore the proposed HMP-CA starts by receiving a dimension division strategy. Based on the given strategy, each local CA is initialized with its corresponding sub-population of randomly generated partial solutions. The local CAs are evolved incorporating the previously mentioned DE for the maximum number of generations such that they transfer their best partial solutions to the belief space in every generation and they request the belief space for the complement parameters for each fitness evaluation.

In order to avoid trapping into local optimal regions within a sub-population, a re-initialization mechanism is incorporated such that if a local CA cannot find a better solution for a number of generations its sub-population will be re-initialized with random partial solutions while the belief space continues with its recorded parameters. It should be mentioned that in the proposed HMP-CA, there is no local search method incorporated to speed up the convergence.

6.3.5 Dimension Decomposition Strategies

In this article, a number of different dimension division strategies are studied which can be categorized into two classes including balanced and imbalanced techniques.

The division strategies which assigns the same number of dimensions to each local CA is classified as balanced divisions. In balanced divisions, the difference of the number of dimensions between any two local CAs is at most one. All decomposition strategies where this difference could be greater than one are considered to be imbalanced division methods.

Within the balanced category, sequential division, jumping division, and overlapped sequential division are investigated. The jumping strategy assigns the adjacent dimensions to different local CAs while the sequential strategy assigns the adjacent dimensions to the same local CA. The overlapped sequential division is a modified version of sequential division such that each dimension is assigned to two different local CAs.

A logarithmic division method is considered as well, belonging to the category of imbalanced strategies. It starts by assigning all the dimensions to the first local CA. Two additional local CAs are generated by selecting each half of the dimensions of the first local CA. Remaining local CAs are generated in a similar fashion by recursively dividing the dimensions in half.

Table 6.5 illustrates the described dimension decomposition strategies dividing 30 dimensions among 15 local CAs. As mentioned before, the balanced strategies assign almost the same number of dimensions to each local CA, while the imbalanced ones assign various number of dimensions to each local CA ranging from one dimension to the all dimensions.

For a 30-dimensional optimization problem, the logarithmic strategy can generate up to 59 local CAs including one local CA with 30 dimensions, two local CAs with 15 dimensions, 4 local CA with 7 or 8 dimensions, 8 local CAs with 3 or 4 dimensions, 14 local CAs of 2 dimensions, and 30 local CAs including only one dimension.

In this article, a new strategy similar to the logarithmic strategy is also defined which is called customized logarithmic. The new strategy incorporates 35 local CAs

Table 6.5: Dividing 30 dimension into 15 local CAs incorporating different strategies.

Local CAs	Balanced			Imbalanced
	Jumping	Sequential	Overlapped Sequential	Logarithmic
1	1,16	1, 2	1, 2, 3, 4	1, 2, 3, ..., 30
2	2, 17	3, 4	3, 4, 5, 6	1, 2, 3, ..., 15
3	3, 18	5, 6	5, 6, 7, 8	16, 17, 18, ..., 30
4	4, 19	7, 8	7, 8, 9, 10	1, 2, 3, ..., 8
5	5, 20	9, 10	9, 10, 11, 12	9, 10, 11, ..., 15
6	6, 21	11, 12	11, 12, 13, 14	16, 17, 18, ..., 23
7	7, 22	13, 14	13, 14, 15, 16	24, 25, 26, ..., 30
8	8, 23	15, 16	15, 16, 17, 18	1, 2, 3, 4
9	9, 24	17, 18	17, 18, 19, 20	5, 6, 7, 8
10	10, 25	19, 20	19, 20, 21, 22	9, 10, 11, 12
11	11, 26	21, 22	21, 22, 23, 24	13, 14, 15
12	12, 27	23, 24	23, 24, 25, 26	16, 17, 18, 19
13	13, 28	25, 26	25, 26, 27, 28	20, 21, 22, 23
14	14, 29	27, 28	27, 28, 29, 30	24, 25, 26, 27
15	15, 30	29, 30	29, 30, 1, 2	28, 29, 30

for a 30-dimensional problem which includes 5 local CAs with the following 5 dimension subsets in addition to the 30 local CAs with one dimension each.

$$\{1, 2, 3, ..., 30\}, \{1, 2, 3, ..., 15\}, \{16, 17, 18, ..., 30\}, \{1, 2, 3, ..., 8\}, \{16, 17, 18, ..., 23\}$$

6.3.6 Experiments and Results

In order to evaluate the mentioned strategies, they are incorporated by the proposed HMP-CA considering the total population size to be set to 1000 individuals which can be divided into any number of sub-populations evenly. Considering 15 local CAs in the proposed HMP-CA, each local CA will have 66 solutions within its sub-population. The size of the belief space is set to 3 and sub-populations transfer their 3 best solutions to the belief space in each generation. The maximum generation to find the optimal solution is considered to be 10,000 generations, and if an optimal solution is not obtained before this upper limit the run is considered to be an unsuccessful one.

For each experiment, 100 independent runs were conducted to provide a statistically significant sample size.

A number of well-known numerical optimization functions were considered for the experiments which have been used by various researchers [218, 133]. The 12 considered benchmark problems are listed as follows and are detailed in the aforementioned references:

- f_1 - Sphere Model.
- f_2 - Generalized Rosenbrock's Function.
- f_3 - Generalized Schwefel's Problem 2.26.
- f_4 - Generalized Rastrigin's Function.
- f_5 - Ackley's Function.
- f_6 - Generalized Griewank's Function.
- f_7 and f_8 - Generalized Penalized Functions.
- f_9 - Schwefel's Problem 1.2.
- f_{10} - Schwefel's Problem 2.21.
- f_{11} - Schwefel's Problem 2.22.
- f_{12} - Step Function.

The generalized Rosenbrock's function (f_2), for instance is represented in Equation 6.19:

$$f_2(x) = \sum_{i=1}^{D-1} \left| 100 \left(x_{i+1} - {x_i}^2 \right)^2 + (x_i - 1)^2 \right| \tag{6.19}$$

$$-30 \leq x_i \leq 30$$

$$min(f_2) = f_2(1, ..., 1) = 0$$

In addition to these 12 functions, a modified version of the function f_2 is considered in this analysis which is called modified Rosenbrock's function (f_{2M}) illustrated in Equation 6.20:

$$f_{2M}(x) = \sum_{i=1}^{D-4} \left| 100 \left(x_{i+4} - {x_i}^2 \right)^2 + (x_i - 1)^2 \right| \tag{6.20}$$

In our experiment, the number of dimensions is set to 30 which is a common number in the area of numerical optimization.

In order to evaluate and compare the strategies using the benchmark functions, two measures are defined including *Success Rate (SR)* and *Convergence Ratio (CR)*. The former measure refers to the percentage of runs which find the optimal solution within one experiment (100 independent runs). An experiment with the 100% SR is referred as a reliable experiment. The later measure deals with the average number of generations required to find the optimal solution. Within the conducted experiments, the experiment which has the minimum average is considered to be the base of the CR calculation, which is as follows.

$$ConvergenceRatio = \frac{AVG - AVG_{min}}{AVG_{max} - AVG_{min}} \times 100\%$$

where AVG refers to the average number of generations required to find the optimal solution for an experiment, and AVG_{min} and AVG_{max} denote the minimum and the maximum averages obtained over all experiments with respect to each optimization function. Therefore, the zero CR value represents the experiment which requires the minimum average number of generations to find the optimal solution.

Each dimensions division strategy is evaluated with different number of local CAs. The sequential, jumping, and logarithmic strategies are experimented by 5, 7, 10, 15, and 30 local CAs. Additionally, the logarithmic strategy with 59 local CAs, overlapped sequential with 15 local CAs, and customized logarithmic with 35 local CAs are evaluated. It should be mentioned here that the sequential and jumping strate-

gies with 30 local CAs are equivalent. Therefore, there are 17 different experiments conducted in this study.

The detail results of all experiments are presented in Tables 6.6 and 6.7 where all the numbers are in percentile form. In these tables the Seq, Jump, Over Seq, and Cust refer to the sequential, jumping, logarithmic, overlapped sequential, and customized logarithmic techniques, respectively. For each optimization function, the first row represents CR value, followed by SR value in the second row. The average CR and SR value with respect to the different experiments are presented in the last two rows. The zero CR values emphasizes with bold face illustrate that the logarithmic strategy with 59 sub-populations offers the zero CR for 8 functions, the sequential and jumping strategies with 30 sub-populations offers the zero CR for 4 other functions and the zero CR for the last one which is function f_{2M} is offered by the customized logarithmic strategy with 35 local CAs.

It should be mentioned here that CR value for each experiment is calculated based on the runs where an optimal solution is found. For instance, the sequential strategy with 30 local CAs can find an optimal solution in only 9 independent runs out of 100, meaning that its CR value is calculated only based on those 9 successful runs. Consequently, for the experiments where no optimal solution is found there is no CR value, such as the jumping strategy with 10 local CAs applied on function f_2.

Table 6.6 illustrates that the sequential strategy works nearly equivalently to the jumping strategy with the exception of function f_2, where the difference value of the adjacent dimensions is a key point. Therefore, for this function the strategies assigning adjacent dimensions to the same local CA work better. In order to prove this claim, function f_{2M} is proposed and the results show that if this key point is removed the sequential strategy works the same as the jumping one. The results also show that the logarithmic strategy works well only with higher number of sub-populations.

Table 6.6: The CR and SR values with respect to each experiments for balanced strategies in percentile form alongside their average in the last two rows.

Style		Balanced Strategies									
		Seq				Jump				Seq/Jump	Over Seq
Local CAs		5	7	10	15	5	7	10	15	30	15
f_1	CR	25.08	18.48	8.62	4.25	26.30	18.13	8.60	4.29	0.50	9.29
	SR	99	100	100	100	100	100	100	100	100	100
f_2	CR	24.23	11.90	11.17	24.59	-	-	-	100.00	99.55	7.85
	SR	95	100	100	100	0	0	0	6	9	100
f_{2M}	CR	-	-	-	-	-	-	-	-	-	-
	SR	0	0	0	0	0	0	0	0	0	0
f_3	CR	26.98	16.84	7.82	3.62	27.61	16.41	7.89	3.63	0.20	8.75
	SR	100	100	100	100	100	100	100	100	100	100
f_4	CR	25.02	15.36	5.89	2.34	32.12	16.22	5.98	2.35	**0.00**	7.01
	SR	100	99	100	100	100	100	100	100	100	100
f_5	CR	32.83	20.98	10.13	5.20	33.41	19.98	10.41	5.13	0.61	11.26
	SR	100	100	100	100	100	100	100	100	100	100
f_6	CR	31.96	18.03	7.16	2.98	24.36	15.19	6.67	2.91	**0.00**	7.12
	SR	100	100	100	100	100	100	100	100	100	97
f_7	CR	27.65	17.49	8.63	4.36	25.88	18.08	8.47	4.35	0.48	9.40
	SR	100	100	100	100	100	100	100	100	100	100
f_8	CR	31.81	19.73	9.78	5.05	32.23	21.14	9.88	4.97	0.59	10.91
	SR	100	100	100	100	100	100	100	100	100	100
f_9	CR	35.08	24.62	10.12	4.25	32.08	20.64	8.99	4.29	**0.00**	10.22
	SR	99	100	100	100	100	100	100	100	100	100
f_{10}	CR	33.17	20.85	5.92	2.43	31.70	20.52	5.95	2.41	2.52	9.03
	SR	93	97	99	100	97	99	98	100	100	100
f_{11}	CR	38.11	24.82	13.27	6.66	41.96	25.83	13.25	6.73	0.74	14.72
	SR	100	100	100	100	99	100	100	100	100	100
f_{12}	CR	12.98	6.53	3.15	1.29	12.49	7.50	3.21	1.27	**0.00**	3.86
	SR	100	100	100	100	100	100	100	100	100	100
Average	CR	28.74	17.97	8.47	5.58	29.10	18.15	8.12	11.86	8.77	9.12
	SR	91.2	92.0	92.2	92.3	84.3	84.5	84.5	85.1	85.3	92.1

Generally, the only configurations that offer the SR of 100% for all 13 optimization functions include: (1) the customized logarithmic strategy with 35 local CAs, (2) the logarithmic strategy with 59 local CAs, and (3) the logarithmic strategy with 30 local CAs. Within these three configurations, the first two offer average CR values less than 1.00% which is much lower than 4.27% of the third one. This is due to the fact that the third one does not include the assignment of each dimension to one local CA.

Table 6.7: The CR and SR values with respect to each experiments for imbalanced strategies in percentile form alongside their average in the last two rows.

Style		Imbalanced Strategies						
		Logarithmic						Cust
Local CAs		5	7	10	15	30	59	35
f_1	CR	100.00	49.14	42.40	11.69	3.61	**0.00**	0.42
	SR	38	95	91	100	100	100	100
f_2	CR	52.78	31.43	22.43	7.70	1.90	**0.00**	1.20
	SR	33	72	73	100	100	100	100
f_{2M}	CR	90.18	84.22	100.00	59.62	18.52	7.43	**0.00**
	SR	33	52	48	100	100	100	100
f_3	CR	100.00	54.49	44.31	11.31	3.33	**0.00**	0.36
	SR	20	100	94	100	100	100	100
f_4	CR	100.00	48.93	48.94	9.51	1.84	0.54	0.15
	SR	49	90	46	99	100	100	100
f_5	CR	100.00	61.21	53.42	14.79	4.24	**0.00**	0.58
	SR	33	96	89	100	100	100	100
f_6	CR	100.00	42.18	38.73	9.21	2.28	0.06	0.29
	SR	54	98	85	100	100	100	100
f_7	CR	100.00	46.73	49.34	11.83	3.54	**0.00**	0.45
	SR	31	94	93	100	100	100	100
f_8	CR	100.00	52.55	53.56	13.63	4.16	**0.00**	0.56
	SR	21	92	86	100	100	100	100
f_9	CR	100.00	51.90	51.95	13.09	3.28	0.03	0.38
	SR	39	92	88	100	100	100	100
f_{10}	CR	100.00	53.89	53.92	10.12	1.68	**0.00**	2.15
	SR	2	71	68	95	100	100	100
f_{11}	CR	100.00	60.93	59.53	19.06	5.75	**0.00**	0.73
	SR	16	94	85	100	100	100	100
f_{12}	CR	100.00	19.33	23.25	5.15	1.35	0.41	0.10
	SR	69	100	100	100	100	100	100
Average	CR	95.61	50.53	49.37	15.13	4.27	0.65	0.57
	SR	33.7	88.2	80.5	99.5	100.0	100.0	100.0

In fact assigning only one dimension to a local CA helps to find an optimal solution when the dimensions of the problem are independent to each other, and assigning a number of dimensions to a local CA is more useful where the dimensions are inter-dependent. Consequently, incorporating a hybrid mechanism works better when there is no prior knowledge about the given optimization function. Therefore, generally the imbalanced dimension division techniques outperform the balanced strategies.

