

2009

A framework for estimating the applicability of GAs for real-world optimization problems

Hsin-yi Jiang
Iowa State University

Follow this and additional works at: <https://lib.dr.iastate.edu/etd>

 Part of the [Computer Sciences Commons](#)

Recommended Citation

Jiang, Hsin-yi, "A framework for estimating the applicability of GAs for real-world optimization problems" (2009). *Graduate Theses and Dissertations*. 11084.

<https://lib.dr.iastate.edu/etd/11084>

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

**A framework for estimating the applicability of GAs for real-world optimization
problems**

by

Hsin-yi Jiang

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Carl K. Chang, Co-major Professor
Andrew S. Miner, Co-major Professor
Samik Basu
Daniel Tauritz
Johnny S. Wong
Dan Zhu

Iowa State University

Ames, Iowa

2009

Copyright © Hsin-yi Jiang, 2009. All rights reserved.

DEDICATION

I would like to dedicate this thesis to my grand parents, parents, and uncle Ming. Without their support, I would not have been able to complete this work. I would also like to thank my friends for their kind help during the writing of this work.

TABLE OF CONTENTS

LIST OF TABLES	vi
LIST OF FIGURES	vii
ACKNOWLEDGEMENTS	xi
ABSTRACT	xii
CHAPTER 1. INTRODUCTION	1
1.1 Overview of this Work	1
1.2 Problem Statement and Motivation of this Work	1
1.3 The Goal of this Work	3
CHAPTER 2. REVIEW OF LITERATURE	6
2.1 Real-World Optimization Problems	6
2.1.1 Problem Domain 1 - Project Management Problems	6
2.1.2 Problem Domain 2 - Software Engineering Problems	7
2.2 Analyzing the Behaviors of GAs	8
2.2.1 Markov Chain Based Approaches	8
2.2.2 Other Approaches	10
CHAPTER 3. BACKGROUND OF GENETIC ALGORITHMS	12
3.1 Optimization Problems and Fitness Functions of GAs	12
3.2 The Canonical Genetic Algorithm and Its Operators	12
3.3 An Overview of Markov Chain	13
3.3.1 Definitions and Theorems in GA, Markov Chain Theory, and Linear Algebra	14

3.3.2	Research Assumptions	18
3.3.3	Markov Chain Analysis for GAs	19
CHAPTER 4. CONVERGENCE RATE AND THE FIRST HITTING TIME		24
4.1	Convergence Rate	24
4.1.1	Eigenvalues of Transition Matrices and Fix Points of Genetic Algorithm	27
4.2	Second Largest Eigenvalue versus Expected First Hitting Time	29
CHAPTER 5. THE EVALUATION METRICS		30
5.1	The Bridge between Theory and Practice	30
5.1.1	Equivalent Forms in terms of Convergence	30
5.2	The Proposed Evaluation Metric	36
5.2.1	Overview of the Methodology	36
5.2.2	Evaluating the Applicability of GAs from Run Time Data	36
5.2.3	Theoretical Framework for Approximating the Applicability of CGAs .	37
5.2.4	Foundation of the Methodology	41
5.2.5	The Implementation and Data Structure of the Proposed Methodology .	43
5.2.6	Complexity Analysis	44
5.3	Experimental Validation of the Proposed Evaluation Metric	45
5.3.1	One Dimensional Fitness Functions	45
5.3.2	Discussion of the Sizes of Samples	45
5.3.3	Discussion of the Precision of the Estimation	48
5.3.4	Discussion of the Convergence Trend	50
5.3.5	Discussions on the Confidence Intervals	51
5.3.6	Discussions on the Smallest Number of Samples Needed for the Estimation	65
5.3.7	Multi-Dimensional Fitness Functions	67
CHAPTER 6. The Estimation of Global Convergence		71
6.1	The Overview of the Estimation	71
6.1.1	The Empirical Estimation on the Traces and the Fitting Curve	71
6.2	The Number of Generations for the Empirical Global Convergence	81

6.3	The Verification of the Proposed Estimation Framework	81
CHAPTER 7. A Case Study - Evolutionary Testing		89
7.1	Overview of Evolutionary Testing	89
7.2	The Flag Problem	92
7.3	Discussion	93
7.4	Experiment Settings	93
7.5	The Result of the Case Study	94
CHAPTER 8. The Generalization of the Proposed Methodology		99
CHAPTER 9. Conclusion and Future Work		102
9.1	Contributions of this Work	102
9.2	Limitations	103
9.3	Future Work	104
BIBLIOGRAPHY		105

LIST OF TABLES

Table 5.1	The Selected Fitness Functions	46
Table 5.2	The Estimated $Trace((CMS)^{n'})$ w.r.t. the Number of Generations and Power n'	46
Table 5.3	The Estimated $Trace((CMS)^{n'})$ With Respect To the Number of Generations and Power n'	47
Table 5.4	The Data of Figure 5.4(a) and 5.4(b)	53
Table 5.5	Comparison with Actual Trace Values	54
Table 5.6	The Range for Numbers of Generations at which the CGA with Best Solution Maintained Over Time Has Probability of 80% (out of 1000 Trials) to Reach the Global (or Near) Optimal Solution	65
Table 5.7	The Selected 3-Dimensional Fitness Functions and Their Features	67
Table 6.1	The Relationships among the Estimated Trace, the Proportion of the Visited States over the Total Number of States ($n' = 10$)	73
Table 6.2	The Relationships among the Estimated Trace, the Proportion of the Visited States over the Total Number of States ($n' = 12$)	74
Table 6.3	The Pair $(\hat{\lambda}_2, \hat{m}_2)$ with respect to Each Fitness Function	82
Table 6.4	The Bins Which First Reach 80% in CDFs	83
Table 6.5	The Empirical Global Convergence for Each Fitness Function	83
Table 7.1	The Fitness Functions of P_1 and P_2	95
Table 7.2	The Results of the Estimated $Trace((CMS)^{n'})$	96
Table 7.3	The Corresponding Information for Fitness Functions f_{14} and f_{15}	97

LIST OF FIGURES

Figure 1.1	General Methodology for Estimating the Applicability of GA \mathbf{G} for Optimization Problem \mathbf{P}	4
Figure 3.1	The Transition Matrix of the Random Walker in the Small Town	15
Figure 3.2	Structure of the Extended Transition Matrix PU	23
Figure 5.1	An illustration of the Product $PU \cdot PU$	32
Figure 5.2	The Estimation on the Trace of $(CMS)^{n'}$	39
Figure 5.3	A possible data structure for the proposed methodology	43
Figure 5.4	The relationship between estimated trace values, actual trace values, and numbers of generations with $m = 3, l = 3$	48
Figure 5.5	The relationship between estimated traces and numbers of generations	49
Figure 5.6	The convergence trends of $Trace((CMS)^{n'})$	51
Figure 5.7	The Distribution with respect to f_1	53
Figure 5.8	The Distribution with respect to f_2	55
Figure 5.9	The Distribution with respect to f_3	56
Figure 5.10	The Distribution with respect to f_4	57
Figure 5.11	The Distribution with respect to f_5	58
Figure 5.12	The Distribution with respect to f_6	59
Figure 5.13	The Distribution with respect to f_7	60
Figure 5.14	The Distribution with respect to f_8	61
Figure 5.15	95% SCI for f_1	62
Figure 5.16	95% SCI for f_2	62

Figure 5.17	95% SCI for f_3	62
Figure 5.18	95% SCI for f_4	62
Figure 5.19	95% SCI for f_5	62
Figure 5.20	95% SCI for f_6	62
Figure 5.21	95% SCI for f_7	62
Figure 5.22	95% SCI for f_8	62
Figure 5.23	The Comparison of 95% SCIs for the Estimated Trace Values Generated by the Proposed Evaluation Metric	63
Figure 5.24	95% WI for f_1	64
Figure 5.25	95% WI for f_2	64
Figure 5.26	95% WI for f_3	64
Figure 5.27	95% WI for f_4	64
Figure 5.28	95% WI for f_5	64
Figure 5.29	95% WI for f_6	64
Figure 5.30	95% WI for f_7	64
Figure 5.31	95% WI for f_8	64
Figure 5.32	The means of WI and SCI for fitness functions f_1 and f_8 , respectively	66
Figure 5.33	The order of fitness functions w.r.t. smaller numbers of generations . .	66
Figure 5.34	The convergence trends of $Trace((CMS)^{n'})$ for multi-dimensional fit- ness functions	68
Figure 6.1	The Overview of the Estimation	72
Figure 6.2	The relationship between the value and the setting of l	75
Figure 6.3	The relationship between the value and the setting of l	76
Figure 6.4	The relationship between the actual curve and the curve fitted by the pair $(n' = 10, n' = 12)$	77
Figure 6.5	The relationship between the actual curve and the curve fitted by the pair $(n' = 10, n' = 20)$	78