Table 6.8: The average SR and the number of experiments with the CR of 100% obtained for each optimization function.

Functions	f_1	f_2	f_{2M}	f_3	f_4	f_5	f_6
Average SR	95.5%	64.0%	31.4%	94.9%	93.1%	95.2%	96.1%
Reliable Experiments	13	8	4	15	12	14	13
Functions	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	
Average SR	95.2%	94.1%	95.2%	89.4%	93.8%	98.2%	
Reliable Experiments	14	14	13	7	13	16	

The overall results are represented in Table 6.8 illustrating the average SR obtained by all the configurations for each optimization function and the number of reliable experiments capable to obtain the SR of 100%. This information determines the most challenging optimization functions which are f_{2M}, f_{10}, and f_2. For the rest of the functions, more than 12 experiments out of 17 are able to find the optimal solutions in all of their 100 independent runs.

6.3.7 Conclusions

HMP-CA is a new architecture for MP-CA which incorporates heterogeneous local CAs with a shared belief space. In HMP-CA, local CAs are working to optimize different subsets of dimensions. Therefore, the mechanism that divides the dimensions among local CAs plays a key role in the performance of the method. In this article, a study is conducted to analyze different dimension division strategies.

The evaluated strategies are classified into two categories; namely balanced and imbalanced. Within the former category, sequential, jumping, and overlapped sequential strategies are taken into account in addition to the logarithmic and customized logarithmic strategies from the latter category.

The results show that generally the imbalanced techniques offer better performance compared to the balanced schemes. More specifically, the logarithmic strategy

with 59 local CAs and the customized logarithmic technique with 35 local CAs present the best success rates as well as the minimum convergence ratios.

Future work may include exploring additional dimension division schemes to discover even better success rates and convergence ratios. The proposed algorithms may also be tested in the future against other optimization functions, particularly on functions with inter-dependent variables.

Acknowledgment

This work is made possible by a grant from the National Science Foundation and NSERC Discovery No. 327482.

6.4 Heterogeneous Multi-Population Cultural Algorithm with a Dynamic Dimension Decomposition Strategy

Abstract. Heterogeneous Multi-Population Cultural Algorithm (HMP-CA) is one of the most recent architecture proposed to implement Multi-Population Cultural Algorithms which incorporates a number of heterogeneous local Cultural Algorithms (CAs) communicating with each other through a shared belief space. The heterogeneous local CAs are designed to optimize different subsets of the dimensions of a given problem. In this article, two dynamic dimension decomposition techniques are proposed including the top-down and bottom-up approaches. These dynamic approaches are evaluated using a number of well-known benchmark numerical optimization functions and compared with the most effective and efficient static dimension decomposition methods. The comparison results reveals that the proposed dynamic approaches are fully effective and outperforms the static approaches in terms of efficiency.

6.4.1 Introduction

Cultural Algorithm (CA) developed by Reynolds [175] is an Evolutionary Algorithm incorporating knowledge to improve its search mechanism. CA extracts knowledge during the process of evolution and incorporates the extracted knowledge to guide the search direction. Although successful applications of CAs have been reported in various research areas, they are still suffering from immature convergence. One of the most common strategy to overcome this limitation is to incorporate multiple populations. In Multi-Population Cultural Algorithm (MP-CA), the population is

divided into a number of sub-populations and each sub-population is directed by a local CA. The local CAs communicate with each other by migrating their extracted knowledge [170].

Different architectures have been proposed to implement MP-CAs. Each architecture has its own strengthes and weaknesses. Homogeneous local CAs is one of the common architectures in which the local CAs have their own local belief spaces and works exactly the same [211, 76, 168]. Heterogeneous local CAs is another architecture for MP-CAs in which the local CAs are designed to optimize one of the problem dimensions. This architecture is introduced by Lin *et al.* [118] in their proposed Cultural Cooperative Particle Swarm Optimization (CCPSO) to train a Neurofuzzy Inference System.

The most recent architecture is Heterogeneous Multi-Population Cultural Algorithm (HMP-CA) which incorporates only one belief space shared among local CAs instead of one local belief space for each local CA [171]. The shared belief space is responsible to keep a track of the best parameters found for each dimension. In this architecture the local CAs are designed to optimize different subsets of problem dimensions.

There are various approaches to divide dimensions among local CAs such as assigning one dimension to a local CA [118] and a jumping strategy with various number of local CAs [171]. Recently we investigated various static dimension decomposition approaches including the balanced and imbalanced techniques [165]. The techniques assigning the same number of dimensions to the local CAs are called balanced techniques while the imbalanced ones assign different number of dimensions to the local CAs. As imbalanced techniques, logarithmic and customized logarithmic approaches were investigated with various number of local CAs. The results of this investigation reveals that the imbalanced techniques outperform the balanced approaches by offering 100% success rate and very low convergence ratio.

Although various static strategies have been investigated, it may would be worthwhile to evaluate dynamic strategies. The goal of this article is to design the dynamic dimension decomposition methods.

6.4.2 Proposed Method

In this article, two new dynamic dimension decomposition approaches are introduced to improve the efficiency of HMP-CA. The framework of the proposed method is similar to the one incorporated to investigate various static dimension decomposition techniques [165]. Like the published HMP-CA, in the proposed algorithm the local CAs communicate with each other by sending their best partial solutions to the shared belief space and use its updated knowledge to evaluation their partial solutions.

In the published HMP-CA, various static dimension decomposition techniques with various number of local CAs are considered such that the whole population is initially divided among the local CAs evenly. In contrast, our main contribution in this article is to incorporate two dynamic dimension decomposition approaches. Therefore, instead of considering a huge population and dividing it into sub-populations, each local CA gets assigned with a small sub-population of a constant size referred by $PopSize$ parameter. Another parameter is also required as a threshold for the number of generations a local CA cannot find a better solution which is denoted by $NoImpT$ parameter.

In order to design a dynamic dimension decomposition technique, there are two approaches including:

- *Top-Down strategy*: In this strategy, the idea is to start with a local CA designed to optimize all the problem dimensions and recursively split the dimensions between two newly generated local CAs if a local CA cannot find a better solution for a number of generations. It should be noted here that the decomposed local CA cooperates with the two new local CAs for the next generations such that

the decomposed local CA will not be split again. Local CAs with only one assigned dimension will not be decomposed as well.

- *Bottom-Up strategy*: The idea of this approach is to merge the dimensions of two local CAs when they reach to the no improvement threshold. This approach starts with a number of local CAs, each of which is designed to optimized only one dimension. The number of initially generated local CAs equals to the number of problem dimensions. These local CAs start to optimize their assigned dimensions until two of them reach to the no improvement threshold. In this stage, a new local CA is generated to optimize all the dimensions of those two local CAs together. Like the top-down approach, each local CA is merged only one time. Therefore, a local CA with all the problem dimensions never get merged.

As the results of the investigation of static dimension decomposition techniques show [165], the imbalanced strategies which assign one dimension to a local CA offer the best efficiency and effectiveness. Therefore, it is expected to see better results from the second approach where the local CAs are initially set to optimize only one dimension compared to the first approach in which it takes a high number of generations to generate the local CAs with only one assigned dimension.

In order to have a fair comparison with static dimension decomposition techniques, the rest of the proposed method is exactly the same as the published HMP-CA [165] such that each local CA incorporates a simple DE with $DE/rand/1$ mutation and binomial crossover operators, and communicates with the shared belief space by sending its best partial solutions. It should be noted here that the belief space contributes to the search process only by providing the complement parameters to evaluate partial solutions.

Table 6.9: Well-known numerical optimization benchmarks.

ID	Function Name	Parameter Range	Minimum Value
f_1	Sphere Model	[-100,100]	0
f_2	Generalized Rosenbrock's Function	[-30,30]	0
f_3	Generalized Schwefel's Problem 2.26	[-500,500]	-12569.5
f_4	Generalized Rastrigin's Function	[-5.12,5.12]	0
f_5	Ackley's Function	[-32,32]	0
f_6	Generalized Griewank's Function	[-600,600]	0
f_7	Generalized Penalized Function 1	[-50,50]	0
f_8	Generalized Penalized Function 2	[-50,50]	0
f_9	Schwefel's Problem 1.2	[-500,500]	0
f_{10}	Schwefel's Problem 2.21	[-500,500]	0
f_{11}	Schwefel's Problem 2.22	[-500,500]	0
f_{12}	Step Function	[-100,100]	0

6.4.3 Experiments and Results

Numerical optimization problems are considered to evaluate the proposed dynamic techniques. The goal of these problems is to find the optimal value of a given function. The numerical optimization function is a black box which gets a D-dimensional vector of continuous parameters as input and returns a real number as the objective value for the given parameters. Researchers in this area usually incorporate the set of well-known numerical optimization benchmarks represented in Table 6.9 [218, 181, 133]. This table lists 12 benchmark problems with their corresponding parameter ranges and optimal values. These functions are more detailed in the aforementioned papers. In addition to these 12 functions, modified Rosenbrock's function (f_{2M}) [165] is also considered.

The proposed dynamic strategies are evaluated with two different values for each of $PopSize$ and $NoImpT$ parameters including 10 and 20 for the former parameter, and 5 and 10 for the latter one. The other parameters are set the same as the ones incorporated to investigate the static dimension decomposition techniques [165]. The number of dimensions is set to 30 which is the common number in the area of numerical optimization and the upper limit for fitness evaluations is considered

Table 6.10: The average number of FE with respect to each experiments.

Style	Top-Down				Bottom-Up			
$PopSize$ ($NoImpT$)	10(5)	10(10)	20(5)	20(10)	10(5)	10(10)	20(5)	20(10)
f_1	123K	120K	235K	215K	**70K**	71K	97K	93K
f_2	391K	**370K**	729K	499K	379K	544K	663K	1,065K
f_{2M}	**986K**	987K	1,586K	1,437K	1,256K	1,192K	1,815K	1,716K
f_3	128K	139K	216K	248K	57K	64K	55K	**46K**
f_4	253K	271K	355K	375K	172K	**149K**	157K	158K
f_5	103K	98K	202K	167K	62K	**61K**	92K	88K
f_6	142K	145K	270K	249K	**97K**	**97K**	116K	110K
f_7	123K	128K	221K	219K	**65K**	67K	96K	91K
f_8	118K	116K	219K	212K	**64K**	66K	93K	89K
f_9	169K	156K	268K	273K	106K	**101K**	123K	103K
f_{10}	197K	196K	357K	353K	**123K**	124K	227K	226K
f_{11}	104K	109K	173K	192K	**55K**	56K	83K	81K
f_{12}	69K	68K	131K	123K	17K	22K	**12K**	13K
Average	224K	223K	382K	351K	**194K**	201K	279K	298K

as 10,000,000. Each experiment is conducted by 100 independent runs to provide statistically significant sample size.

In order to evaluate the effectiveness of the proposed approaches, *Success Rate (SR)* is incorporated which refers to the percentage of the independent runs where an optimal solution is reached. A higher SR means more reliability of the method to find optimal solutions. Therefore, a method with the 100% SR is the most reliable method being able to find an optimal solution within each run. For the efficiency evaluation, the number of *Fitness Evaluations (FE)* is considered in contrast to the number of generations incorporated to evaluate static dimension decomposition approaches [165]. This is due to the fact that dynamic approaches use different number of solutions within each generations.

The experimental results represent that both proposed approaches with all of the four configurations are able to find the optimal solution with respect to each optimization function in every independent run. Therefore, the SR for all the experiments is 100% which implies that both proposed techniques are fully effective methods. The

average number of fitness evaluations used to find the optimal solution are illustrated in Table 6.10 such that the minimum values with respect to each optimization function is emphasized with bold face. The letter K in the table means that the numbers are in thousands and the last two rows represent the average number of fitness evaluation over all the 13 optimization functions.

In order to evaluate the efficiency of the proposed techniques *Convergence Ratio (CR)* measure [165] is incorporate:

$$ConvergenceRatio = \frac{AVG - AVG_{min}}{AVG_{max} - AVG_{min}} \times 100\%$$

where AVG denotes the average number of fitness evaluation required by a specific algorithm to find an optimal solution, and AVG_{min} and AVG_{max} refer to the minimum and the maximum averages obtained by all methods for each optimization function. Therefore, the lower CR value represents the more efficient algorithm.

In order to compare the proposed dynamic approaches with the static techniques, the most efficient ones with 100% success ratio are selected which includes the logarithmic approach with 30 and 60 local CAs and the customized logarithmic approach with 35 local CAs [165]. The CR values are calculated with respect to each optimization function and then they are averaged over all the 13 optimization functions. Therefore, each algorithm gets assigned with a CR value.

Table 6.11 illustrates the average CR values for both static and proposed dynamic dimension decomposition techniques. The customized logarithmic approach is denoted by Cust in the table. The numbers on the third row specify the configuration of each technique which is the number of local CAs for each static technique and the values for $PopSize$ and $NoImpT$ parameters for each dynamic method with the format of $PopSize(NoImpT)$. The lowest average CR is represented in bold face.

Table 6.11: The average CR values to compare dimension decomposition techniques.

Static Approaches			Dynamic Approaches							
Logarithmic		Cust	Top-Down				Bottom-Up			
30	60	35	10(5)	10(10)	20(5)	20(10)	10(5)	10(10)	20(5)	20(10)
75.71%	36.30%	29.01%	29.85%	30.06%	84.79%	79.94%	**3.37%**	4.61%	20.09%	21.70%

As expected, the table shows that the best efficiency is offered by the dynamic bottom-up approach with various parameter values. However the best configuration for this strategy is $PopSize$ of 10 and $NoImpT$ of 5.

6.4.4 Conclusions

HMP-CA is the most recent architecture proposed to implement MP-CA, in which local CAs are designed to optimized a subset of the problem dimensions. Therefore, the incorporated dimension decomposition approach has a major effect on the algorithm performance. In this article, two dynamic strategies are proposed including the top-down and bottom-up approaches.