Figure 6.6	The relationship between the actual curve and the curve fitted by the pair ($n' = 20, n' = 22$)	78
Figure 6.7	The CDF for Number of Trials to Find the Global Optimal Solution with respect to f_1 and f_2 (X-Axis represents the number of generations; Y-Axis represents the cumulated number of trials to find the global optimal solution (out of 1000 trials))	85
Figure 6.8	The CDF for Number of Trials to Find the Global Optimal Solution with respect to f_3 and f_4 (X-Axis represents the number of generations; Y-Axis represents the cumulated number of trials to find the global optimal solution (out of 1000 trials))	86
Figure 6.9	The CDF for Number of Trials to Find the Global Optimal Solution with respect to f_5 and f_6 (X-Axis represents the number of generations; Y-Axis represents the cumulated number of trials to find the global optimal solution (out of 1000 trials))	87
Figure 6.10	The CDF for Number of Trials to Find the Global Optimal Solution with respect to f_7 and f_8 (X-Axis represents the number of generations; Y-Axis represents the cumulated number of trials to find the global optimal solution (out of 1000 trials))	88
Figure 7.1	Classification of Dynamic Structural Test Data Generation Techniques Using EAs [McMinn, P. (2004)]	91
Figure 7.2	An Example: The Source Code	92
Figure 7.3	Program with Flags	94
Figure 7.4	Program without Flags	94
Figure 7.5	The Control Flow Graph for P_1 and P_2	96
Figure 7.6	The PDF and CDF for Number of Trials to Find the Global Optimal Solution with respect to f_{14} (X-Axis represents the number of generations; Y-Axis represents the cumulated number of trials to find the global optimal solution (out of 1000 trials))	98

Figure 8.1 An Illustration of the Conversion 101

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Carl K. Chang and Dr. Andrew S. Miner for their guidance, patience and support throughout this research and the writing of this thesis. Their insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Samik Basu, Dr. Daniel Tauritz, Dr. Johnny S. Wong, and Dr. Dan Zhu. I would additionally like to thank Dr. Daniel Tauritz, and his students (Ekaterina Holdener and Travis Service) for their valuable comments on improving this dissertation.

ABSTRACT

Genetic Algorithms (GAs) have been gradually identified as an optimization-problem solver for certain classes of real-world applications. As GAs are increasingly utilized, a foundational study on how well GAs can perform with respect to varying problem domains becomes crucial. Yet, none of the prevalent theoretical studies are built upon the linkage between the theory and application of GAs. This dissertation introduces a methodology for estimating the applicability of a GA configuration for an arbitrary optimization problem based on run-time data. More specifically, this work analyzes the convergence behavior within a finite number of generations for each GA run through the estimation of the trace of the transition matrix of the corresponding Markov chain from run-time data. The analytical and empirical results show that the methodology is helpful for evaluating the applicability of GAs to optimization problems. Through the methodology, the number of generations needed for empirical convergence with respect to a fitness function (or a problem) can be estimated. The proposed methodology entails an evaluation metric and connects theory to application of GAs, for estimating the applicability of a GA to a problem. The methodology is demonstrated through a case study on evolutionary testing.

CHAPTER 1. INTRODUCTION

1.1 Overview of this Work

This dissertation develops a framework for estimating the applicability of a particular Genetic Algorithm (GA) configuration for an arbitrary optimization problem based on run-time data. GAs are increasingly employed to solve complex real-world optimization problems featuring ill-behaved search spaces (e.g., non-continuous, non-convex, non-differentiable) for which traditional algorithms fail. The quality of the optimal solution (i.e., the fitness value of the global optimum) is typically unknown in a real-world problem, making it hard to assess the absolute performance of an algorithm that is being applied to that problem. In other words, with a solution provided by a GA run, a method or theory to measure the quality of the solution is generally lacking. Although many researchers applying GAs have provided experimental results showing their successful applications, those are merely averaged-out, *ad hoc* results. They cannot represent nor guarantee the usability of the best solutions obtained from a single GA run, since the solutions can be very different for each run. Therefore, it is desirable to provide a formalized measurement to estimate the applicability of GAs to real-world problems.

1.2 Problem Statement and Motivation of this Work

In many real-world optimization problem domains, researchers have gradually found that GAs possess the properties suitable for certain classes of applications. Many researchers have reported success in applying GAs to real-world problems, but have failed to provide a theoretical foundation to explore or explain why GAs were appropriate for the specific problems. As GAs are increasingly utilized, a foundational study on how well GAs can perform for each of

the various problem domains becomes crucial. Note that many theoretical studies have investigated the behaviors of GAs using Markov chains [Eiben, A. E. and Aarts, E. H. L. and Hee, K. M. V. (1991); Nix, A. E. and Vose, M. D. (1992); Ding, L. and Yu, J. (2005); Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007); Fogel, D. B. (1995); Rudolph, G. (1994); Suzuki, J. (1995, 1998); Vose, M. D. and Liepins, G. E. (1991); Rudolph, G. (1996); Davis, T. E. and Principe, J. C. (1993)]. Yet, no existing theoretical studies are built upon the linkage between the theory and application of GAs. Through existing analyses, the convergence of a canonical form of Genetic Algorithm termed CGA with best solutions maintained has been proven [Rudolph, G. (1994)], the expected value of the first hitting time of GAs has been calculated [Ding, L. and Yu, J. (2005)], and the convergence rate of GAs has been predicted by the second largest eigenvalue of the transition matrix [Suzuki, J. (1995); Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007)]. Other methods, such as the analysis on Walsh transformation [Bethke, A. D. (1980); Forrest, S. and Mitchell, M. (1993)] or Fourier transformation of fitness functions of GAs [Kosters, W. A. and Kok, J. N. and Leiden, P. F. (1999)], have also been adopted by some researchers. Although many papers have proposed various approaches to analyzing the behaviors of GAs, most existing GA theories lack concern for practicality [Schoenauer, M. et al. (2007); Jiang, H. and Chang, C. K. (2008); Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007)]. For instance, the convergence of GAs studied by Rudolph is only meant as a theoretical study, since time is assumed to go to infinity [Rudolph, G. (1994)]. In reality, it is impossible for any application to wait for an infinite amount of time to obtain the optimal solution. Moreover, the computation time for deriving transition matrices of GAs with respect to real-world problems is much larger than calculating fitness values of the entire search space (i.e., the state space). It is not practical because researchers will not waste more time to obtain weaker solutions. In general, “theoretical studies of GAs are criticized for rarely being applicable to the real-world [Schoenauer, M. et al. (2007)],” and applications of GAs to real-world problems are frequently studied without foundational support [Jiang, H. and Chang, C. K. (2008); Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007)]. In application domains of GAs, a more practical and

functionally equivalent approach to evaluate the applicability of GAs is desired in view of the current state of the art.

1.3 The Goal of this Work

This work aims to develop a practical support for researchers and practitioners to evaluate the applicability of GAs to their problem domains. We have proven that the convergence rate of Markov transition matrices with respect to encodings of optimization problems is related to the second largest eigenvalue of the transition matrices in absolute value with its physical meaning explained. The second largest eigenvalue can bound the expected value of the first hitting time of the optimal solutions corresponding to those optimization problems [Ding, L. and Yu, J. (2005); Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007)]; however, the computation time of constructing the transition matrix with respect to an optimization problem takes more than the computation time of calculating all the feasible solutions [Jiang, H. and Chang, C. K. (2008)]. Relying on the second largest eigenvalue of the transition matrix with respect to a problem to acquire convergence degrees of GAs is impractical due to the long computation time. With a concern for practicality, we propose a novel method, an important approach for real-world applications, to derive the applicability of a GA to a problem based on the approximate sum of eigenvalues. Mathematically, the sum of eigenvalues of a matrix is equal to the sum of the diagonal elements (which is called “trace”) of the matrix. According to that property, our methodology is developed to estimate the trace of the corresponding transition matrix. Through this method, a degree of convergence can be determined for each GA run. Being aware of the degree of convergence, researchers and practitioners will be able to obtain certain information about the applicability of GAs and know how good the solutions generated by GAs are, so that correct decisions can be made. Moreover, a possible approach for estimating the number of generations needed for global convergence is also proposed. The general methodology is illustrated in Figure 1.1. In summary, this dissertation makes the following contributions to theory and application of GAs:

- a problem statement for the current state of the art;

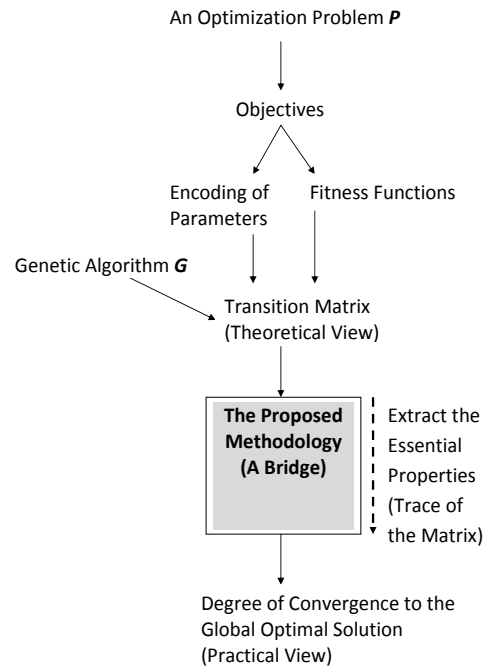


Figure 1.1 General Methodology for Estimating the Applicability of GA \mathbf{G} for Optimization Problem \mathbf{P}

- a novel and possible approach to build a linkage between theory and application of GAs;
- the verification of the proposed approach.