The results of extensive experiments reveals that both dynamic approaches are fully effective to find optimal solutions. In addition to the effectiveness, the results illustrate that the bottom-up approach outperforms the top-down approach by offering the better efficiency. Compared to the most effective and efficient static dimension decomposition methods, the bottom-up approach offers better efficiency while the top-down technique presents competitive efficiency.

As incorporating extra knowledge to choose dimensions for merging or splitting may results in a more efficient method, it has been considered as the future direction for this research area. Furthermore, incorporating more sophisticated optimization functions and higher number of dimensions would be useful to highly evaluate the proposed methods and deeply differentiate them based on their performance.

Acknowledgment

This work is made possible by a grant from the National Science Foundation and NSERC Discovery No. 327482.

6.5 Adaptive Heterogenous Multi-Population Cultural Algorithm for Large Scale Global Optimization

Abstract. Detecting interacting variables has a major effect on the performance of an optimization algorithm especially in large scale global optimization. In this article, a technique based on additively interdependency is introduced which is called Variable Additively Interdependence Learning (VAIL). VAIL uses a very simple mathematical equation to evaluate the additively interdependencies. The proposed method incorporating VAIL is called Adaptive Heterogeneous Multi-Population Cultural Algorithm (A-HMP-CA) which consists of a number of local Cultural Algorithms (CAs) and a shared belief space. The proposed method which has the ability to generate new local CAs during the evolutionary process, incorporates VAIL to detect additively interactive variables, merges the interactive variables into one group and assigns the group to a new local CA to improve the optimization. The A-HMP-CA is evaluated over large scale global optimization problems and compared with the state-of-the-art methods. The comparison which is statistically analyzed reveals that the A-HMP-CA significantly outperforms the existing methods by offering better results.

6.5.1 Introduction

Optimization problems are a set of problems where the goal is to make a system as effective as possible by adjusting its input parameters. The optimization problems are categorized based on the type of their input parameters. The problems with

continuous parameters are called global optimization problems, while the others with discrete parameters are called combinatorial optimization problems.

The research area of optimization is very well-known due its wide range of applications. Numerical optimization problems are a subset of optimization problems where the goal is to find the optimal value of a given function. The numerical optimization function is considered as a black box which gets a D-dimensional vector of continuous parameters as input referred by a solution and returns a real number as the objective value for the given solution. The problems where D is a large number are called large scale global optimization problems.

Various kinds of algorithms are proposed to deal with optimization problems. As reported in the literature, working with a population of solutions instead of only one solution usually increases the performance of an optimization algorithm. Evolutionary Algorithms (EAs) is a subset of population-based methods which are successfully applied to solve optimization problems. Although there are different types of EAs introduced in the literature, their common strategy is the evolutionary process inspired by the natural selection which states that the fitter individuals to the environment are more likely the ones surviving for the next generations.

The most popular EA is Genetic Algorithm (GA) which is proposed by Holland [93] and improved by various researchers consequently. The most recently introduced evolutionary approach in the literature is Differential Evolution (DE) designed by Storn and Price [191] to solve global optimization problems. DE is well-known due to its efficient search space exploration which makes it the best candidate to be joined with a local search heuristic to provide an effective algorithm.

Cultural Algorithm (CA) is also an EA which incorporates knowledge to improve its search mechanism. CA which is developed by Reynolds [175] is designed to extract knowledge during the evolutionary process and to incorporate the extracted knowledge to guide the search direction. CA works in two different spaces, namely

population space and belief space. The former one is where the individuals are getting evolved which is similar to the solution population of other types of EAs. The latter one is designed to extract, update and record knowledge.

Although a wide range of successful applications of CAs has been reported in the literature, they are still suffering from their immature convergence. This is mainly due to the fact that they are not capable to preserve the population diversity over generations. Various strategies have been proposed in order to help CAs to overcome this limitation such as rejecting duplicate solutions, incorporating a high mutation rate, and removing solutions with the same objective value. One of the most common strategy is to incorporate multiple populations with lower size instead of a big single population. In Multi-Population Cultural Algorithm (MP-CA), the single population is divided into a number of sub-populations and each sub-population is directed by a local CA. Digalakis and Margaritis [51] introduced the first MP-CA which was designed to schedule electrical generators.

Within MP-CAs, the local CAs communicate with each other over generations by exchanging their extracted knowledge. The local CAs can be homogeneous such that they work exactly the same, or they can be heterogeneous such that they cooperate with each other by working on different parts of the given problem. In order to incorporate heterogeneous local CAs, first the given problem should be decomposed into a number of sub-problems. For the numerical optimization problems, the problem dimensions are divided into a number of groups. There are various decomposition techniques proposed in the literature, each of which has its own strengthes and weaknesses.

In this article, a technique is proposed to detect the dimension interdependencies in order to provide an adaptive algorithm for large scale global optimization. The proposed technique detects the interactions over the evolutionary process and merges the interacting dimensions into one group to speed up their optimization. The re-

maining of this article is structured as follows. Subsection 6.5.2 briefly describes the large scale global optimization, followed by reviewing the MP-CA architectures in Subsection 6.5.3. Subsection 6.5.4 represents the existing dimension decomposition techniques. The proposed method is represented in Subsection 6.5.5, followed by illustrating the experiments and results in Subsection 6.5.6. Finally the last subsection represents concluding remarks and future directions.

6.5.2 Large Scale Global Optimization

Numerical optimization is a class of optimization problems where the goal is to find the optimal objective value of a mathematical optimization function with a set of continuous input parameters. A numerical optimization problem is mathematically defined in Equation 6.21:

$$\begin{aligned} Minimize \quad & f(X) \\ & X = \{x_1, x_2, ..., x_D\} \\ & f : \mathbf{R}^D \rightarrow \mathbf{R} \\ Subject\ to \quad & g_i(X) = 0, i = 1, ..., m \\ & h_j(X) \leq 0, j = 1, ..., n \end{aligned} \tag{6.21}$$

where $f(X)$ denotes to the optimization function where its input is a D-dimensional vector of continuous parameters denoted by X including D parameters denoted by x_i where i is the dimension index. Each optimization function is accompanied with a number of equality and inequality constraints denoted by $g_i(X)$ and $h_j(X)$, respectively.

The optimization problems with a large number of dimensions are called large scale global optimizations. The common number of dimensions in this area is 1000.

The numerical optimization functions are usually categorized into three classes based on the interdependency of their input variables. These classes include fully

separable functions, partially separable functions and non-separable functions. A fully separable function is defined in Definition 6.5.1 [193].

Definition 6.5.1. A function f(x) is fully separable if and only if

$$\arg \min_{(x_1,...,x_n)} f(x_1, ..., x_n) = \left(\arg \min_{x_1} f(x_1, ...), ..., \arg \min_{x_n} f(..., x_n) \right) \tag{6.22}$$

The fully separable functions are the functions that can be optimized by optimizing their dimensions separately since there is no interdependencies between their dimensions. These interdependencies are also known as variable interactions. The partially separable functions and the non-separable functions are also defined in Definition 6.5.2.

Definition 6.5.2. The optimization functions where the interacting variables can be grouped into a number of independent subsets are called partially separable functions, and the functions where there is only one independent group including all the dimensions are called non-separable functions.

The benchmark functions for the CEC'2010 competition on large scale global optimization [193] is one of the well-known benchmarks in the literature. This benchmark includes twenty 1000-dimensional numerical optimization functions which can be categorized into the three following classes based on Definitions 6.5.1 and 6.5.2:

1. Shifted Fully Separable Functions ($f_1 - f_3$).
2. Shifted Partially Separable Functions ($f_4 - f_{18}$).
3. Shifted Nonseparable Functions ($f_{19} - f_{20}$).

6.5.3 Multi-Population Cultural Algorithm

As mentioned before, MP-CAs are the improved version of the traditional CAs. In order to implement MP-CAs, there are different architectures proposed in the liter-

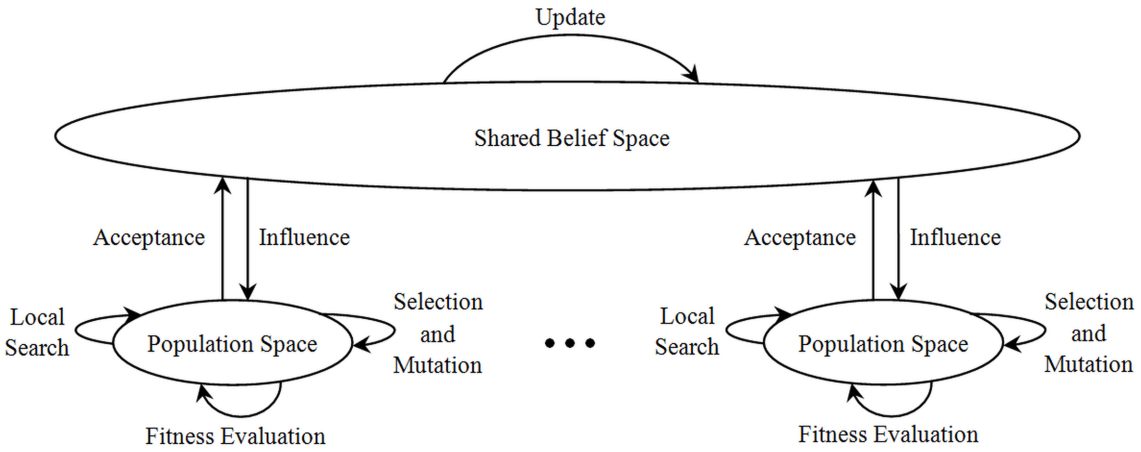


Figure 6.6: HMP-CA Architecture [171]

ature. The most common architecture is homogeneous local CAs [211, 76, 168] in which the local CAs have the same goal. Although in this architecture the local CAs may use different parameters or even different evolutionary approaches, they have the same goal which is to solve the whole given problem.

The architectures in which the local CAs have different goals belong to heterogeneous local CAs category. Cultural Cooperative Particle Swarm Optimization (CCPSO) proposed by Lin *et al.* [120] is one of the architectures in this category. In CCPSO, the local CAs are designed to optimize only one of the problem dimensions separately in parallel. Each local CA incorporates Particle Swarm Optimization (PSO) to evolve its population while it has its own local belief space to record the extracted knowledge.

The most recent architecture is Heterogenous Multi-Population Cultural Algorithm (HMP-CA) [171] which is illustrated in Figure 6.6. As presented in the figure, instead of a local belief space for each local CA, HMP-CA incorporates only one belief space shared among local CAs which is responsible to keep a track of the best parameters found for each dimension.

HMP-CA has the flexibility to design local CAs to optimize any subset of the problem dimensions. Additionally, HMP-CA [171] incorporates the concept of par-

tial solution. A partial solution is a solution including the parameters for specific dimensions. In HMP-CA each sub-population is designed to carry only the partial solutions with respect to its assigned dimensions instead of complete solutions. Since the total number of solutions within each generation is constant, the number of parameters for each generation is dramatically decreased which makes the HMP-CA an efficient method in terms of both CPU time and memory usage.

Although dealing with partial solutions utilized the CPU time and memory usage, since fitness evaluation is conducted only on complete solutions, a partial solution has to be completed with its complement parameters before being evaluated. Therefore, incorporating partial solution does not affect the efforts needed for fitness evaluations. It should be noted here that the only influence of the shared belief space is its participation in fitness evaluations of partial solutions.

Dynamic HMP-CA (D-HMP-CA) [174] is recently introduced which is the HMP-CA improved by incorporating dynamic decomposition techniques. The dynamic version is able to generate new local CAs with different dimension subsets during the evolutionary process. Additionally, it is able to terminate the previously generated local CAs.

6.5.4 Dimension Decomposition Techniques

In order to decompose dimensions between local CAs, there are various approaches proposed in the literature which can be divided into two categories including static approaches and dynamic ones. In static decomposition techniques, the dimensions are initially divided among local CAs such that this division will not be altered during the evolutionary process, while in dynamic approaches the dimension groups or even the number of local CAs will be changed.

The static decomposition techniques usually work with specific number of local CAs. The CCPSO [120], for instance, assigns each dimension to a local CA meaning

that the number of local CAs in this approach equals to the number of problem dimensions. Unlike the CCPSO, HMP-CA [171] incorporates a jumping strategy with four different number of local CAs including 5, 10, 15, and 30.

Various static dimension decomposition approaches with different number of local CAs are recently investigated [165]. This investigation classifies the techniques into two groups of balanced and imbalanced approaches. The techniques assigning the same number of dimensions to different local CAs are considered as balanced techniques and others which assign different number of dimensions to local CAs are called imbalanced approaches. In this investigation, sequential, jumping and sequential with overlap approaches as balanced techniques are compared with logarithmic and customized logarithmic approaches as imbalanced techniques. The investigation results reveal that the imbalanced techniques including local CAs with one assigned dimension outperform others in both efficiency and effectiveness.

Two dynamic dimension decomposition techniques are introduced in D-HMP-CA [174] including the top-down and bottom-up techniques. In the former approach the algorithm starts with a local CA designed to optimize all the problem dimensions and if it cannot find a better solution after a specific number of generations two new local CAs will be generated to optimize half of its dimensions. The latter approach for optimizing a D-dimensional optimization problem starts with D local CAs, each of which gets assigned with only one dimension. In this approach if two local CAs cannot find a better solution after a specific number of generations, a new local CA will be generated to optimize their dimensions together.

The dimension decomposition techniques are not limited to the area of HMP-CA. Another area where the dimension decomposition is highly important is Cooperative Coevolution (CC) [160]. CC incorporates a divide-and-conquer approach to decompose a problem into a number of smaller sub-problems such that each sub-problem is getting optimized by an EA separately in parallel. Although the framework of

CC is similar to the HMP-CA, the HMP-CA benefits from its belief space which incorporates knowledge to guide the search direction.

CC is first introduced in a Genetic Algorithm which is called Cooperative Co-evolutionary Genetic Algorithm (CCGA) [160]. CCGA uses the same dimension decomposition technique as CCPSO [120] which is dividing D-dimensional optimization problem into D 1-dimensional sub-problems. Fast Evolutionary Programming with Cooperative Co-evolution(FEPCC) [124] which is the first attempt for applying CC to large scale global optimization also used the same strategy for dimension decomposition.

Another decomposition strategy is splitting-in-half [186] in which all the dimensions are decomposed into two halves and each half is optimized with a DE. The main drawback of this mechanism is that it cannot be scaled up for large scale global optimization. This is due to the fact that in high dimensional problem, the two sub-problems are also high dimensional.