The rest of this dissertation is organized as follows. Chapter 2 provides the literature review of this work. Chapter 3 reviews the basic operators of GAs, Markov chains, and how to model the operators of GAs using Markov chains. Research assumptions and definitions are also presented in this chapter. Chapter 4 proves that the convergence rate of a CGA (see Section 3.3.3.2) with best solution maintained over time depends on the second largest eigenvalue of the corresponding transition matrix, and explains the relationship between the second largest eigenvalue of the transition matrix, and the first hitting time (i.e., expected waiting time) of the optimal solution of a GA. Chapter 5 explores an evaluation matrix to evaluate the applicability of GAs to real world optimization problems. In Chapter 6, a case study about an evolutionary algorithm (EA) used in software testing (also called evolutionary testing) is provided. Chapter

7 proposes a possible approach for estimating the number of generations needed for the global convergence. Chapter 8 proves that the proposed metric can be generalized to support certain classes of EAs. Chapter 9 concludes the dissertation and outlines future research work.

CHAPTER 2. REVIEW OF LITERATURE

2.1 Real-World Optimization Problems

Generally, if a problem has more than one feasible solution, the problem can be considered as an optimization problem. Theoretically, the optimization problem is defined as the problem which can be solved by more than one feasible solution and has at least one criterion to evaluate solutions, with the goal being the search for the best solution within the domain of all the feasible solutions.

In the real-world, there are many optimization problems in which GAs are applicable. Among many fields of study, such as combinatorial optimization problem domains, machine learning, information retrieval, and data mining, project management and software engineering are two heavily studied fields that provide ample opportunities to apply GAs for performance improvement.

I have research interests in both Project Management (PM) and Software Engineering (SE). Some problems in those two domains are listed below.

2.1.1 Problem Domain 1 - Project Management Problems

In PM, optimal scheduling is one of the typical optimization problems [Chang, C. K. and Christensen, M. J. (1999); Chang, C. K. and Christensen, M. J. and Zhang, T. (2001); Chao, C. (1995); Ge, Y. (2004)]. One branch in PM research explores ways to intelligently match employees to tasks with respect to the factors gathered during early project development. Typical factors include salaries of employees (costs), capabilities of employees, learning curves of employees, and potential hazard levels of assignments [Xu, R. and Qian, L. and Jing, X. (2003)]. Researchers may expand or narrow this list of factors depending upon their research

goals.

2.1.2 Problem Domain 2 - Software Engineering Problems

Software Testing Problems - In SE, the traditional waterfall model regards testing as a key component of verification and validation (V&V) activities [Pressman, R. S. (2005)]. Verification is to inspect whether specific functions are built correctly in the software, and validation is to examine whether the software meets customer requirements. To conduct testing on a piece of software, various strategies have been proposed. Some strategies are formulated as optimization problems [Berndt, D. and Fisher, J. and Johnson, L. and Pinglikar, J. and Watkins, A. (2003); Berndt, D. J. and Watkins, A. (2004); Briand, L. C. and Labiche, Y. and Shousha, M. (2004, 2005); Clark, J. and Dolado, J. J. and Harman, M. and Hierons, R. M. and Jones, B. and Lumkin, M. and Mitchell, B. and Mancoridis, S. and Rees, K. and Roper, M. and Shepperd, M. (2003); Vieira, F. E. and Menezes, R. and Braga, M. (2006)]. For instance, with regard to branch testing in a structural testing problem [Clark, J. and Dolado, J. J. and Harman, M. and Hierons, R. M. and Jones, B. and Lumkin, M. and Mitchell, B. and Mancoridis, S. and Rees, K. and Roper, M. and Shepperd, M. (2003)], the condition statements of the test-aim branch are formulated into a fitness function, which guides the search of input data to satisfy all the conditions of the test-aim branch. That is, researchers measure the “distance” between the test-aim branch and the branch caused by a set of input data. GAs minimize the distance so that the input data (called test cases) of a test aim can be generated.

Software Module Clustering Problems - Owing to the rapid development of computer technologies, industrial applications are increasingly equipped with highly complex software systems, which often consist of a large number of components. To streamline the design phase, a large-scale system can be designed hierarchically. Instead of directly integrating all of the components into a system, a set of congruent components are first grouped into a subsystem. How to cluster the original set of components into several subsystems has attracted the attention of researchers for many years [Chang, C. K. and Cleland-Haung, J. and Hua, S.

and Kuntzmann-Combelles, A. (2001)]. Oftentimes, they try to maximize cohesion inside a component and minimize coupling among components. This kind of problem also represents a class of optimization problems in SE [Mitchell, B. S. and Mancoridis, S. (2002)].

2.2 Analyzing the Behaviors of GAs

2.2.1 Markov Chain Based Approaches

A variety of methods have been employed to analyze the behaviors of GAs [Rudolph, G. (1994); DeJong, K. A. and Spears, W. M. and Gordon, D. F. (1995); Suzuki, J. (1995); He, J. and Kang, L. (1999); He, J. and Yao, X. (2001); Ding, L. and Yu, J. (2005); Coley, D. A. (1999); Mitchell, M. (1996); Goldberg, D. E. (1989); Grefenstette, J. J. (1992); Bethke, A. D. (1980); Forrest, S. and Mitchell, M. (1993); Bridges, C. L. and Goldberg, D. E. (1991); Naudts, B. and Kallel, L. (2000)]. Among them, different approaches are derived and asserted. While each method has its own merits, Markov chain analysis can be most successfully applied to capture the essential spirit of GAs due to the following reasons:

- The initial population of a GA run is based on a probability distribution. Usually, it is a uniform distribution.
- From one generation to another, the GA selects the individuals based on the proportions of the fitness values of the individuals in the current population. In other words, the selection operator selects the individuals from (and only from) the current state where each individual has some probability of being preserved in the next generation. Note that the fitness function of the GA run is considered in the selection operation.
- Regardless of the type of recombination employed to solve the optimization problem, the recombination and mutation operators are both related to probability issues.

My research adopts Markov chain analysis to investigate the behaviors of GAs. Previous work by Rudolph [Rudolph, G. (1994)], De Jong [DeJong, K. A. and Spears, W. M. and Gordon, D. F. (1995)], Suzuki [Suzuki, J. (1995)], He [He, J. and Kang, L. (1999); He, J.

and Yao, X. (2001)], and Ding [Ding, L. and Yu, J. (2005)], including their methodologies and results, are particularly relevant to my approach.

Rudolph’s main contribution on the behavior of GAs is to prove that the CGA with best solution maintained converges to its global optimal solution [Rudolph, G. (1994)]. Additionally, he mentions that in an ergodic Markov chain, the expected value of the transient time in which an arbitrary state i goes to any other state j is finite. This claim is a well established result in Markov chain theory. The detailed proof can be seen in [Iosifescu, M. (1980) (p. 133)]. This implies that the expected value of the transient time in which the initial state moves to the optimal state is finite, since both the initial state and the optimal state are the states in Markov chain.

Later, De Jong et al. proposed a method to capture the “hardness” of a GA (i.e., the level of difficulty to apply it) by computing the expected waiting time (i.e., the first hitting time) through the use of transition matrices [DeJong, K. A. and Spears, W. M. and Gordon, D. F. (1995)]. To my knowledge, this was the first attempt to use the first hitting time to predict the applicability of GAs. In 2005, the expected first hitting time of the optimal state was investigated and calculated again by Ding et al. [Ding, L. and Yu, J. (2005)]. They proposed an approach to reformulate the transition matrices so that the formula to derive the expected value of the first hitting time became simpler than before. Besides the computation of the expected waiting time, De Jong et al. introduced the concept of predicting the behaviors of GAs within a fixed number of generations based on the derived transition matrices [DeJong, K. A. and Spears, W. M. and Gordon, D. F. (1995)]. This concept provides an insight for the waiting time in practical use, which is similar to Assumption 3 in Chapter 3 of this dissertation.

In Suzuki’s work, the investigation of behaviors of GAs was first linked to the second largest eigenvalue of the transition matrix and convergence rates of Markov chains [Suzuki, J. (1995)]. My preliminary result, which is presented in Chapter 4 and derived from a different approach in Markov chain theory, matches this assertion in some sense. In addition to the eigenvalue-based approach, He and Kang proposed another perspective to bound the convergence rate through the “minorization condition” in Markov chain theory [He, J. and Kang, L. (1999)].