One of the most popular dimension decomposition strategy in CC is random grouping which is proposed by Yang et al. [214]. Their algorithm which is called DECC-G divides a D-dimensional problem into m s-dimensional problem satisfying $m \times s = D$. This division is a dynamic process such that the groups will be changed in every cycles. DECC-G which uses a constant group size works efficient with a small group size for separable problems and with a large group size for non-separable problems. Consequently, in order to be an efficient algorithm DECC-G needs to know the best group size in advance.

DECC-G is improved by incorporating a multilevel strategy for decomposition. The improved version which is called Multilevel Cooperative Coevolution (MLCC) [215] uses a pool of different group sizes and select a group size based on the problem under investigation and the stage of the evolution. Although MLCC works better than DECC-G, determining a good pool of group sizes is also difficult.

Consequently, a dynamic decomposition technique considering the variable interactions would results in a better performance. Recently a technique for detecting the variable interactions is proposed which is called Variable Interaction Learning (VIL) [31]. This technique which is computationally expensive is incorporated within a CC framework so-called Cooperative Coevolution with Variable Interaction Learning (CCVIL).

Another systematic way to detect interacting variables is proposed by Omidvar *et al.* [148]. Their proposed method so-called DECC-DML incorporates a delta grouping technique which considers the low-improved dimensions to be interacting.

6.5.5 Adaptive Heterogeneous Multi-Population Cultural Algorithm

As determining the variable interactions is a key factor in the performance of an optimization algorithm especially in large scale global optimization, in this article a new technique to detect the additively interdependent variables is introduced. This technique which is the main contribution of this article is called Variable Additively Interdependence Learning (VAIL).

The VAIL technique is incorporated in a D-HMP-CA [174] framework which has the ability to generate new local CAs as well as to terminate the existing local CAs. The belief space of the D-HMP-CA is upgraded by incorporating the VAIL module in order to keep the independent variables in different groups and merge the interdependent variables into one group. Since with this mechanism the proposed algorithm can be adapted to the given optimization problem, it is called Adaptive Heterogeneous Multi-Population Cultural Algorithm (A-HMP-CA). Sub-subsection 6.5.5.1 further describes the VAIL technique, followed by representing the evolutionary approach in Sub-subsection 6.5.5.2. The overall framework of the proposed A-HMP-CA is described in Sub-subsection 6.5.5.3.

6.5.5.1 Variable Additively Interdependence Learning

In addition to the separability concept defined in Definition 6.5.1, there is another important concept to evaluate functions separability which is called additively separability as defined in Definition 6.5.3.

Definition 6.5.3. A function f(x) is fully additively separable if and only if

$$f(x_1, x_2, ..., x_n) = f_1(x_1) + f_2(x_2) + ... + f_n(x_n) \tag{6.23}$$

Based on this definition, a fully additively separable function consists of a number of sub-functions with respect to each dimension. Consequently, a fully additively separable function is fully separable as well, but a fully separable function is not necessarily a fully additively separable. Ackley's function (f_3 in the CEC'2010 benchmark [193]), for instance, is a fully separable function, but it is not a fully additively separable function.

Additionally, Definition 6.5.3 states that a partially additively separable function consists of a number of sub-functions with respect to different mutually exclusive subsets of dimensions. For instance, the function f illustrated in Equation 6.24 is partially additively separable.

$$f(x_1, ..., x_n) = f_1(x_1, ..., x_i) + f_2(x_{i+1}, ..., x_n) \tag{6.24}$$

In partially additively separable functions, the dimensions in the same subsets are additively dependent to each other while they are additively independent to the rest of the dimensions. Therefore if two variables are additively independent, their changes should only affect their corresponding sub-functions. In function f represented in Equation 6.24, for instance, if x_i will be changed to x_i', the results of f_2 will not be

changed. Therefore, the difference in the total objective equals to the changes in f_1.

$$f\left(x_1, ..., x_i', ..., x_n\right) - f\left(x_1, ..., x_i, ..., x_n\right) = f_1\left(x_1, ..., x_i'\right) - f_1\left(x_1, ..., x_i\right) \quad (6.25)$$

Based on Equation 6.25 if two different dimensions x_i and x_j are additively independent, the total objective changes due to the changing x_i is independent to x_j:

$$f\left(x_1, ..., x_i', ..., x_j, ..., x_n\right) - f\left(x_1, ..., x_i, ..., x_j, ..., x_n\right) = f_1\left(x_1, ..., x_i'\right) - f_1\left(x_1, ..., x_i\right) \quad (6.26)$$

$$f\left(x_1, ..., x_i', ..., x_j', ..., x_n\right) - f\left(x_1, ..., x_i, ..., x_j', ..., x_n\right) = f_1\left(x_1, ..., x_i'\right) - f_1\left(x_1, ..., x_i\right) \quad (6.27)$$

Consequently, Equations 6.26 and 6.27 results in the following fact:

$$f\left(x_1, ..., x_i', ..., x_j, ..., x_n\right) - f\left(x_1, ..., x_i, ..., x_j, ..., x_n\right) = f\left(x_1, ..., x_i', ..., x_j', ..., x_n\right) - f\left(x_1, ..., x_i, ..., x_j', ..., x_n\right) \quad (6.28)$$

Since if two different dimensions x_i and x_j are additively independent, they should satisfy Equation 6.28 with any feasible values. Therefore, failure in satisfying this equation means that dimensions x_i and x_j are not additively independent and they should be considered as additively interacting variables. Since the other dimensions in Equation 6.28 are constants, its equivalency is also dependent to the values of those dimensions. Therefore, satisfying this equation does not infer the additively independency of dimensions x_i and x_j.

Our proposed VAIL module incorporates Equation 6.28 to detect variable interactions during the process of evolution. This module which is included in the shared belief space recognizes the interacting variables and nominates them to be merged into one group, while it emphasizes to keep non-interacting variables in different groups.

6.5.5.2 Evolutionary Approach

Like the main HMP-CA [171], the proposed A-HMP-CA incorporates a simple DE with the following components for each local CA:

- $DE/rand/1$ mutation operator represented in Equation 6.29 with a scale factor selected randomly from intervals $[0.5, 2.5]$ for each generation.

$$V_{i,g} = X_{r1,g} + F \times (X_{r2,g} - X_{r3,g}) \tag{6.29}$$

  where $X_{r1,g}$, $X_{r2,g}$, and $X_{r3,g}$ are three randomly selected target vectors in generation g. F is a scale factor determining the amount of perturbation of the base vector $X_{r1,g}$.

- Binomial crossover operator illustrated in Equation 6.30 with crossover probability 0.5 for all generations.

$$z_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } r_j \leq Cr \text{ or } j = j_{rand} \\ x_{j,i,g} & \text{otherwise} \end{cases} \tag{6.30}$$

  where Cr is the crossover probability and r_j is a randomly selected value within the interval $[0, 1)$ for the j^{th} dimension. j_{rand} is a randomly selected index to ensure that the trial vector $Z_{i,g}$ differs from target vector $X_{i,g}$ at least in one dimension.

- Selection mechanism depicted in Equation 6.31.

$$X_{i,g+1} = \begin{cases} Z_{i,g} & \text{if } f(Z_{i,g}) \leq f(X_{i,g}) \\ X_{i,g} & \text{otherwise} \end{cases} \tag{6.31}$$

where $X_{i,g}$ and $Z_{i,g}$ denote a target vector and its corresponding trial vector in the current generation g, respectively, and $X_{i,g+1}$ represents the selected target vector for the next generation.

6.5.5.3 A-HMP-CA Framework

For a D-dimensional optimization problam, the proposed A-HMP-CA starts with D local CAs each of which gets assigned with only one dimension. In every generation, the local CAs gets their complement parameters from the shared belief space to evaluate their partial solutions. Then they sends back their best partial solutions to update the belief space. The belief space reviews the coming best partial solutions and updates its records of best solutions with respect to each dimension. This routine continues until the shared belief space detects at least two non-improving local CAs.

The non-improving local CAs are the ones which are not able to find a better solution after a specific number of generations. When the shared belief space detects two non-improving local CAs, it sends their dimension subsets to the VAIL module for interdependency evaluation. If VAIL nominates them to be merged, the belief space merges their dimension subsets and generates a new local CA assigned with the merged dimension subset. Otherwise, the local CAs will not be touched. It should be noted here that each dimension subset will be merged only once. Therefore, merging ends up in a dimension subset including all the dimensions.

6.5.6 Experiments and Results

The CEC'2010 benchmark functions for large scale global optimization [193] are considered in order to evaluate the proposed A-HMP-CA. Although there are more recent benchmark sets in the literature, due to their usage in evaluating related methods the CEC'2010 benchmark functions are considered to provide a fair platform for comparison. Based on the regulation of the CEC'2010 benchmark, the number of dimensions is set to 1000 for all problems, the maximum number of fitness evaluations is considered as 3.00E+6, and each experiment is conducted by 25 independent runs.

Due to the extensive experiments conducted in D-HMP-CA [174] to adjust the algorithm parameters, the population size for each local CA is set to 10 and the threshold for the number of generations with no improvement is considered as 5.

The results of applying the proposed A-HMP-CA on CEC'2010 benchmark functions [193] are represented in Table 6.12[1]. This table also illustrates the results of the state-of-the-art methods including DECC-G [214], MLCC [215], DECC-DML [148] and CCVIL [31].

Table 6.12 represents that the proposed method can find the best solution compared to the rest of the methods for 15 functions out of 20. Therefore, it gets the best rank for these functions. Conversely, the proposed method shows very weak performance on functions f_6, f_{11} and f_{16} which are the methods including at least one group of 50-rotated Ackley's function. Since A-HMP-CA can find the exact additively independent groups of variables for these functions which are 2, 11 and 20, respectively, the only reason for this failure could be the simple evolutionary approach incorporated in local CAs.

In order to statistically compare all the five algorithms over all the 20 benchmark functions, a non-parametric procedure is incorporated [64]. This procedure starts by

[1]To see the complete results including the obtained solutions with 1.20E+5 and 6.00E+5 fitness evaluations please refer to Subsection 6.5.8. These results are also accessible online at http://raeesim.myweb.cs.uwindsor.ca/ECAI2014/CompleteResults.pdf.

Table 6.12: Comparing the proposed A-HMP-CA with the state-of-the-art methods over the CEC'2010 benchmark functions [193].

Function	DECC-G	MLCC	DECC-DML	CCVIL	A-HMP-CA		
	Mean	Mean	Mean	Mean	Mean	Std Dev	Rank
f_1	2.93E-07	1.53E-27	1.93E-25	1.55E-17	**0.00E+00**	0.00E+00	1
f_2	1.31E+03	5.55E-01	2.17E+02	6.71E-09	**4.94E-09**	1.41E-08	1
f_3	1.39E+00	9.86E-13	**1.18E-13**	7.52E-11	3.51E-13	6.49E-14	2
f_4	5.00E+12	1.70E+13	3.58E+12	9.62E+12	**1.01E+12**	3.54E+11	1
f_5	2.63E+08	3.84E+08	2.99E+08	1.76E+08	**1.03E+08**	1.41E+08	1
f_6	4.96E+06	1.62E+07	7.93E+05	**2.94E+05**	5.24E+06	7.19E+05	4
f_7	1.63E+08	**6.89E+05**	1.39E+08	8.00E+08	5.16E+07	3.64E+07	2
f_8	6.44E+07	4.38E+07	3.46E+07	6.50E+07	**8.35E+06**	1.56E+07	1
f_9	3.21E+08	1.23E+08	5.92E+07	6.66E+07	**1.44E+07**	1.61E+06	1
f_{10}	1.06E+04	3.43E+03	1.25E+04	1.28E+03	**7.72E+02**	3.62E+02	1
f_{11}	2.34E+01	1.98E+02	**1.80E-13**	3.48E+00	5.65E+01	2.54E+00	4
f_{12}	8.93E+04	3.48E+04	3.80E+06	8.95E+03	**2.07E+03**	9.82E+02	1
f_{13}	5.12E+03	2.08E+03	1.14E+03	5.72E+02	**7.96E+01**	5.88E+01	1
f_{14}	8.08E+08	3.16E+08	1.89E+08	1.74E+08	**3.22E+07**	3.71E+06	1
f_{15}	1.22E+04	7.10E+03	1.54E+04	2.65E+03	**2.17E+03**	6.79E+02	1
f_{16}	7.66E+01	3.77E+02	**5.08E-02**	7.18E+00	1.15E+02	4.28E+00	4
f_{17}	2.87E+05	1.59E+05	6.54E+06	2.13E+04	**9.33E+03**	1.96E+03	1
f_{18}	2.46E+04	7.09E+03	2.47E+03	1.33E+04	**4.17E+02**	1.51E+02	1
f_{19}	1.11E+06	1.36E+06	1.59E+07	3.52E+05	**3.39E+05**	3.76E+04	1
f_{20}	4.06E+03	2.05E+03	9.91E+02	1.11E+03	**8.19E+02**	1.76E+02	1
Avg. Rank	4.10	3.60	3.00	2.75	1.55		

applying Friedman's ranking test followed by Bonferroni-Dunns test to evaluate the significance level of the efficiency differences between the 5 algorithm. The Friedman's ranking test results in the Friedman's statistic value of 29.88, the p-value for which in a chi-squared distribution is less than 0.0001. It means that there are significant differences between the compared algorithms.

For the second part of the statistical analysis, the algorithm with the minimum average rank which is our proposed A-HMP-CA is selected as the control algorithm. Bonferroni-Dunns test calculates the critical differences (CD) for the control algorithm based on the significance level. In this stage, the two most common significance levels in the literature is incorporated which are $\alpha = 0.05$ and $\alpha = 0.10$. The results of the

Bonferroni-Dunns test are as follows:

$$CD = \begin{cases} 1.2490 & \text{for } \alpha = 0.05 \\ 1.1205 & \text{for } \alpha = 0.10 \end{cases}$$

The next step is to calculate the threshold ranks with respect to both significance levels which are the summation of the CDs and the average rank of the control algorithm. The calculated threshold ranks are as follows:

$$Threshold\ Rank = \begin{cases} 2.7990 & \text{for } \alpha = 0.05 \\ 2.6705 & \text{for } \alpha = 0.10 \end{cases}$$

These threshold ranks are used as a threshold to determine the algorithms which are significantly outperformed by the control algorithm with respect to each significance level. In other words, this statistical procedure states that the control algorithm significantly outperforms the algorithms with average rank higher than the threshold ranks. This concept is illustrated in Figure 6.7 graphically. The solid line and the dashed line in the chart represent the threshold ranks for the significance levels $\alpha = 0.05$ and $\alpha = 0.10$, respectively. This figure shows that the control algorithm significantly outperforms the algorithms whose bar exceeds the threshold lines.