He and Yao devoted their work to calculating the computational time complexity of evolutionary algorithms (EAs) [He, J. and Yao, X. (2001)]. Their work generalized Droste’s work, which provided a rigorous complexity analysis of the $(1 + 1)$ EA (i.e., EA with a population size of 1 and only with mutations), for a class of fitness functions [Droste, S. and Jansen, T. and Wegener, I. (1998)]. With He and Yao’s approach, Droste’s work was extended to general EAs through drift analysis. Several drift conditions were studied for deriving the computational time. Their work began with modeling the evolution of an EA population as a random sequence, e.g., a Markov chain. The general case was considered with three operators (selection, crossover, mutation) and a population with multiple individuals. Then, they analyzed the relationship between the drift of the sequence and the optimal solution of the problem. Various bounds on the first hitting time were derived under different drift conditions. They also asserted that some drift conditions caused the random sequence to drift away from the optimal solution, while others enabled the sequence to drift towards the optimal solution. The conditions used to determine the time complexity of an EA to solve a problem were also investigated and proposed [He, J. and Yao, X. (2001)].

2.2.2 Other Approaches

Starting from the early 1970s, a series of approaches different from Markov chain analysis were proposed to analyze the behaviors of GAs. In 1975, Holland introduced the notion of schemas to formalize the informal notion of “building blocks” [Coley, D. A. (1999) Mitchell, M. (1996)]. His building block hypothesis stated that GAs attempt to find highly fit solutions to a problem through short, low-order, and above-average schemata. However, the schema theory merely demonstrated a rough idea in high level that better performing schemata will receive an increasing number of trials in the next generation. It does not give us much information about the detailed analysis of the behaviors of GAs. In 1987, the term “deception problem” was coined by Goldberg, who said that a problem is deceptive if certain hyperplanes guide the search toward some solutions or genetic building blocks that are not globally competitive [Goldberg, D. E. (1989)]. Comparing Goldberg’s statement with Holland’s building block hypothesis,

the deception problem appears to suggest a contradiction. Hence, deception problems are considered “hard” problems for GAs. Nevertheless, the deception is neither a sufficient nor necessary condition to characterize problems that are hard for GAs [Grefenstette, J. J. (1992)].

Goldberg isolated the deception problems from Bethke’s work [Bethke, A. D. (1980); Forrest, S. and Mitchell, M. (1993)]. Bethke used discrete Walsh functions to analyze the fitness functions of GAs. He developed the Walsh-Schema transform to calculate the average fitness of schema efficiently and used it to characterize functions as easy or hard for GAs to optimize. This method helps to produce ideas for solving a problem; however, it can be difficult to convert functions to Walsh polynomials. This method, a static analysis, examines only a flat population, where every possible string is assumed to be represented in equal proportion [Bridges, C. L. and Goldberg, D. E. (1991)]. Bethke’s method fails to capture the more dynamic aspects found in GAs. To address this issue, Bridges and Goldberg proposed another approach called the Nonuniform Walsh Transform [Bridges, C. L. and Goldberg, D. E. (1991)]. Unfortunately, as in the case of Walsh transformations, it is difficult to convert fitness functions into such forms.

In 2000, Naudts and Kallel studied two widely-known predictive measures of problem difficulty in GAs (with both the GA-easy and GA-hard functions): epistasis variance and fitness distance correlation [Naudts, B. and Kallel, L. (2000)]. They found that the values of the measures can be completely unreliable and entirely uncorrelated to the convergence quality and speed of GAs.

In general, the GA theory developed thus far shows that it is difficult to fully capture the behaviors of GAs, especially in finite time with different types of fitness functions. The missing link is the bridge between the theory and its applicability to practical problems [Jiang, H. and Chang, C. K. (2008)]. This observation coincides well with the recent claims stated in the front pages of the journal [Schoenauer, M. et al. (2007)] concerning the difficulty in directly linking GA theory to real-world applications. In this study, the objective is to develop the necessary support theory to effectively bridge the gap between evolutionary computation and real-world applications.

CHAPTER 3. BACKGROUND OF GENETIC ALGORITHMS

This chapter reviews and presents the background information for GAs, Markov chains, how to model GAs using Markov chains, existing theorems, and research assumptions of my work.

3.1 Optimization Problems and Fitness Functions of GAs

As mentioned before, an optimization problem is defined as the problem which can be solved by more than one feasible solution, has at least one criterion to evaluate solutions, with the goal of the problem being the search for the best solution within the domain of all the feasible solutions. The (feasible) solutions of GAs are encoded into strings, usually called chromosomes or individuals. With a GA being chosen as the optimization method, the criteria to evaluate solutions are formulated as a fitness function for the GA.

3.2 The Canonical Genetic Algorithm and Its Operators

The CGA (also called a simple GA) can be sketched as follows [Rudolph, G. (1994)].

Choose an initial population (i.e., a list of a fixed number of individuals)

Compute the fitness of each individual

Perform selection

Repeat

 Perform crossover

 Perform mutation

 Compute the fitness of each individual

 Perform selection

Until stopping criterion is satisfied

In other words, it is composed of three operators:

- Selection (also called Reproduction)
- Crossover
- Mutation

Selection is a process in which individuals are copied according to their fitness values. Usually, the individuals with higher fitness values have higher probabilities to be selected into the next generation. Therefore, a typical fitness function for a selection operator should be the function to be maximized. This operator is actually an artificial version of natural selection, the Darwinian theory of “survival of the fittest” [Goldberg, D. E. (1989)]. Various selection methods, such as roulette wheel selection (proportional selection), tournament selection, and (μ, λ) selection, etc., are proposed. Among them, roulette wheel selection is commonly adopted in the literature. It selects individuals based on their proportions of the fitness values among the individuals in the current population (generation). For demonstration purposes, this dissertation mainly discusses roulette wheel selection. In Chapter 8, the selection operator is generalized to any type of selection methods.

Crossover, including one point crossover, two point crossover, and uniform crossover, etc., mimics the mating of creatures. It swaps some bits of two chosen individuals. The resulting individuals are passed into the next generation.

The mutation operator simulates biological mutation, maintaining genetic diversity from one generation to another. A simple and common way to implement it is to sweep each individual bit of strings within a population once, and each bit has a fixed probability to be flipped to another number.

3.3 An Overview of Markov Chain

Markov chains are named after Prof. Andrei A. Markov (1856-1922) who first published his result in 1906 [Ching, W. K. and Ng, M. K. (2006)]. His research work on Markov

chains launched the study of stochastic processes, which was followed by a large variety of applications. In this work, the discrete homogeneous finite state Markov chain is applied. Specifically, a Markov chain is a sequence of random variables X_1, X_2, X_3, \dots with the Markov property [Iosifescu, M. (1980)]. The Markov property is the property that:

$$Pr(X_{n+1} = x | X_n = x_n, \dots, X_1 = x_1) = Pr(X_{n+1} = x | X_n = x_n), \forall n \in N. \quad (3.1)$$

The future state is only dependent on the current state and independent of the past states.

A Markov chain is called *time-homogeneous* if:

$$Pr(X_{n+1} = x | X_n = y) = Pr(X_n = x | X_{n-1} = y), \forall n \in N. \quad (3.2)$$

To form a Markov chain, three basic components should be considered: a state space, an initial distribution, and a transition matrix.

Consider a *random walker* in a small town. Within the town, there are a finite number of places to go. Suppose that at time t , $t \in N$, the random walker stands in a place in the town. At time $t+1$, he walks to any place in the town with a certain probability, dependent only upon the place he was in at time t (i.e., time is a non-factor). If each place in the town is assigned a distinct number (as an index), say, $1, 2, \dots, k$ ($k \in N$), and at each time t , X_t is a random variable denoting the index of the random walker's location, (X_0, X_1, \dots) is a random process taking values in $\{1, 2, \dots, k\}$. Such a random process can be treated as a discrete homogeneous finite state Markov chain. Figure 3.1 illustrates the transition matrix of a Markov chain for the random walker in the small town. In Figure 3.1, places are considered as states of the Markov chain. If the current time is t , '*' represents the probability of transitioning from state i to state j ($i, j \in \{1, \dots, k\}$) at time $t+1$.

3.3.1 Definitions and Theorems in GA, Markov Chain Theory, and Linear Algebra

Definition 1. A nonnegative square matrix A is said to be stochastic if and only if the sum of the entries in any row of A is 1.

		<i>To</i>				
		<i>j</i>				
		Place 1	Place 2	Place k	
<i>From</i>	Place 1	*	*	*	
	Place 2	*	*	*	
	Place 3	*	*	*	
	Place 4	*	*	*	
<i>i</i>	.					
	.					
	.					
		Place k	*	*	*

Figure 3.1 The Transition Matrix of the Random Walker in the Small Town

Definition 2. A square matrix $A_{r \times r}$ is called positive (i.e., $A > 0$) if and only if $a_{ij} > 0 \forall i, j \in \{1, 2, \dots, r\}$.

Definition 3. A stochastic matrix A is said to be regular if and only if there exists a natural number r such that A^r is positive (i.e., $A^r > 0$).