Therefore based on the Friedman's ranking and Bonferroni-Dunn's tests, it can be stated that the proposed A-HMP-CA outperforms DECC-G, MLCC and DECC-DML with the significance level $\alpha = 0.05$ and it also outperforms CCVIL with the significance level $\alpha = 0.10$.

6.5.7 Conclusions

In this article, a new technique is proposed to detect the variable additively interaction. The proposed technique which is called Variable Additively Interdependence

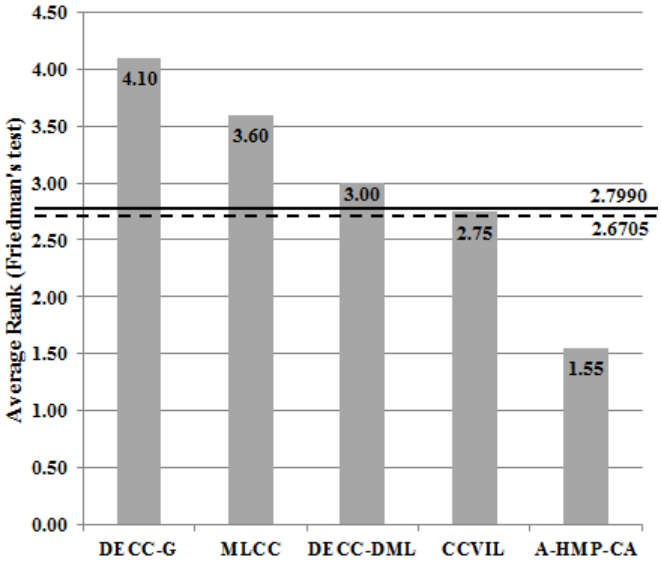


Figure 6.7: The Graphical Representation of Statistical Analysis with Friedman's test and Bonferroni-Dunn's method

Learning (VAIL) incorporates a very simple mathematical equation to evaluated the interactions. The proposed algorithm incorporating the VAIL module is called Adaptive HMP-CA which is able to optimize the interacting dimensions within the same local CA while optimizing the independent dimensions within different local CAs.

The proposed A-HMP-CA is evaluated over large scale global optimization benchmark functions. Although the results show that the proposed method is not effective in only one type of optimization functions, they represents the great performance of the proposed method on the rest of optimization functions. The proposed method is also compared to the state-of-the-art methods. The comparison is statistically analyzed which reveals that the proposed algorithm significantly outperforms the state-of-the-art methods.

Calculating the computational cost of the proposed VAIL technique is an important issue which is considered as the future direction for this research. Another direction would be incorporating a more efficient evolutionary approach for local CAs to overcome the algorithm failure in optimizing Ackley's and similar functions.

6.5.8 Complete Results

The complete results of applying Adaptive Heterogenous Multi-Population Cultural Algorithm (A-HMP-CA) on the CEC'2010 benchmark functions for large scale global optimization [193] is represented in Table 6.13.

Table 6.13: The results of applying A-HMP-CA on CEC'2010 benchmark functions for large scale global optimization [193]

FEs	Function	f_1	f_2	f_3	f_4	f_5
	Best	1.17E+07	2.56E+02	1.96E+00	2.05E+13	3.06E+08
	Median	1.95E+07	2.85E+02	2.06E+00	4.86E+13	4.14E+08
1.20E+05	Worst	5.75E+07	3.11E+02	2.21E+00	1.09E+14	5.56E+08
	Mean	2.19E+07	2.85E+02	2.07E+00	5.46E+13	4.20E+08
	Std	8.95E+06	1.42E+01	7.17E-02	2.23E+13	6.24E+07
	Best	2.37E-02	2.87E+01	1.68E-03	2.81E+12	2.99E+08
	Median	1.39E-01	4.27E+01	2.05E-03	6.40E+12	4.08E+08
6.00E+05	Worst	8.60E-01	5.73E+01	3.06E-03	1.50E+13	5.46E+08
	Mean	1.88E-01	4.24E+01	2.13E-03	6.86E+12	4.13E+08
	Std	1.98E-01	7.12E+00	3.52E-04	3.08E+12	6.14E+07
	Best	0.00E+00	2.88E-12	3.13E-13	3.84E+11	8.95E+06
	Median	0.00E+00	1.17E-10	3.34E-13	9.93E+11	2.39E+07
3.00E+06	Worst	0.00E+00	5.95E-08	6.36E-13	1.80E+12	3.86E+08
	Mean	0.00E+00	4.94E-09	3.51E-13	1.01E+12	1.03E+08
	Std	0.00E+00	1.41E-08	6.49E-14	3.54E+11	1.41E+08

Table 6.13 – Continued on next page

Table 6.13: **(continued from previous page)**

FEs	Function	f_6	f_7	f_8	f_9	f_{10}
	Best	1.85E+07	2.72E+10	6.29E+08	4.70E+08	4.04E+03
	Median	2.02E+07	4.57E+10	2.80E+09	5.79E+08	4.37E+03
1.20E+05	Worst	2.04E+07	7.82E+10	1.75E+10	7.43E+08	4.73E+03
	Mean	2.01E+07	4.63E+10	4.40E+09	5.82E+08	4.38E+03
	Std	3.84E+05	1.01E+10	4.65E+09	7.02E+07	1.70E+02
	Best	9.59E+06	2.49E+09	9.57E+06	8.29E+07	3.43E+03
	Median	1.94E+07	5.56E+09	9.34E+07	1.04E+08	4.09E+03
6.00E+05	Worst	1.98E+07	1.54E+10	1.57E+08	1.32E+08	4.47E+03
	Mean	1.90E+07	6.38E+09	9.13E+07	1.05E+08	3.98E+03
	Std	1.99E+06	2.69E+09	4.32E+07	1.18E+07	2.67E+02
	Best	3.51E+06	1.97E+07	1.22E+05	1.08E+07	8.36E+01
	Median	5.23E+06	4.18E+07	3.15E+06	1.48E+07	7.91E+02
3.00E+06	Worst	6.51E+06	1.73E+08	7.42E+07	1.70E+07	1.55E+03
	Mean	5.24E+06	5.16E+07	8.35E+06	1.44E+07	7.72E+02
	Std	7.19E+05	3.64E+07	1.56E+07	1.61E+06	3.62E+02
FEs	Function	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
	Best	1.99E+02	4.31E+05	3.46E+04	9.93E+08	8.00E+03
	Median	2.03E+02	4.92E+05	1.15E+05	1.23E+09	8.54E+03
1.20E+05	Worst	2.05E+02	6.21E+05	8.41E+05	1.42E+09	9.10E+03
	Mean	2.03E+02	5.05E+05	2.05E+05	1.22E+09	8.55E+03
	Std	1.02E+00	4.82E+04	1.99E+05	1.10E+08	2.41E+02

Table 6.13 – Continued on next page

Table 6.13: **(continued from previous page)**

	Best	1.55E+02	7.46E+04	8.70E+02	2.44E+08	7.67E+03
	Median	1.68E+02	9.16E+04	1.19E+03	2.73E+08	8.11E+03
6.00E+05	Worst	1.94E+02	1.45E+05	7.41E+03	3.10E+08	8.59E+03
	Mean	1.70E+02	9.88E+04	1.50E+03	2.72E+08	8.12E+03
	Std	1.12E+01	2.02E+04	1.28E+03	1.87E+07	2.22E+02
	Best	5.16E+01	1.05E+03	9.92E-03	2.51E+07	9.92E+02
	Median	5.65E+01	1.87E+03	8.04E+01	3.19E+07	2.21E+03
3.00E+06	Worst	6.20E+01	4.74E+03	1.82E+02	4.47E+07	3.19E+03
	Mean	5.65E+01	2.07E+03	7.96E+01	3.22E+07	2.17E+03
	Std	2.54E+00	9.82E+02	5.88E+01	3.71E+06	6.79E+02
FEs	Function	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
	Best	3.99E+02	9.91E+05	1.41E+05	2.22E+07	1.58E+05
	Median	4.03E+02	1.08E+06	3.08E+05	5.24E+07	3.49E+05
1.20E+05	Worst	4.05E+02	1.19E+06	1.38E+06	1.36E+08	8.14E+05
	Mean	4.02E+02	1.09E+06	4.06E+05	5.89E+07	3.77E+05
	Std	1.25E+00	5.15E+04	3.15E+05	2.99E+07	1.84E+05
	Best	3.06E+02	2.06E+05	2.34E+03	1.09E+06	1.84E+03
	Median	3.52E+02	2.61E+05	2.68E+03	1.62E+06	2.15E+03
6.00E+05	Worst	3.90E+02	3.18E+05	5.85E+03	2.18E+06	2.46E+03
	Mean	3.51E+02	2.61E+05	2.83E+03	1.58E+06	2.14E+03
	Std	2.38E+01	2.46E+04	6.77E+02	2.75E+05	1.69E+02
	Best	1.08E+02	6.36E+03	1.84E+02	2.54E+05	3.81E+02
	Median	1.14E+02	8.81E+03	4.01E+02	3.35E+05	8.22E+02
3.00E+06	Worst	1.25E+02	1.42E+04	8.05E+02	3.97E+05	1.10E+03
	Mean	1.15E+02	9.33E+03	4.17E+02	3.39E+05	8.19E+02
	Std	4.28E+00	1.96E+03	1.51E+02	3.76E+04	1.76E+02

Acknowledgment

This work is made possible by a grant from the National Science Foundation and NSERC Discovery No. 327482.

6.6 Conclusions

My proposed HMP-CA and its improved versions which are the key contributions of this dissertation are fully characterized in this chapter. HMP-CA is a new architecture for optimization algorithms, in which there are a number of local CAs designed to work on different parts of the given problem. In this architecture, there is only one belief space which is shared between all the local CAs. The shared belief space is responsible for tracking the best component found for each sub-problem.

Due to incorporating the concept of partial solution in the proposed architecture, it has been shown that HMP-CA is an efficient method in terms of both CPU time and memory usage.

Since the local CAs in a HMP-CA are designed to optimize different sub-problems, the incorporated problem decomposition technique has a major effect on the algorithm performance. The first HMP-CA which is represented in Section 6.2 uses a static decomposition technique which is called jumping approach.

In order to improve the first HMP-CA, a research is conducted to evaluate various static decomposition techniques which is described in Section 6.3. The investigated techniques are categorized into two classes of balanced and imbalanced approaches. The results of this investigation shows that the imbalanced techniques which assign different number of dimensions to different local CAs outperform the balanced techniques. Furthermore, the best results are obtained by the imbalanced techniques including the local CAs with only one assigned dimension.

The results of this investigation suggest to incorporate a dynamic decomposition technique instead of the static ones. Section 6.4 presents my proposed dynamic approaches including top-down and bottom-up methods. The results of evaluating HMP-CA with these dynamic techniques reveal that the dynamic techniques outperforms the static ones and also they show that the bottom-up approach is a more

efficient approach compared to the top-down technique. HMP-CA incorporating a dynamic decomposition technique is also called Dynamic HMP-CA (D-HMP-CA).

The bottom-up decomposition technique in D-HMP-CA starts by a number of local CAs assigned with only one dimensions, and then the non-improving local CAs will be merged together to improve the convergence rate. This merging mechanism is kind of random. Therefore, incorporating a more systematic way to choose dimensions for merging would results in a better performance. Section 6.5 introduces a new technique to detect the variable additively interactions which is called Variable Additively Interdependence Learning (VAIL). The VAIL module is incorporated in the dynamic bottom-up decomposition approach in order to force merging the additively interacting dimensions into one local CA and to avoid merging additively independent dimensions. The dynamic approach incorporating the VAIL module is called an adaptive approach and consequently the HMP-CA using the adaptive approach is called Adaptive HMP-CA (A-HMP-CA).

The proposed HMP-CA and its improved versions are evaluated by incorporating a number of numerical optimization benchmark problems. The results of this evaluation reveal that the proposed methods offer a very good performance compared to the state-of-the-art methods. Furthermore, A-HMP-CA is evaluated over the large scale global optimization problems the results of which show that it significantly outperforms the state-of-the-art methods.

In my proposed HMP-CAs including the dynamic and adaptive ones, a simple DE is incorporated as the evolutionary approach within each local CA. Therefore, the algorithm performance can also be further improved by incorporating more effective evolutionary approaches. This improvement is considered as a future direction for this research study.

Chapter 7

Conclusions

Optimization problems is a class of problems where the goal is to make a system as effective as possible. Since the solution space of optimization problems is very large, an optimization algorithm should be efficient as well. In other words, an optimization algorithm should be able to find the optimal solution within an acceptable time. Therefore, there are two measures for evaluating optimization algorithms including effectiveness and efficiency.

Evolutionary Algorithms (EAs) is a subset of population-based methods which are successfully applied to solve optimization problems. In this dissertation, I start reviewing the existing evolutionary methods and looking into their corresponding future directions. Due to this review, I found some rooms to improve the existing algorithms.

My first published algorithm [166] was a MA consisting of a GA and a local search heuristic benefiting from a new representation for classical JSSPs. This MA is improved by incorporating a new local search heuristic as well as a new fitness function which is called Priority-Based Fitness Function [169].

I proposed a more powerful MA by combining a GA with a VNS [167]. This combination shows that the VNS is more likely a local search heuristic and it will

be improved by incorporating a more explorative method such as a GA. This MA is improved by incorporating a more explorative method, namely a DE [172]. This MA is further improved by incorporating a Multi-Population DE which is more explorative compared to the traditional DE [172].

After investigating MAs, I found that incorporating knowledge may still improve the existing methods. Therefore, I focused on the area of CAs. Within this area, I found that although their successful applications have been reported in various area, CAs still suffer from their immature convergence. Since one of the most useful strategies to overcome this limitation is to incorporate multiple populations, I conducted a literature survey to highly investigate the area and its applications. This survey helps me to better understand the existing architectures to implement MP-CAs and their corresponding strengthes and weaknesses.

Based on the conducted survey, I introduced the applications of MP-CAs in JSSPs for the first time by publishing a MP-CA incorporating normative and topographic knowledge [168]. Afterwards, I proposed another MP-CA to deal with JSSPs by introducing a new knowledge called structured belief [170].