Note that the product of stochastic matrices is a stochastic matrix.

Definition 4. A state in a Markov transition matrix is called transient if there is a non-zero probability that once the chain leaves that state, it will never return.

Definition 5. A state in a Markov transition matrix is called absorbing if once the chain enters that state, it never leaves.

Definition 6. The trace of a square matrix $A_{r \times r}$ is defined to be the sum of the elements on the main diagonal of A , i.e.,

$$\text{Trace}(A) = a_{11} + a_{22} + \dots + a_{rr}.$$

Definition 7. [Burden, R. L. and Faires, J. D. (2005)] [Convergence Rate] Suppose $\{\beta_n\}_{n=1}^{\infty}$

is a sequence known to converge to zero, and $\{\alpha_n\}_{n=1}^{\infty}$ converges to a number α . Then it is called that $\{\alpha_n\}_{n=1}^{\infty}$ converges to α with convergence rate (or rate of convergence) $O(\beta_n)$ if there exist a constant K and a number $M' > 0$ such that

$$|\alpha_n - \alpha| \leq K|\beta_n|, \text{ for all } n \geq M'. \quad (3.3)$$

In addition, Inequality (3.3) indicates that $\alpha_n = \alpha + O(\beta_n)$ for large n .

Definition 8. [*Empirical Convergence*] Let P be a square matrix of order r and $\{P^n\}_{n \in \mathbb{N}}$ converges to P^* as $n \rightarrow \infty$. In other words, for all $\epsilon > 0$, there exists a constant $n_1(\epsilon)$ such that

$$\left(\sum_{i,j=1}^r |p_{ij}^{(n)} - p_{ij}^*|^2 \right)^{1/2} < \epsilon, \forall n \geq n_1(\epsilon),$$

where $p_{ij}^{(n)}$ is the element in the i^{th} row and j^{th} column of P^n , and p_{ij}^* is the element in the i^{th} row and j^{th} column of P^* . Then $\{P^n\}_{n \in \mathbb{N}}$ is called to empirically converge to P^* with respect to a given $\epsilon > 0$ at $n = k$ if $k \geq n_1(\epsilon)$.

Theorem 1 [*Rudolph, G. (1994)*] *The transition matrix of the CGA with mutation probability $p_m \in (0, 1)$, crossover probability $p_c \in [0, 1]$ and fitness proportional survivor selection is regular.*

Theorem 1, formulated by Rudolph, is used to prove the convergence of CGA, with best solution maintained, to its global optimal solution [Rudolph, G. (1994)].

Theorem 2 [*Iosifescu, M. (1980)*] *If P is a $r \times r$ regular stochastic matrix, then P^n converges as $n \rightarrow \infty$ to a positive stable stochastic matrix $\Pi = ev^T$, where $e = (1, 1, \dots, 1)^T$ is a $r \times 1$ column vector in which all elements are of the value 1, and $v^T = (v_1, v_2, \dots, v_r)$ is a $1 \times r$ probability row vector with non-null entries. Moreover, there exists a constant $a > 0$ such that*

$$|p_{ij}^{(n)} - v_j| \leq an^{m_2-1}|\lambda_{2-P}|^n, \quad (3.4)$$

where $p_{ij}^{(n)}$ is the i^{th} row and j^{th} column of P^n , λ_{2-P} is the second largest eigenvalue of P in absolute value, and m_2 is the (algebraic) multiplicity of λ_{2-P} .

Theorem 3 [Iosifescu, M. (1980)] Let the transition matrix P be

$$P = \begin{pmatrix} P_1 & 0 \\ R & A \end{pmatrix},$$

where P_1 is regular, $R \neq 0$. Then

$$P^n = \begin{pmatrix} P_1^n & 0 \\ \sum_{i=0}^{n-1} A^{n-1-i} R P_1^i & A^n \end{pmatrix}. \quad (3.5)$$

As $n \rightarrow \infty$, P^n converges to

$$\lim_{n \rightarrow \infty} P^n = \begin{pmatrix} e_1 v^T & 0 \\ e_2 v^T & 0 \end{pmatrix}, \quad (3.6)$$

where $v^T = (v_1, v_2, \dots, v_{r_1})$ is a $1 \times r_1$ probability row vector with non-null entries, $e_1 = (1, 1, \dots, 1)^T$ is a $r_1 \times 1$ column vector if r_1 is the number of rows in P_1 , and $e_2 = (1, 1, \dots, 1)^T$ is a $r_2 \times 1$ column vector if r_2 is the number of rows in R .

Note that A^n converges to 0 as $n \rightarrow \infty$. Moreover, $\lim_{n \rightarrow \infty} P^n$ has the property that each of the columns has the same entry value. This is insightful for the proof that the initial state will not impact the final state if time goes to infinity.

Theorem 4 [Iosifescu, M. (1980)][Perron's Formula] If A is a square matrix of order r , $\lambda_1, \lambda_2, \dots, \lambda_q$, $q \leq r$, are the eigenvalues of A , and m_1, m_2, \dots, m_q are the (algebraic) multiplicities of the eigenvalues, respectively, $m_1 + m_2 + \dots + m_q = r$, then

$$a_{ij}^{(n)} = \sum_{k=1}^q \frac{1}{(m_k - 1)!} \left(\frac{d^{m_k-1}}{d\lambda^{m_k-1}} \left(\frac{\lambda^n A_{ij}(\lambda)}{\prod_{i \neq k} (\lambda - \lambda_i)^{m_i}} \right) \right)_{\lambda=\lambda_k},$$

where $a_{ij}^{(n)}$ is the element in the i^{th} row and j^{th} column of A^n , and $A_{ij}(\lambda)$ is the element in the i^{th} row and j^{th} column of the adjoint of the matrix $(\lambda I_r - A)$; that is, $A_{ij}(\lambda)$ is equal to the product of $(-1)^{i+j}$ and determinant of the minor of $(\lambda I_r - A)_{ji}$. Note that the minor of $(\lambda I_r - A)_{ji}$ is derived by deleting the j^{th} row and i^{th} column from the matrix $(\lambda I_r - A)$, and I_r is the identity matrix of order r .

Theorem 5 [Iosifescu, M. (1980)] If A is a regular matrix, then there exists a real eigenvalue $\lambda_1 > 0$ which is simple (i.e., of algebraic multiplicity 1) and which exceeds the absolute values of all other eigenvalues of A .

Lemma 1 [Iosifescu, M. (1980)] *If A is a stochastic matrix (i.e., A is nonnegative and the sum of the elements in any row of A is 1), then the eigenvalues of A are in absolute value at most equal to 1. Moreover, 1 is an eigenvalue of A .*

Theorem 6 [Horn, R. A. and Johnson, C. R. (1985)] [Schur] *If A is a square matrix of order r with eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_r$ in any prescribed order, there exists a unitary matrix S (square matrix) of order r such that*

$$S^*AS = T = [t_{ij}]$$

is upper triangular, with diagonal entries $t_{ii} = \lambda_i$, $i = 1, 2, \dots, r$. That is, every square matrix A is unitarily equivalent to a triangular matrix whose diagonal entries are the eigenvalues of A in a prescribed order.

Theorem 2, 3, 4, 5, 6, and Lemma 1 are directly from Markov chain theory and linear algebra. Theorem 4, 5, 6, and Lemma 1 are utilized to deduce my preliminary result [Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007)].

3.3.2 Research Assumptions

The first two assumptions adopted in this research are based on Rudolph's work [Rudolph, G. (1994)].

Assumption 1 [Problem Definition] *The problems $\max\{f(b)|b \in IB^l\}$, where $0 < f(b) < \infty$ for all $b \in IB^l = \{0,1\}^l$, and l is the length of the binary strings which represent feasible solutions, are the subjects for discussion.*

Assumption 2 [Choice of Solution Method] *The CGA, which only has selection, crossover, and mutation operators, with the best solution maintained, is the algorithm to be analyzed. (More specifically, roulette wheel selection, any crossover operation, and bit mutation are considered before Chapter 8 in this dissertation.)*

The last assumption pertains to the concerns for practicality and applicability.

Assumption 3 *The number of generations of GAs is a reasonably large number, and it is fixed.*

3.3.3 Markov Chain Analysis for GAs

3.3.3.1 State Representations

Two types of state representations for the finite state homogeneous Markov transition matrices are commonly adopted in Markov chain analysis. Michael D. Vose, Joe Suzuki, et al. applied the transition matrix with the states representing the occurrences of the individuals [Suzuki, J. (1995, 1998); Vose, M. D. and Liepins, G. E. (1991)]. The cardinality of different populations (i.e., the dimension of the state space), becomes [Suzuki, J. (1995)]

$$|S| = \binom{m + 2^l - 1}{m},$$

where S is the state set, m is the size of a population, and l is the length of the binary strings.

Günter Rudolph and David B. Fogel [Fogel, D. B. (1995); Rudolph, G. (1994)] proposed the transition matrix in which the states are defined by every possible configuration of an entire population of bit strings. Therefore, there are 2^{ml} states, where m is the size of a population, and l is the length of the binary strings.