After publishing a couple of MP-CAs using the existing architectures, I introduced a new architecture for optimization problems. The new architecture which is called Heterogeneous MP-CA is the key contribution of this dissertation. The proposed architecture incorporates a number of CAs designed to work on different parts of the given problem [171].

In order to incorporate HMP-CAs, first a problem decomposition technique should be used to divide the given problem into a number of sub-problems. The first version of HMP-CA [171] incorporates a jumping approach to divide the problem dimensions among local CAs. In order to improve the proposed architecture, a research study is conducted to evaluate and compare a number of static decomposition techniques. The results of this research show that the imbalanced techniques which assigns different

number of dimensions to different local CAs outperforms others by presenting better solutions [165].

The HMP-CA is further improved by incorporating a dynamic decomposition technique which is called bottom-up approach [174]. The improved HMP-CA which is called Dynamic HMP-CA (D-HMP-CA) starts with a number of local CAs, each of which is designed to optimize only one of the problem dimensions, and then it merges the non-improving local CAs to speed up the algorithm convergence.

The D-HMP-CA is also improved by incorporating a more systematic merging strategy. The improved D-HMP-CA which is called Adaptive HMP-CA (A-HMP-CA) incorporates a technique to detect variable additively interactions [173]. The new technique which is called Variable Additively Interdependence Learning (VAIL) is incorporated within the dynamic decomposition technique in order to merge the interacting dimensions into a local CA and preserve the independent dimensions within different local CAs.

The proposed HMP-CA architecture is evaluated over numerical optimization problems. The evaluation results reveal that the proposed method is an effective as well as efficient algorithm compared to the state-of-the-art methods. The improved version, D-HMP-CA, is more effective such that it is able to find the optimal solution for every single run. The further improved version, A-HMP-CA, shows a great performance to deal with large scale global optimization problems such that it significantly outperforms the state-of-the-art methods.

Since A-HMP-CA incorporates a simple DE as the evolutionary approach within its local CAs, it can be further improved by incorporating more powerful evolutionary approaches recently introduced in the literature. This improvement is considered as the future direction for this research.

Bibliography

[1] M. A. Abido, "Optimal design of power-system stabilizers using particle swarm optimization," *IEEE Transactions on Energy Conversion*, vol. 17(3), pp. 406–413, 2002.

[2] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, vol. 34(3), pp. 391–401, 1988.

[3] C. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier, "A global optimization method, abb, for general twice-differentiable constrained nlps-1. theoretical advances," *Computers and Chemical Engineering*, vol. 22, pp. 1137–1158, 1998.

[4] K. Al-Mutawah, V. Lee, and Y. Cheung, "An intelligent algorithm for modeling and optimizing dynamic supply chains complexity," in *International Conference on Intelligent Computing (ICIC)*, vol. 4113/2006. Kunming Yunnan Province, China: Springer-Verlag, Berlin, Germany, 2006, pp. 975–980.

[5] ——, "Modeling supply chain complexity using a distributed multi-objective genetic algorithm," in *International Conference on Computational Science and its Applications (ICCSA)*, Glasgow, Scotland, 2006.

[6] J. Alami, L. Benameur, and E. Imrani, "Fuzzy clustering based parallel cultural algorithm," *Soft Computing*, vol. 2(4), pp. 562–571, 2007.

[7] J. Alami and A. E. Imrani, "Dielectric composite multimodal optimization using a multipopulation cultural algorithm," *Intelligent Data Analysis*, vol. 12(4), pp. 359–378, 2008.

[8] J. Alami, A. E. Imrani, and A. Bouroumi, "A multipopulation cultural algorithm using fuzzy clustering," *Applied Soft Computing*, vol. 7(2), pp. 506–519, 2007.

[9] M. Asano and H. Ohta, "A heuristic for job shop scheduling to minimize total weighted tardiness," *Computers and Industrial Engineering*, vol. 42, pp. 137–147, 2002.

[10] A. Bagheri, M. Zandieh, I. Mahdavi, and M. Yazdani, "An artificial immune algorithm for the flexible job-shop scheduling problem," *Future Generation Computer Systems*, vol. 26, pp. 533–541, 2010.

[11] K. R. Baker, *Introduction to Sequencing and Scheduling.* Wiley, 1974.

[12] E. Balas and A. Vazacopoulos, "Guided local search with shifting bottleneck for job shop scheduling," *Management Science*, vol. 44 (2), pp. 262–275, 1998.

[13] D. Beasley, D. Bull, and R. Martin, "An overview of genetic algorithms. part 2. research topics," *University Computing*, vol. 15(4), pp. 170–181, 1993.

[14] R. Becerra and C. Coello, "A cultural algorithm for solving the job-shop scheduling problem," in *Knowledge Incorporation in Evolutionary Computation, Studies in Fuzziness and Soft Computing*, vol. 167. Springer, 2005, pp. 37–55.

[15] R. Becerra and C. C. Coello, "A cultural algorithm with differential evolution to solve constrained optimization problems," in *IBERAMIA*, ser. LNCS(LNAI), C. Lemaitre, C. Reyes, and J. Gonzalez, Eds., vol. 3315. Springer, Heidelberg, 2004, pp. 881–890.

[16] R. Becerra and C. Coello, "Cultured differential evolution for constrained optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, pp. 4303–4322, 2006.

[17] F. V. D. Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8(3), pp. 225–239, 2004.

[18] C. Bierwirth, "A generalized permutation approach to job shop scheduling with genetic algorithms," *OR Spectrum. Special Issue on Applied Local Search*, vol. 17 (2-3), pp. 87–92, 1995.

[19] P. Bin, "Knowledge and population swarms in cultural algorithms for dynamic environments," Ph.D. dissertation, Wayne State University, USA, 2005.

[20] J. Blazewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: Conventional and new solution techniques," *European Journal of Operational Research*, vol. 93 (1), pp. 1–33, 1996.

[21] J. Bracken and G. McCormick, *Selected applications of nonlinear programming.* New York: John Wiley and Sons, 1968.

[22] P. Brandimarte, "Routing and scheduling in a flexible job shop by taboo search," *Annals of Operations Research*, vol. 41, pp. 157–183, 1993.

[23] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10 (6), pp. 646–657, 2006.

[24] J. Brimberg, N. Mladenovic, D. Urosevic, and E. Ngai, "Variable neighborhood search for the heaviest k-subgraph," *Computers and Operations Research*, vol. 36 (11), pp. 2885–2891, 2009.

[25] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Computing*, vol. 45, pp. 369–375, 1990.

[26] E. Burke and S. Alistair, "A memetic algorithm for the maintenance scheduling problem," in *International Conference on Neural Information Processing (ICONIP)*. Dunedin, New Zealand: Springer, Berlin, 1997, pp. 469–474.

[27] E. Burke and A. Smith, "Hybrid evolutionary techniques for the maintenance scheduling problem," *IEEE Transactions on Power Systems*, vol. 15(1), pp. 122–128, 2000.

[28] A. Caumond, P. Lacomme, and N. Tcherneva, "A memetic algorithm for the job-shop with time-lags," *Computers and Operations Research*, vol. 35(7), pp. 2331–2356, 2008.

[29] C.-H. Chen, Y.-C. Liu, C.-J. Lin, and C.-T. Lin, "A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks," in *IEEE International Conference on Fuzzy Systems, FUZZ-IEEE (IEEE World Congress on Computational Intelligence)*, 2008, pp. 238–245.

[30] H. Chen, J. Ihlow, and C. Lehmann, "A genetic algorithm for flexible job-shop scheduling," in *IEEE International Conference on Robotics and Automation*, Detroit, 1999, pp. 1120–1125.

[31] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Parallel Problem Solving from Nature (PPSN XI)*, vol. 6239, 2010, pp. 300–309.

[32] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms i: representation," *Computers and Industrial Engineering*, vol. 30 (4), pp. 983–997, 1996.

[33] T.-C. Chiang, H.-C. Cheng, and L.-C. Fu, "Nnma: An effective memetic algorithm for solving multiobjective permutation flow shop scheduling problems," *Expert Systems with Applications*, 2010.

[34] L. Coelho and V. Mariani, "An efficient particle swarm optimization approach based on cultural algorithm applied to mechanical design," in *IEEE Conference of Evolutionary Computation*, 2006, pp. 1099–1104.

[35] C. C. Coello and R. Becerra, "A cultural algorithm for constrained optimization," in *Second Mexican International Conference on Artificial Intelligence: Advances in Artificial Intelligence (MICAI)*, vol. 2313, 2002, pp. 98–107.

[36] ——, "Evolutionary multiobjective optimization using a cultural algorithm," in *IEEE swarm intelligence symposium*, 2003, pp. 6–13.

[37] O. Cordon, M. del Jesus, F. Herrera, and M. Lozano, "Mogul: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach," *International Journal of Intelligent Systems*, vol. 14(11), pp. 1123–1153, 1999.

[38] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena, "Ten years of genetic fuzzy systems: Current framework and newtrends," *Fuzzy Sets and Systems*, vol. 141(1), pp. 5–31, 2004.

[39] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic Fuzzy Systems*. Singapore: World Scientific, 2001.

[40] D. Cortes, R. Becerra, and C. Coello, "Cultural algorithms, an alternative heuristic to solve the job shop scheduling problem," *Engineering Optimization*, vol. 39 (1), pp. 69–85, 2007.

[41] R. S. Cowder, "Predicting the mackey-glass time series with cascadecorrelation learning," in *1990 Connectionist Models Summer School*, 1990, pp. 117–123.

[42] T. Crainic, M. Gendreau, P. Hansen, and N. Mladenovic, "Cooperative parallel variable neighborhood search for the p-median," *Journal of Heuristics*, vol. 10, pp. 289–310, 2004.

[43] F. Croce, R. Tadei, and G. Volta, "A genetic algorithm for the job shop problem," *Computers in Operations Research*, vol. 22, pp. 15–24, 1995.

[44] D. da Silva and R. de Oliveira, "A multipopulation cultural algorithm based on genetic algorithm for the mkp," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO 09)*, Montral, Qubec, Canada, 2009.

[45] R. Q. dao-er ji and Y. Wang, "A new hybrid genetic algorithm for job shop scheduling problem," *Computers and Operations Research*, vol. 39 (10), pp. 2291–2299, 2012.

[46] R. Davies, "An introduction to mpi and parallel genetic algorithms," Cardiff HPC Training and Education Centre, Tech. Rep., 1996.

[47] L. Davis, "Job shop scheduling with genetic algorithms," in *In J. Grefenstette (ed.) Proceedings of the First International Conference on Genetic Algorithms*, Hillsdale, 1985, pp. 136–140.

[48] L. de Castro and J. Timmis, "An artificial immune network for multimodal function optimization," in *IEEE Congress on Evolutionary Computation (CEC)*, vol. 1, May, Hawaii, 2002, pp. 669–674.

[49] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6(2), pp. 182–197, 2002.

[50] M. Dell'Amico and M. Trubian, "Applying tabu search to the job-shop scheduling problem," *Annals of Operations Research*, vol. 41 (3), pp. 231–252, 1993.

[51] J. Digalakis and K. Margaritis, "A multipopulation cultural algorithm for the electrical generator scheduling problem," *Mathematics and Computers in Simulation*, vol. 60(3-5), pp. 293–301, 2002.

[52] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Sixth International Symposium on Micro Machine and Human Science (MHS)*, Nagoya, Japan, 1995, pp. 39–43.

[53] E. Falkenauer and S. Bouffoix, "A genetic algorithm for job shop," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, 1991.

[54] H. Fang, P. Ross, and D. Corne, "A promising genetic algorithm approach to job shop scheduling, rescheduling and open shop scheduling," in *In S. Forrest (ed.) Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993, pp. 375–382.

[55] A. Felipe, M. T. Ortuno, and G. Tirado, "The double traveling salesman problem with multiple stacks: a variable neighborhood search approach," *Computers and Operations Research*, vol. 36 (11), pp. 2983–2993, 2009.

[56] K. Fleszar and K. S. Hindi, "An effective vns for the capacitated p-median problem," *European Journal of Operational Research*, vol. 191 (3), pp. 612–622, 2008.

[57] K. Fleszar, I. H. Osman, and K. S., "A variable neighbourhood search algorithm for the open vehicle routing problem," *European Journal of Operational Research*, vol. 195, pp. 803–809, 2009.

[58] D. Fonseca and D. Navaresse, "Artificial neural networks for job shop simulation," *Advanced Engineering Informatics*, vol. 16(4), pp. 241–246, 2002.

[59] S. French, *Sequencing and Scheduling.* Ellis Horwood, Chichester, 1982.

[60] Z. L. Gaing, "A particle swarm optimization approach for optimum design of pid controller in avr system," *IEEE Transactions on Energy Conversion*, vol. 19(2), pp. 384–391, 2004.

[61] F. Gao, G. Cui, and H. Liu, "Integration of genetic algorithm and cultural algorithms for constrained optimization," in *International Conference on Neural Information Processing (ICONIP)*, I. King, J. Wang, L.-W. Chan, and D. Wang, Eds., vol. 4234. Springer, Heidelberg, 2006, pp. 817–825.

[62] J. Gao, M. Gen, L. Sun, and X. Zhao, "A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems," *Computers and Industrial Engineering*, vol. 53(1), pp. 149–162, 2007.

[63] J. Gao, L. Sun, and M. Gen, "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems," *Computers and Operations Research*, vol. 35(9), pp. 2892–2907, 2008.

[64] S. Garca, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the cec'2005 special session on real parameter optimization," *Journal of Heuristics*, vol. 15, pp. 617–644, 2009.

[65] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, pp. 117–129, 1976.

[66] M. Gen, J. Tsujimura, and E. Kubota, "Solving job-shop scheduling problem using genetic algorithms," in *In M. Gen and S. Kobayashi (eds.) Proceedings of the 16th International Conference on Computers and Industrial Engineering*, Ashikaga, Japan, 1994, pp. 576–579.

[67] D. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, Boston, MA, USA, 1989.

[68] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison Wesley, 1989.

[69] D. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *2nd International Conference on Genetic Algorithms*, ser. 41-49, 1987.

[70] J. Goncalves, J. de Magalhaes Mendes, and M. G. C. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," AT&T Labs, Tech. Rep. TD-5EAL6J, 2002.

[71] D. Gong, Y. Zhou, and T. Li, "Cooperative interactive genetic algorithm based on user's preference," *International Journal of Information Technology*, vol. 11(10), pp. 1–10, 2005.

[72] A. Gonzalez and R. Perez, "Slave: A genetic learning system based on an iterative approach," *IEEE Transactions on Fuzzy Systems*, vol. 7(2), pp. 176–191, 1999.

[73] J. Gu, M. Gu, C. Cao, and X. Gu, "A novel competitive coevolutionary quantum genetic algorithm for stochastic job shop scheduling problem," *Computers and Operations Research*, vol. 37, pp. 927–937, 2010.

[74] Y.-N. Guo, Y.-Y. Cao, Y. Lin, and H. Wang, "Knowledge migration based multi-population cultural algorithm," in *Fifth International Conference on Natural Computation (ICNC 09)*, 2009, pp. 331–335.

[75] Y.-N. Guo, Y.-Y. Cao, and D.-D. Liu, "Multi-population multi-objective cultural algorithm," *Advanced Materials Research*, vol. 156-157, pp. 52–55, 2010.

[76] Y.-N. Guo, J. Cheng, Y.-Y. Cao, and Y. Lin, "A novel multi-population cultural algorithm adopting knowledge migration," *Soft Computing*, vol. 15(5), pp. 897–905, 2011.

[77] Y.-N. Guo, J. Cheng, D.-W. Gong, and D.-Q. Yang, "Knowledge-inducing interactive genetic algorithms based on multi-agent," in *The Second International Conference on Advances in Natural Computation (ICNC)*, ser. Part I, L. Jiao, L. Wang, X.-B. Gao, J. Liu, and F. Wu, Eds., vol. 4221, 2006, pp. 759–768.

[78] Y.-N. Guo, J. Cheng, and Y. Lin, "Cooperative interactive cultural algorithms adopting knowledge migration," in *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC)*, 2009, pp. 193–199.

[79] Y.-N. Guo, D.-W. Gong, and D.-Q. Yang, "Interactive genetic algorithms based on implicit knowledge model," in *6th International Conference on Simulated Evolution and Learning (SEAL)*, China, 2006.

[80] Y.-N. Guo, Y. Lin, M. Yang, and S.-G. Zhang, "User's preference aggregation based on parallel interactive genetic algorithms," *Applied Mechanics and Materials*, vol. 34-35, pp. 1159–1164, 2010.

[81] Y.-N. Guo, Y. Lin, and S.-G. Zhang, "Interactive genetic algorithms based on frequent pattern mining," in *Sixth International Conference on Natural Computation (ICNC)*, 2010, pp. 2381–2385.

[82] Y.-N. Guo and D. Liu, "Multi-population cooperative particle swarm cultural algorithms," in *Seventh International Conference on Natural Computation (ICNC)*, Shanghai, 2011, pp. 1351–1355.

[83] Y.-N. Guo, S. Zhang, J. Cheng, and Y. Lin, "Cooperative interactive cultural algorithms based on dynamic knowledge alliance," in *Proceedings of International Conference on Artificial Intelligence and Computational Intelligence (AICI)*, H. Deng, D. Miao, J. Lei, and F. L. Wang, Eds., vol. 7004, 2011, pp. 204–211.

[84] S. Hasan, R. Sarker, and D. Cornforth, "Hybrid genetic algorithm for solving job-shop scheduling problem," in *6th IEEE/ACIS international conference on computer and information science*, Melbourne, 2007, pp. 519–524.

[85] ——, "Modified genetic algorithm for job-shop scheduling: a gap-utilization technique," in *IEEE congress on evolutionary computation*, Singapore, 2007, pp. 3804–3811.

[86] ——, "Ga with priority rules for solving job-shop scheduling problems," in *IEEE world congress on computational intelligence*, Hong Kong, 2008, pp. 1913–1920.

[87] S. Hasan, R. Sarker, D. Essam, and D. Cornforth, "Memetic algorithms for solving job-shop scheduling problems," *Memetic Computing*, vol. 1, pp. 69–83, 2008.

[88] Y. He and D. Chen, "Constrained ant colony system and its application in process optimization of butene alkylation," *Journal of Chemical Industry and Engineering*, vol. 56, pp. 1708–1713, 2005, in Chinese.

[89] F. Herrera, "Genetic fuzzy systems: Status, critical considerations and future directions," *International Journal of Computational Intelligence Research*, vol. 1(1), pp. 59–67, 2005.

[90] T. Hiroyasu and H. Yokouchi, "Extraction of design variables using collaborative filtering for interactive genetic algorithms," in *IEEE International Conference on Fuzzy Systems*, 2009, pp. 1579–1584.

[91] N. Ho and J. Tay, "Genace: An efficient cultural algorithm for solving the flexible job-shop problem," in *IEEE Congress on Evolutionary Computation (CEC)*, 2004, pp. 1759–1766.

[92] ——, "Lega: an architecture for learning and evolving flexible job-shop schedules," in *IEEE Congress on Evolutionary Computation (CEC)*, 2005, pp. 1380–1387.

[93] J. H. Holland, *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[94] H. Huang, X. Gu, , and M. Liu, "Research on cultural algorithm for solving nonlinear constrained optimization," *ACTA AUTOMATICA SINICA*, vol. 33, pp. 1115–1120, 2007, in Chinese.

[95] K. Huang and C. Liao, "Ant colony optimization combined with taboo search for the job shop scheduling problem," *Computers and Operations Research*, vol. 35(4), pp. 1030–1046, 2008.

[96] W. Huang and A. Yin, "An improved shifting bottleneck procedure for the job shop scheduling problem," *Computers and Operations Research*, vol. 31, pp. 2093–2110, 2004.

[97] E. Hurink, B. Jurisch, and M. Thole, "Tabu search for the job shop scheduling problem with multi-purpose machines," *Operations Research Spektrum*, vol. 15, pp. 205–215, 1994.

[98] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multi-dimensional pattern classification problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29(5), pp. 601–618, 1999.

[99] J.-S. R. Jang, "Anfis: Adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23(3), pp. 665–685, 1993.

[100] C. Janikow and Z. Michalewicz, "An experimental comparison of binary and floating point representations in genetic algorithms," in *4th international conference on genetic algorithms*, San Mateo, CA, 1991, pp. 151–157.

[101] J. A. Joines, D. Kupta, M. A. Gokce, R. E. King, and M. Kay, "Supply chain multi-objective simulation optimization," in *Winter Simulation Conference*, vol. 2, Raleigh, NC, USA, 2002, pp. 1306–1314.

[102] C. F. Juang, J. Y. Lin, and C. T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 30(2), pp. 290–302, 2000.

[103] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multi objective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 32(1), pp. 1–13, 2002.

[104] ——, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic," *Journal of Mathematics and Computers in Simulation*, vol. 60, pp. 245–276, 2002.

[105] N. K. Kasabov and Q. Song, "Denfis: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Transactions on Fuzzy Systems*, vol. 10(2), pp. 144–154, 2002.

[106] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, vol. 4, Perth, W.A., Australia, 1995, pp. 1942–1948.

[107] S. Kirkpatrick, C. G. Jr., and M. Vecchi, "Optimisation by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[108] M. Kolonko, "Some new results on simulated annealing applied to the job-shop scheduling problem," *European Journal of Operation Research*, vol. 113, pp. 123–136, 1999.

[109] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[110] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evolutionary Computation*, vol. 7, pp. 19–44, 1999.

[111] J. Laredo, P. Castillo, M. Mora, and J. Merelo, "Evolvable agents, a fine grained approach for distributed evolutionary computing: walking towards the peer-to-peer computing frontiers," *Soft Computing*, vol. 12(12), pp. 1145–1156, 2008.

[112] D. Lawrence, "Job shop scheduling with genetic algorithms," in *First international conference on genetic algorithms*, Mahwah, New Jersey, 1985, pp. 136–140.

[113] S. Lawrence, "Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques," Master's thesis, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[114] K. M. Lee, T. Yamakawa, and K. M. Lee, "A genetic algorithm for general machine scheduling problems," *International Journal of Knowledge-Based Electronic*, vol. 2, pp. 60–66, 1998.

[115] X. Li, "Adaptively choosing neighbourhood bests using species in swarm optimizer for multimodal function optimization," in *Genetic and Evolutionary Computation Conference (GECCO)*, 2004, pp. 105–116.

[116] C. J. Lin, "A ga-based neural fuzzy system for temperature control," *Fuzzy Sets and Systems*, vol. 143(2), pp. 311–333, 2004.

[117] C.-J. Lin, C.-H. Chen, and C.-T. Lin, "Efficient self-evolving evolutionary learning for neurofuzzy inference systems," *IEEE Transactions on Fuzzy Systems*, vol. 16(6), pp. 1476–1490, 2008.

[118] ——, "A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 39(1), pp. 55–68, 2009.

[119] C. J. Lin and C. T. Lin, "An art-based fuzzy adaptive learning control network," *IEEE Transactions on Fuzzy Systems*, vol. 5(4), pp. 477–496, 1997.

[120] C.-J. Lin, C.-C. Weng, C.-L. Lee, and C.-Y. Lee, "Using an efficient hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy network design," in *International Conference on Machine Learning and Cybernetics*, 2009, pp. 3076–3082.

[121] W. Lin, T. Hong, and S. Liu, "On adapting migration parameters for multi-population genetic algorithms," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 6, 2004, pp. 5731–5735.

[122] S. Ling, F. Leung, H. Lam, Y. Lee, and P. Tam, "A novel genetic-algorithm-based neural network for short-term load forecasting," *IEEE Transactions on Industrial Electronics*, vol. 50(4), pp. 793–799, 2003.

[123] Y. Liu, Z. Qin, Z. Shi, and J. Lu, "Center particle swarm optimization," *Neurocomputing*, vol. 70, pp. 672–679, 2007.

[124] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *IEEE Congress on Evolutionary Computation (CEC)*, 2001, pp. 1101–1108.

[125] H.-M. Ma and C.-M. Ye, "Parallel particle swarm optimization algorithm based on cultural evolution," *Computer Engineering*, vol. 34(2), pp. 193–195, 2008.

[126] I. Mahdavi, B. Shirazi, and M. Solimanpur, "Development of a simulation-based decision support system for controlling stochastic flexible job shop manufacturing systems," *Simulation Modelling Practice and Theory*, vol. 18, pp. 768–786, 2010.

[127] R. Mansini and B. Tocchella, "The traveling purchaser problem with budget constraint," *Computers and Operations Research*, vol. 36 (7), pp. 2263–2274, 2009.

[128] M. Mansour, S. F. Mekhamer, and N. E.-S. El-Kharbawe, "A modified particle swarm optimizer for the coordination of directional overcurrent relays," *IEEE Transactions on Power Delivery*, vol. 22(3), pp. 1400–1410, 2007.

[129] H. Matsuo, C. Suh, and R. Sullivan, "A controlled search simulated annealing method for the general job shop scheduling problem," in *Working paper 03-04-88*. University of Texas at Austin, 1988.

[130] D. Mattfeld and C. Bierwirth, "An efficient genetic algorithm for job shop scheduling with tardiness objectives," *European Journal of Operational Research*, vol. 155, pp. 616–630, 2004.

[131] A. Mendes, F. Muller, P. Franca, and P. Moscato, "Comparing meta-heuristic approaches for parallelcascheduling problems with sequence dependent setup times," in *15th International Conference on CAD/CAM Robotics & Factories of the Future (CARS & FOF)*, 1999.

[132] R. Mendes and A. S. Mohais, "DynDE: A differential evolution for dynamic optimization problems," in *IEEE Congress on Evolutionary Computation (CEC)*, vol. 2, 2005, pp. 2808–2815.

[133] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Genetic and Evolutionary Computation Conference (GECCO)*, 2006, pp. 485–492.

[134] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, 1994.

[135] M. Miki, Y. Yamamoto, and S. Wake, "Global asynchronous distributed interactive genetic algorithm," in *IEEE International Conference on Systems, Man and Cybernetics*, 2006, pp. 3481–3485.

[136] N. Mladenovic and P. Hansen, "Variable neighborhood search," *Computers and Operations Research*, vol. 24, pp. 1097–1100, 1997.

[137] M. Mobini, Z. Mobini, and M. Rabbani, "An artificialimmunealgorithmfortheprojectschedulingproblemunder resource constraints," *Applied Soft Computing*, 2010.

[138] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Machine Learning*, vol. 22, pp. 11–32, 1996.

[139] P. Moscato, "Memetic algorithms: A short introduction," *New Ideas in Optimization*, vol. 14, pp. 219–234, 1999.

[140] P. Moscato and D. Holstein, *New Ideas in Optimisation*. McGraw-Hill, New York, 1999, ch. Memetic algorithms using guided local search: a case study, pp. 235–244.

[141] B. Naderi, S. F. Ghomi, M. Aminnayeri, and M. Zandieh, "Scheduling open shops with parallel machines to minimize total completion time," *Journal of Computational and Applied Mathematics*, 2010.

[142] N. Nedjah, E. Alba, and L. de Macedo Mourelle, *Parallel Evolutionary Computations*. Springer, 2006.

[143] D. Nguyen and B.Widrow, "The truck backer-upper: An example of selflearning in neural network," in *International Joint Conference on Neural Networks (IJCNN)*, vol. 2, 1989, pp. 357–363.

[144] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12 (1), pp. 107–125, 2008.

[145] H. Nomura, I. Hayashi, and N. Wakami, "A learning method of fuzzy inference rules by descent method," in *IEEE International Conference on Fuzzy Systems*, San Diego, CA, USA, 1992, pp. 203–210.

[146] E. Nowicki and C. Smutnicki, "A fast taboo search algorithm for the job shop scheduling problem," *Management Science*, vol. 42 (6), pp. 797–813, 1996.

[147] B. Ombuki and M. Ventresca, "Local search genetic algorithms for the job shop scheduling problem," *Applied Intelligence*, vol. 21, pp. 99–109, 2004.

[148] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *IEEE Congress on Computational Intelligence (CEC)*, 2010, pp. 1–8.

[149] Y. Ong and A. Keane, "Meta-lamarckian learning in memetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 99–110, 2004.

[150] Z. Ong, J. Tay, and C. Kwoh, "Applying the clonal selection principle to find flexible job-shop schedules," in *LNCS - Vol. 3627*, 2005, pp. 442–455.

[151] G. Onwubolu and D. Davendra, "Scheduling flow shops using differential evolution algorithm," *European Journal of Operational Research*, vol. 171 (2), pp. 674–692, 2006.

[152] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Transactions on Fuzzy Systems*, vol. 3(3), pp. 370–379, 1995.