Although the approaches of both Vose and Rudolph possess similar concepts and functionalities, and they can be converted to each other (see Theorem 11), the representations are different. Each approach has its advantages. While fewer states and the distributions of the individuals can be obtained in Vose’s approach, the analysis of this approach is not as intuitive as Rudolph’s approach. For instance, if there are four individuals (3-digit binary strings) within a generation, say “101, 111, 011, 111”, this generation is represented by the state “101111011111” in Rudolph’s approach. For Vose’s approach, one has to list the sorted state space “000, 001, 010, 011, 100, 101, 110, 111” first, and obtain the state “111011011100”. Since my research goal is to seek an evaluation metric to estimate the applicability of GAs for real-world applications and Markov chain analysis is not so practical (discussed in Section VI), either approach can be chosen as the transient analyzing method. Due to the fact

that Rudolph's approach is more intuitive than Vose's, my work uses Rudolph's approach to transform the encodings of optimization problems to Markov chains [Rudolph, G. (1994)]. Rudolph's work studies the convergence of CGAs with the best solution maintained. The transition matrices of his Markov chains are described below.

3.3.3.2 Transition Matrices

A CGA is a *canonical* GA which consists of an m -tuple of binary strings $b_i, i \in \{1, 2, \dots, m\}$, of length l . The bits of each string are considered to be the genes of an individual chromosome. The m -tuple of individual chromosomes is said to be a population of a generation. From one generation to another, CGA applies three operators on the population. The operators include selection, crossover, and mutation operators.

The selection operator, in which the roulette wheel selection (proportional selection) is assumed, forms a transition matrix S . As mentioned before, there are m (an even number) individuals for each generation. By Assumption 1, S is a $2^{ml} \times 2^{ml}$ matrix with the element

$$s_{ij} = \frac{\prod_{k=1}^m O_k \cdot f(\pi_k(j))}{(\sum_{k=1}^m f(\pi_k(i)))^m}$$

if $\{\pi_1(j), \pi_2(j), \dots, \pi_m(j)\} \subseteq \{\pi_1(i), \pi_2(i), \dots, \pi_m(i)\}$, where $\pi_k(i), k \in \{1, \dots, m\}$ are the k^{th} segment of length l from the state i , O_k is the number of occurrences of $\pi_k(j)$ in state i , and $f(\cdot)$ is the fitness function. Otherwise, $s_{ij} = 0$.

By the same token, the crossover transition matrix C is also a $2^{ml} \times 2^{ml}$ matrix. After the crossover method is determined, I_i is defined to be the index set in which each element is a binary string with length ml representing a possible mating method for the individuals of state i . For each $r \in I_i$, let p_r be the probability of r being selected as a mating method. If p_c is the crossover probability, then

$$c_{ij} = (1 - p_c)\delta_{ij} + p_c \cdot \left(\sum_{r \in I_i} p_r \cdot \prod_{k=1}^{\frac{m}{2}} Pr\{Cr(\pi_{2k-1}(r), \pi_{2k}(r)) = (\pi_{2k-1}(j), \pi_{2k}(j))\} \right), \quad (3.7)$$

where

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases},$$

$Cr(\cdot, \cdot)$ is the result of the crossover operation, and $Pr\{\cdot\}$ is the probability of the event. In (3.7),

$$Pr\{Cr(\pi_{2k-1}(r), \pi_{2k}(r)) = (\pi_{2k-1}(j), \pi_{2k}(j))\}$$

depends on the crossover method applied.

Bit mutation is designed to serve as a background operator to ensure that all possible alleles can occur in the population [Fogel, D. B. (1995)]. Once the population of chromosomes reaches a configuration such that crossover no longer produces offspring outperforming their parents, it is the only operator which leads the population to leap out of the homogeneous populations. The mutation transition matrix is denoted as M , which is a $2^{ml} \times 2^{ml}$ matrix. Let $p_m \in (0, 1)$ be the probability of flipping each individual bit, then

$$m_{ij} = \prod_{k=1}^m p_m^{H(\pi_k(i), \pi_k(j))} (1 - p_m)^{l - H(\pi_k(i), \pi_k(j))},$$

where $H(\cdot, \cdot)$ is the Hamming distance of the strings (chromosomes). The matrix CMS , the product of C , M , and S , forms a transition matrix for CGA.

In order to show that the CGA with the best solution maintained converges to its global optimum, the state space is extended from 2^{ml} to $2^{(m+1)l}$. That is, for each state (m individuals), there is a referenced individual (assumed to be the leftmost individual) with it. The referenced individual is for the calculation with respect to the presence of the best solution. Since there are 2^l distinct referenced individuals, the new state space is $2^{ml} \cdot 2^l = 2^{(m+1)l}$. The new transition matrix P is

$$P = \begin{pmatrix} CMS & & & \\ & CMS & & \\ & & \ddots & \\ & & & CMS \end{pmatrix}, \quad (3.8)$$

where each of the diagonal squares CMS' s is corresponding to a referenced individual. The referenced individuals are sorted by fitness values in descending order. That is, the first diagonal square represents the highest fitness value, and the second square represents the second highest fitness value, and so on.

The upgrade matrix U is also for maintaining the best solution. If a state in the i^{th} diagonal square has a best fitness value higher than the fitness value of the referenced individual corresponding to the i^{th} diagonal square, this state is upgraded to the j^{th} , $j < i$, diagonal square in which the fitness value is equal to the best fitness value of the state. The structure of U is

$$U = \begin{pmatrix} U_{11} & & & & \\ U_{21} & U_{22} & & & \\ \vdots & \vdots & \ddots & & \\ U_{2^l,1} & U_{2^l,2} & \cdots & U_{2^l,2^l} & \end{pmatrix}, \quad (3.9)$$

which is a lower triangular matrix. Moreover, since it is assumed that there is a unique optimal solution for the optimization problem in Rudolph's work [Rudolph, G. (1994)], U_{11} is the only $2^{ml} \times 2^{ml}$ identity matrix. That is, none of the states in the first diagonal square need to be upgraded.

Figure 3.2 shows an example of the structure of PU with the dashes representing nonzero entries of PU , where $m = 1$ and $l = 3$. In fact, the figure can be generalized to any value of m and l .

3.3.3.3 The Proof of Convergence of CGA

Rudolph's proof on the convergence of CGA, with best solution maintained to its global optimal solution, is quite unique [Rudolph, G. (1994)]. The concept and method will be briefly reviewed in this subsection.

From Equation (3.8) and (3.9) in the previous subsection, we have

$$PU = \begin{pmatrix} CMSU_{11} & 0 \\ R & A \end{pmatrix}, \quad (3.10)$$

where R and A are corresponding sub-matrices. Since U_{11} is a $2^{ml} \times 2^{ml}$ identity matrix, the block matrix $CMSU_{11}$ in Equation (3.10) can be simplified as CMS .

According to Theorem 1, CMS is regular. Theorem 3 shows that the matrix PU converges (i.e., CGA with best solution maintained converges). In addition, based on Equation (3.5) and

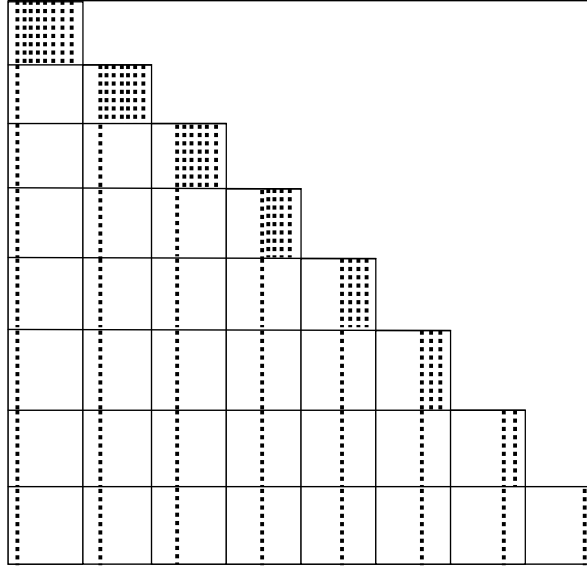


Figure 3.2 Structure of the Extended Transition Matrix PU

Equation (3.6), the first 2^{m_l} states are absorbing states. That is, regardless of the initial population (state), the populations (states) of CGA with best solution maintained will eventually be restricted within a subset of all the populations (states). According to the design of upgrade matrix U in the previous subsection, all the populations in the subset are under the condition that the global optimal solution is found. Their referenced individuals have the highest fitness value. Hence, we can conclude that CGA with best solution maintained converges to its global optimal solution.

CHAPTER 4. CONVERGENCE RATE AND THE FIRST HITTING TIME

Rudolph's proof shows that a CGA with the best solution maintained over time will eventually converge to its global optimal solution; however, how long it takes to converge is not specified. To address this problem, a study was conducted to investigate and analyze the convergence rate of a CGA with the best solution maintained over time.