[153] Q.-K. Pan, M. F. Tasgetiren, and Y.-C. Liang, "A discrete differential evolution algorithm for the permutation flowshop scheduling problem," *Computers & Industrial Engineering*, vol. 55 (4), pp. 795–816, 2008.

[154] B. Park, H. Choi, and H. Kim, "A hybrid genetic algorithm for the job shop scheduling problems," *Computers and Industrial Engineering*, vol. 45, pp. 597–613, 2003.

[155] A. Parodi and P. Bonelli, "A new approach to fuzzy classifier systems," in *5th International Conference on Genetic Algorithms*, 1993, pp. 223–230.

[156] J. Peter and B. Jordan, "Competitive environments evolve better solutions for complex tasks," in *Fifih International Conference on Genetic Algorithm*, 1993, pp. 264–270.

[157] F. Pezzella and E. Merelli, "A tabu search method guided by shifting bottleneck for the job shop scheduling problem," *European Journal of Operational Research*, vol. 120, pp. 297–310, 2000.

[158] F. Pezzella, G. Morganti, and G. C. ., "A genetic algorithm for the flexible jobshop scheduling problem," *Computers and Operations Research*, vol. 35(10), pp. 3202–3212, 2008.

[159] S. Ponnambalam, P. Aravindan, and P. S. Rao, "Comparative evaluation of genetic algorithms for job-shop scheduling," *Production Planning and Control*, vol. 12 (6), pp. 560–574, 2001.

[160] M. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," in *The 3rd Conference on Parallel Problem Solving from Nature (PPSN)*, vol. 2, 1994, pp. 249–257.

[161] K. Price, R. Storn, and J. Lampinen, *Differential evolution: a practical approach to global optimization (Natural Computing Series)*. New York, NY: Springer, 2005.

[162] B. Qian, L. Wang, R. Hu, W.-L. Wang, D.-X. Huang, and X. Wang, "A hybrid differential evolution method for permutation flow-shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 38 (7), pp. 757–777, 2008.

[163] L. Qiang, L. Ruiyu, and Y. Dongyun, "The adaptive particle swarm optimization based on the fuzzy cultural algorithm," *Computer Engineering and Science*, vol. 30(1), pp. 88–92, 2008.

[164] M. R. Raeesi N. and Z. Kobti, "A machine operation lists based memetic algorithm for job shop scheduling," in *IEEE Congress on Evolutionary Computation (CEC)*, New Orleans, LA, USA, 2011.

[165] M. R. Raeesi N., J. Chittle, and Z. Kobti, "A new dimension division scheme for heterogenous multi-population cultural algorithm," in *The 27th Florida Artificial Intelligence Research Society Conference (FLAIRS-27)*, Pensacola Beach, FL, USA, May 21-23 2014, pp. 75–80.

[166] M. R. Raeesi N. and Z. Kobti, "A machine operation lists based memetic algorithm for job shop scheduling," in *IEEE Congress on Evolutionary Computation (CEC)*, New Orleans, LA, USA, June 5-8 2011, pp. 2436–2443.

[167] ——, "Incorporating a genetic algorithm to improve the performance of variable neighborhood search," in *The Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC)*, Mexico City, Mexico, November 5-9 2012, pp. 144–149.

[168] ——, "A knowledge-migration-based multi-population cultural algorithm to solve job shop scheduling," in *The 25th Florida Artificial Intelligence Research Society Conference (FLAIRS-25)*, Marco Island, FL, USA, May 23-25 2012, pp. 68–73.

[169] ——, "A memetic algorithm for job shop scheduling using a critical-path-based local search heuristic," *Memetic Computing*, vol. 4 (3), pp. 231–245, 2012.

[170] ——, "A multiagent system to solve jssp using a multi-population cultural algorithm," in *The 25th Canadian Conference on Artificial Intelligence (Canadian AI)*, vol. 7310, Toronto, ON, Canada, May 28-30 2012, pp. 362–367.

[171] ——, "Heterogeneous multi-population cultural algorithm," in *IEEE Congress on Evolutionary Computation (CEC)*, Cancun, Mexico, June 20-23 2013, pp. 292–299.

[172] ——, "Incorporating highly explorative methods to improve the performance of variable neighborhood search," *Transactions on Computational Science XXI*, vol. 8160, pp. 315–338, 2013.

[173] ——, "Adaptive heterogenous multi-population cultural algorithm for large scale global optimization," 2014, (To Be Appeared).

[174] ——, "Heterogeneous multi-population cultural algorithm with a dynamic dimension decomposition strategy," in *The 27th Canadian Conference on Artificial Intelligence (Canadian AI)*, vol. 8436, Montreal, QC, Canada, May 6-9 2014, pp. 345–350.

[175] R. G. Reynolds, "An introduction to cultural algorithms," in *Thirs Annual Conference on Evolutionary Programming*, A. V. Sebald and L. J. Fogel, Eds. River Edge, New Jersey: World Scientific, 1994, pp. 131–139.

[176] T. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4(3), pp. 284–294, 2000.

[177] Y. Saab and V. Rao, "Combinatorial optimisation by stochastic evolution," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, pp. 525–535, 1991.

[178] S. Saleem, "Knowledge-based solution to dynamic optimization problems using cultural algorithms," Ph.D. dissertation, Wayne State University, USA, 2001.

[179] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Transactions on Evolutionary Computation*, vol. 2(3), pp. 97–106, 1998.

[180] T. Satoh and K. Nara, "Maintenance scheduling by using the simulated annealing method," *IEEE Transactions on Power Systems*, vol. 6, pp. 850–857, 1991.

[181] O. Schutze, E.-g. Talbi, C. Coello Coello, L. V. Santana-Quintero, and G. T. Pulido, "A memetic PSO algorithm for scalar optimization problems," in *IEEE Swarm Intelligence Symposium (SIS)*, Honolulu, HI, USA, April 1-5 2007, pp. 128–134.

[182] G. Shafer, *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, New Jersey, 1976.

[183] M. Sharma, M. Pandit, and L. Srivastava, "Reserve constrained multi-area economic dispatch employing differential evolution with time-varying mutation," *International Journal of Electrical Power and Energy Systems*, vol. 33 (3), pp. 753–766, 2011.

[184] L. Sheng, W. Xingyu, and Y. Xiaoming, "Cultured differential particle swarm optimization for numerical optimization problems," in *Third International Conference on Natural Computation*, 2007, pp. 642–646.

[185] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII (EP98)*, 1998, pp. 591–600.

[186] Y. Shi, H. Teng, and Z. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *The 1st International Conference on Natural Computation*, 2005, pp. 1080–1088.

[187] A. Silva, A. Neves, and E. Costa, "An empirical comparison of particle swarm and predator prey optimization," in *Proceedings of the Thirteenth Irish Conference on Artificial Intelligence and Cognitive Science*, ser. Lecture Notes in Computer Science, vol. 2464, 2002, pp. 103–110.

[188] S. H. Stewart, S. Taylor, J. M. Baker, F. Hoffmann, and G. Pfister, "Evolutionary design of a fuzzy knowledge base for a mobile robot," *International Journal of Approximate Reasoning*, vol. 17(4), pp. 447–469, 1997.

[189] R. Storn and K. Price, "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces," International Computer Science Institute, Berkeley, CA, Tech. Rep., 1995.

[190] ——, "Minimizing the real functions of the icec'96 contest by differential evolution," in *IEEE international conference on evolutionary computation*, New York, 1996, pp. 842–844.

[191] ——, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11 (4), pp. 341–359, 1997.

[192] X.-Y. Sun, X.-F. Wang, and D.-W. Gong, "A distributed co-interactive genetic algorithm and its applications to group decision-making," *Information and Control*, vol. 36(5), pp. 557–561, 2007.

[193] K. Tang, X. Li, S. P.N., Z. Yang, and T. Weise, "Benchmark functions for the CEC'2010 special session and competition on large scale global optimization," NICAL, USTC, Hefei, Anhui, China, Tech. Rep., 2009. [Online]. Available: http://nical.ustc.edu.cn/cec10ss.php

[194] M. F. Tasgetiren, M. Sevkli, Y. C. Liang, and G. Gencyilmaz, "Particle swarm optimization algorithm for single-machine total weighted tardiness problem," in *Congress on Evolutionary Computation (CEC)*, Portland, Oregan, USA, 2004, pp. 1412–1419.

[195] M. F. Tasgetiren and P. N. Suganthan, "A multi-populated differential evolution algorithm for solving constrained optimization problem," in *IEEE Congress on Evolutionary Computation (CEC)*, 2006, pp. 340–354.

[196] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Parallel differential evolution," in *IEEE Congress on Evolutionary Computation (CEC)*, 2004, pp. 2023–2029.

[197] M. Tomassini, *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[198] P. Tripathi, S. Bandyopadhyay, and S. Pal, "Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients," *Information Sciences*, vol. 177(22), pp. 5033–5049, 2007.

[199] T. H. Truong and F. Azadivar, "Simulation based optimization for supply chain configuration design," in *Winter Simulation Conference*, vol. 2, New Bedford, MA, USA, 2003, pp. 1268–1275.

[200] P. J. M. Van Laarhoven, E. H. L. Aarts, and J. K. Lenstra, "Job shop scheduling by simulated annealing," *Operations Research*, vol. 40 (1), pp. 113–125, 1992.

[201] M. Vasile, E. Minisci, and M. L. 2011., "An inflationary differential evolution algorithm for space trajectory optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15 (2), pp. 267–281, 2011.

[202] C. Wang, Y. Cao, and G. Dai, "Bi-directional convergence aco for job-shop scheduling," *Computer Integrated Manufacturing Systems*, vol. 10(7), pp. 820–824, 2005.

[203] L. Wang, Q.-K. Pan, P. N. Suganthan, W.-H. Wang, and Y.-M. Wang, "A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems," *Computers & Operations Research*, vol. 37 (3), pp. 509–520, 2010.

[204] R.-L. Wang and K. Okazaki, "An improved genetic algorithm with conditional genetic operators and its application to set-covering problem," *Soft Computing*, vol. 11(7), pp. 687–694, 2007.

[205] X. Wang and L. Tang, "A population-based variable neighborhood search for the single machine total weighted tardiness problem," *Computers and Operations Research archive*, vol. 36 (6), pp. 2105–2110, 2009.

[206] M. Watanabe, K. Ida, and M. Gen, "A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem," *Computers and Industrial Engineering*, vol. 48, pp. 743–752, 2005.

[207] W. Wisittipanich and V. Kachitvichyanukul, "Two enhanced differential evolution algorithms for job shop scheduling problems," *International Journal of Production Research*, vol. 50 (10), pp. 2757–2773, 2012.

[208] L.-N. Xing, Y.-W. Chen, and K.-W. Yang, "Multi-objective flexible job shop schedule: Design and evaluation by simulation modeling," *Applied Soft Computing*, vol. 9, pp. 362–376, 2009.

[209] W. Xu, R. Wang, L. Zhang, and X. Gu, "A multi-population cultural algorithm with adaptive diversity preservation and its application in ammonia synthesis process," *Neural Computing & Applications*, Online 2011.

[210] W. Xu, L. Zhang, and X. Gu, "A competitive co-evolutionary cultural differential evolution and its application to constrained optimization in butane alkylation process," in *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, Changsha, 2010, pp. 401 – 405.

[211] ——, "A novel cultural algorithm and its application to the constrained optimization in ammonia synthesis," *Communications in Computer and Information Science*, vol. 98(1), pp. 52–58, 2010.

[212] Z. Yamayee, K. Sidenblad, and M. Yoshimura, "A computationally efficient optimal maintenance scheduling method," *IEEE Transactions on Power Apparatus and Systems*, vol. 102(2), pp. 330–338, 1983.

[213] J.-H. Yang, L. Sun, H. Lee, Y. Qian, and Y.-C. Liang, "Clonal selection based memetic algorithm for job shop scheduling problems," *Journal of Bionic Engineering*, vol. 5(2), pp. 111–119, 2008.

[214] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178 (15), pp. 2985–2999, 2008.

[215] ——, "Multilevel cooperative coevolution for large scale optimization," in *IEEE Congress on Evolutionary Computation (CEC)*, 2008, pp. 1663–1670.

[216] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Transactions on Power Systems*, vol. 15(4), pp. 1232–1239, 2000.

[217] A. Yu and X. Gu, "Application of cultural algorithm to earliness/tardiness flow shop with uncertain processing time," in *Third International Conference on Natural Computation*, vol. 4, 2007, pp. 24–27.

[218] W.-j. Yu and J. Zhang, "Multi-population differential evolution with adaptive parameter control for global optimization," in *Genetic and Evolutionary Computation Conference (GECCO)*, Dublin, Ireland, 2011, pp. 1093–1098.

[219] W. Zangwill, "Nonlinear programming via penalty functions," *Management Science*, vol. 13(5), pp. 344–358, 1967.

[220] Z.-H. Zhan and J. Zhang, "Self-adaptive differential evolution based on pso learning strategy," in *Genetic and Evolutionary Computation Conference (GECCO)*, 2010, pp. 39–47.

[221] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, 2010.

[222] J. Q. Zhang and A. C. Sanderson, "Jade: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13 (5), pp. 945–958, 2009.

[223] R. Zhang and C. Wu, "A hybrid differential evolution and tree search algorithm for the job shop scheduling problem," *Mathematical Problems in Engineering*, vol. 2011, 2011, article ID 390593.

[224] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," Swiss Federal Institute of Technology (ETH) Zurich, Department of Electrical Engineering, Tech. Rep., 2001.

[225] G. I. Zobolas, C. D. Tarantilis, and G. Ioannou, "A hybrid evolutionary algorithm for the job shop scheduling problem," *Journal of the Operational Research Society*, vol. 60, pp. 221–235, 2009.

[226] H. Zurn and V. Quintana, "Generator maintenance scheduling via successive approximations dynamic programming," *IEEE Transactions on Power Apparatus and Systems*, vol. 94(2), pp. 665–671, 1975.

VITA AUCTORIS

NAME:	Mohammadrasool Raeesi Nafchi
PLACE OF BIRTH:	Esfahan, Iran
YEAR OF BIRTH:	1985
EDUCATION	Sharif University of Technology, Tehran, Iran B.Sc., 2003 - 2007
	University of Windsor, Windsor, ON, Canada M.Sc., 2008 - 2010
	University of Windsor, Windsor, ON, Canada Ph.D., 2010 - 2014