4.1 Convergence Rate

From Theorem 4 [Perron's Formula], without loss of generality, we can bound $a_{ij}^{(n)}$ by estimating the term,

$$\left(\frac{d^{m_k-1}}{d\lambda^{m_k-1}} \left(\frac{\lambda^n A_{ij}(\lambda)}{\prod_{i \neq k} (\lambda - \lambda_i)^{m_i}} \right) \right)_{\lambda=\lambda_k}, \quad (4.1)$$

where $k \in \{1, 2, \dots, q\}$, as follows. Since $A_{ij}(\lambda)$ is equal to the product of $(-1)^{i+j}$ and determinant of the minor of $(\lambda I_r - A)_{ji}$, it is a polynomial of order $(r-1)$. Let

$$f_{ij}(\lambda) = \frac{\lambda^n A_{ij}(\lambda)}{\prod_{i \neq k} (\lambda - \lambda_i)^{m_i}}, \quad (4.2)$$

then we get the order of the function $f_{ij}(\lambda)$ is $(n + (r-1)) - (r - m_k) = n + (m_k - 1)$. Therefore, the order of $(m_k - 1)^{th}$ derivative of $f_{ij}(\lambda)$ in equation (4.1) is n . Moreover, a multiplier of $O(n)$ is applied everytime during the process in which the derivative is obtained. Hence, it is concluded that if A is a square matrix of order r , $\lambda_1, \lambda_2, \dots, \lambda_q$, $q \leq r$, are the eigenvalues of A , and m_1, m_2, \dots, m_q are the (algebraic) multiplicities of the eigenvalues, respectively, $m_1 + m_2 + \dots + m_q = r$, then there exists a positive number K such that

$$|a_{ij}^{(n)}| \leq K n^{m_{max}-1} |\lambda_{max}|^n, \quad (4.3)$$

where $a_{ij}^{(n)}$ is the element of A^n in the i^{th} row and j^{th} column, $m_{max} = \max\{m_1, m_2, \dots, m_q\}$, and $|\lambda_{max}| = \max\{|\lambda_1|, |\lambda_2|, \dots, |\lambda_q|\}$. This inequality will be exploited to show that the convergence rate is related to one of the eigenvalues of the transition matrix later.

From section 2, the transition matrix of the CGA with the best solution maintained is

$$PU = \begin{pmatrix} CMSU_{11} & & & \\ CMSU_{21} & CMSU_{22} & & \\ \vdots & \vdots & \ddots & \\ CMSU_{2^l,1} & CMSU_{2^l,2} & \cdots & CMSU_{2^l,2^l} \end{pmatrix}. \quad (4.4)$$

From the form of PU in Equation (4.4), we get that the eigenvalues of PU are the eigenvalues of the diagonal blocks [Suzuki, J. (1995)]. Moreover, in the matrix $(PU)^n$, where n is the fixed number of generations, the elements of the first 2^{ml} columns are related to the probabilities of convergence to the global optimum. With the initial distribution p_0 (a row vector) being known beforehand, to compute the probability of convergence to the global optimum, we have to sum up the first 2^{ml} elements of $p_0(PU)^n$. Without loss of generality, we can assume that $p_0 = (p_{0,1}, p_{0,2}, \dots, p_{0,2^{(m+1)l}})$. As an alternative, we compute the summation of the elements other than the first 2^{ml} ones of $p_0(PU)^n$.

In equation (4.4), the matrix PU can be represented as

$$PU = \begin{pmatrix} CMSU_{11} & 0 \\ R & A \end{pmatrix}, \quad (4.5)$$

where

$$R = \begin{pmatrix} CMSU_{21} \\ \vdots \\ CMSU_{2^l,1} \end{pmatrix}, \quad (4.6)$$

and

$$A = \begin{pmatrix} CMSU_{22} & & & \\ CMSU_{32} & CMSU_{22} & & \\ \vdots & \vdots & \ddots & \\ CMSU_{2^l,2} & CMSU_{2^l,3} & \cdots & CMSU_{2^l,2^l} \end{pmatrix}. \quad (4.7)$$

Then,

$$(PU)^n = \begin{pmatrix} (CMSU_{11})^n & 0 \\ \sum_{i=0}^{n-1} A^i R(CMSU_{11})^{(n-1)-i} & A^n \end{pmatrix}. \quad (4.8)$$

Let $v(n) = p_0(PU)^n$. Since A is a square matrix, for the element $a_{ij}^{(n)}$ of A^n , the inequality (4.3) holds. Hence, we have

$$\sum_{i=2^{2m}+1}^{2^{(m+1)l}} v_i(n) = \sum_{i=1}^{2^{(m+1)l}-2^{2m}} p_{0,2^{2m}+i} \sum_{j=1}^{2^{(m+1)l}-2^{2m}} a_{ij}^{(n)} \leq (2^{(m+1)l} - 2^{2m}) K n^{m_{max}-1} |\lambda_{max}|^n, \quad (4.9)$$

The following shows $|\lambda_{max}| < 1$. Since the upgrade matrix upgrades some columns of A to the first 2^{2m} columns of PU , the sum of each row of A is less than 1. Therefore, if λ is an eigenvalue of $A = (a_{ij}^{(1)})$ and $u = (u(i))$ is its left eigenvector, we have

$$\lambda u(j) = \sum_{i=1}^{2^{(m+1)l}-2^{2m}} u(i) a_{ij}^{(1)}, \quad j = 1, 2, \dots, 2^{(m+1)l} - 2^{2m}. \quad (4.10)$$

That is,

$$\sum_{j=1}^{2^{(m+1)l}-2^{2m}} |\lambda u(j)| \leq \sum_{i=1}^{2^{(m+1)l}-2^{2m}} |u(i)| \sum_{j=1}^{2^{(m+1)l}-2^{2m}} a_{ij}^{(1)} < \sum_{i=1}^{2^{(m+1)l}-2^{2m}} |u(i)|. \quad (4.11)$$

Hence, $|\lambda| < 1$. Since the eigenvalues of PU are the eigenvalues of the diagonal blocks, we get that the set of eigenvalues of A is contained in the set of eigenvalues of PU . Moreover, since $CMSU_{11}$ is a regular (stochastic) matrix (Theorem 3 in [Rudolph, G. (1994)]), from Lemma 1 and Theorem 5, we obtain that $\lambda_1 = 1$ is an eigenvalue of $CMSU_{11}$, i.e., $\lambda_1 = 1$ is an eigenvalue of PU , and λ_1 is *simple*. If we agree that the eigenvalues, $\lambda_1, \dots, \lambda_k$, $k \leq 2^{(m+1)l}$, of PU are in descending order $1 = \lambda_1 > |\lambda_2| \geq \dots \geq |\lambda_k|$, with m_1, m_2, \dots, m_k being the (algebraic) multiplicities, respectively, the following inequality can be derived.

$$\sum_{i=2^{2m}+1}^{2^{(m+1)l}} v_i(n) \leq K' n^{m'_{max}-1} |\lambda_2|^n, \quad (4.12)$$

where $K' > 0$ and m'_{max} is the maximal (algebraic) multiplicity of the eigenvalues of PU . That is, regardless the initial distribution of the populations, the probability of convergence to the global optimum is greater than or equal to $1 - K' n^{m'_{max}-1} |\lambda_2|^n$.

If $K' n^{m'_{max}-1} |\lambda_2|^n > 1$, that means either the fixed n is not large enough, or λ is not small enough, this equation is meaningless. For any optimization problem, we first need to

compute its transition matrix CMS , extend it to PU , find the eigenvalue λ_2 , which is related to the convergence rate of the problem, of PU , and then apply λ_2 to $K'n^{m'_{max}-1}|\lambda_2|^n$. With n satisfying $K'n^{m'_{max}-1}|\lambda_2|^n \leq 1$, we can obtain the CGA's hardness of the problem (i.e., the applicability of CGA to the problem). Nevertheless, it is concluded that the computations described above are time consuming and impractical for the current state of the art. By the following subsection, we get that traces of the transition matrices $(PU)^n$ is highly related to $|\lambda_2|$. In Chapter 5, the relationship is modified and applied so that an alternative approach to substituting the estimation of $|\lambda_2|$ is proposed to estimate the applicability of the CGA to optimization problems. Chapter 7 uses the proposed method in Chapter 5 and provides a possible approximation method to estimate the number of generations needed for the global convergence.

4.1.1 Eigenvalues of Transition Matrices and Fix Points of Genetic Algorithm

The convergence rate is shown to be related to $|\lambda_2|$, the second largest eigenvalue of the transition matrix in absolute value. This subsection provides the relationship between the second largest eigenvalue of the transition matrix and the fix points of CGA.

Theorem 6 demonstrates that every square matrix is unitarily equivalent to a triangular matrix whose diagonal entries are the eigenvalues of the matrix. Hence, the transition matrix $(PU)^n$, representing transition probabilities for every n generations of a CGA, can be written as

$$(PU)^n = (STS^*)^n = ST^nS^*, \quad (4.13)$$

where S is a unitary matrix and T is upper triangular with diagonal entries are the eigenvalues of PU . Let the eigenvalues of PU be denoted as $\lambda_1 = 1, \lambda_2, \dots, \lambda_{2(m+1)l}$ (in absolute descending order). Then T^n is an upper triangular matrix with diagonal entries are $\lambda_1^n, \lambda_2^n, \dots, \lambda_{2(m+1)l}^n$. Moreover, the trace of $(PU)^n$ is

$$\text{Trace}((PU)^n) = \text{Trace}((ST^n)S^*) = \text{Trace}(S^*ST^n) = \text{Trace}(T^n) = \sum_{i=1}^{2(m+1)l} \lambda_i^n. \quad (4.14)$$

By the inequality

$$\left| \sum_{i=1}^{2^{(m+1)l}} \lambda_i^n \right| \leq \sum_{i=1}^{2^{(m+1)l}} |\lambda_i|^n \leq 1 + (2^{(m+1)l} - 1)|\lambda_2|^n, \quad (4.15)$$

it is obtained that

$$\text{Trace}((PU)^n) \leq 1 + (2^{(m+1)l} - 1)|\lambda_2|^n. \quad (4.16)$$

That is, $|\lambda_2|^n$ is bounded (from the left) by the trace of the transition matrix $(PU)^n$ with a constant multiple. Since $|\lambda_i| < 1$ for $i \in \{2, \dots, 2^{(m+1)l}\}$, there exists a number $M_1 > 0$ such that

$$\text{Trace}((PU)^n) = 1 + \lambda_2^n + \dots + \lambda_{2^{(m+1)l}}^n \geq C_1, \quad \forall n \geq M_1,$$

where $C_1 \in (0, 1]$ is a constant. That implies that there exists a constant K^n such that

$$1 + (2^{(m+1)l} - 1)|\lambda_2|^n \leq 1 + (2^{(m+1)l} - 1) \leq K^n \text{Trace}((PU)^n), \quad \forall n \geq M_1. \quad (4.17)$$

Equation (4.17) shows that if n is large enough, $\text{Trace}((PU)^n)$ with a constant multiple can also form an upper bound for $|\lambda_2|^n$. Combined with Inequality (4.16), it can be known that computing bounds for $|\lambda_2|^n$ with $n \geq M_1$ is equivalent to computing the trace of the transition matrix to the power n . Additionally, the diagonal element of the transition matrix corresponding to each state shows probability of a fix point over the solutions in the CGA search space. The product of any distribution of transient states and the diagonal elements of the transition sub-matrix corresponding to the transient states represents the total probability of fix points, which are outside the set of optimal populations (absorbing states) of CGA. Note that a population ξ^* is called an optimal population if there exists i , $1 \leq i \leq m + 1$, such that $\pi_i(\xi^*)$ is an optimal solution. For searching purpose, we do not want CGA to stay within a transient state too long since the state is already visited. The revisiting of a state will cost some time without any improvement of solutions. In other words, if the total probability of fixed points of CGA can be reduced, the probability of CGA to search other candidate states will be increased for the number of the states is finite.

4.2 Second Largest Eigenvalue versus Expected First Hitting Time

The research on computation time of GAs used to solve optimization problems is important for the foundation and theory of evolutionary algorithms. Both convergence rate and expected first hitting time express the information on time complexity.

From Ding's work [Ding, L. and Yu, J. (2005)], the *first hitting time* is defined as

$$\tau(\xi^*) = \min\{k \geq 0 | \xi_k = \xi^*\},$$

where ξ^* is an optimal population, and $\{\xi_k | k \in N\}$ is a discrete homogeneous Markov chain. Moreover, the expected first hitting time is calculated as

$$E[\tau(\xi^*)] = \sum_{k \geq 0} k \times P\{\tau(\xi^*) = k\}. \quad (4.18)$$

As mentioned in Chapter 2, a Markov theorem tells us that the expected value of the transient time in which an arbitrary state i goes to any other state j is finite. That is, there exists $0 < M_1 < \infty$ such that

$$E[\tau(\xi^*)] < M_1,$$

since the optimal state is also a state in the state space of the Markov chain. With respect to the same problem, if the second largest eigenvalue in absolute value of PU : $\lambda_2 \neq 0$ (i.e., CMS does not have identical rows), we can get that there exists $0 < C_1 < \infty$ such that

$$E[\tau(\xi^*)] < C_1 |\lambda_2|. \quad (4.19)$$

For an arbitrary fitness function, an inequality similar as Inequality (4.19) can be derived. In fact, if a problem has the corresponding $\lambda_2 \neq 0$, it is possible to use λ_2 to estimate the first hitting time. Suppose $\lambda_2 \neq 0$ is given while m and l are also given. Based on the arrangement of eigenvalues in Section 4.1.1, the inequality

$$1 + |\lambda_2|^n + |\lambda_3|^n + \dots + |\lambda_{2^{(m+1)l}}|^n \leq 1 + (2^{(m+1)l} - 1) |\lambda_2|^n$$

always holds. For a small $\epsilon > 0$, the solution of $\epsilon = (2^{(m+1)l} - 1) |\lambda_2|^n$ on n can be an estimation of the first hitting time.

CHAPTER 5. THE EVALUATION METRICS

5.1 The Bridge between Theory and Practice

Although the expected first hitting time and the convergence rate can be mathematically computed, there is still a huge gap between the theoretical prediction and the estimation on the applicability of GAs to problems in real world applications. As I know, the calculations on the expected first hitting time or convergence rate include the computation on the corresponding transition matrix of problems. To obtain the transition matrix with respect to a problem, the matrices C , M , and S should be considered. Since S is derived from selection operator, in which the proportional selection is applied, the construction time is much longer than the total computation time on the fitness values of the entire search space. Hence, the theory is only an ideal view. The real world demands a practical approach for the estimation on applicability of GAs.

In order to derive a more practical approach, a direction related to Markov chain analysis is suggested. It is introduced as follows.

5.1.1 Equivalent Forms in terms of Convergence

The extended transition matrix PU is used to prove that the CGA with best solution maintained converges to the global optimal solution. Because the computation time of the transition matrix is not practical, CMS and PU cannot be computed directly. PU includes even more states than CMS . My goal is to find a way which can extract only the essential properties of the transition matrix so that an evaluation metric can be formulated to evaluate the applicability of CGAs to real-world problems.

Instead of PU , the transition matrix CMS is used to develop the metric in the next section.

According to Theorem 3, the convergence of $(PU)^n$ as $n \rightarrow \infty$ is proved; however, it does not show much information about the relationship between $(PU)^n$ and $(CMS)^n$, especially when n is finite. In order to further investigate the convergence behavior among $(PU)^n$, $(CMS)^n$, and A^n (the sub-matrix of $(PU)^n$ in Equation (3.10)) within finite steps (i.e., $n \in N$ and $n < \infty$), Theorem 7 is formulated. Note that the term *empirical convergence* is defined in Definition 8.

Theorem 7 *The empirical convergence of $\{(CMS)^n\}_{n \in N}$ at $n = n_1$ and $\{A^n\}_{n \in N}$ at $n = n_2$ with respect to $\frac{\epsilon}{2\sqrt{2(2^l-1)}}$ implies the empirical convergence of $\{(PU)^n\}_{n \in N}$ at $n = k$ with respect to ϵ , where $k \geq \max\{n_1, n_2\}$, A is the sub-matrix of PU in Equation (3.10), and $\epsilon > 0$ is a sufficiently small number.*

Proof. To prove the theorem, one has to first verify with respect to each row of blocks of $(PU)^n$, if all of the blocks are summed, the result is equal to $(CMS)^n$, for all $n \in N$. The statement is proved by applying Mathematical Induction on n (the steps).

First, suppose that $n = 2$. Based on the structure of PU (e.g., Figure 3.2), the property can be obtained that each row of blocks has exactly 2^{ml} (the order of CMS) nonzero columns. Let all of the 2^{ml} columns be named from left to right as $column_1, column_2, \dots, column_{2^{ml}}$. It can be shown that their positions within any row of blocks are distinct and sorted. That is, within a row of blocks, no matter which block it is in, $column_1$ is always the first column in the block, $column_2$ is always the second column in the block, and so on. To compute $PU \cdot PU$, for any row i , $1 \leq i \leq 2^{(m+1)l}$, of the left PU , the first nonzero entry, which is in the first column, is multiplied with the first row of the right PU , the second nonzero entry, which is in the second column, is multiplied with the second row in the corresponding row of blocks of the right PU , and so on. (Figure 5.1 shows an illustration of $PU \cdot PU$, where $m = 1$ and $l = 3$.) Since the 2^{ml} nonzero entries in the i^{th} row of the left PU are from a row of CMS , and rows with 2^{ml} nonzero entries of the right PU are from rows of CMS , it can be derived that

$$Row_j((CMS)^2) = \sum_{g=1}^{2^l} Row_j((PU)_{hg}^2), \quad (5.1)$$

where $j = (i \bmod 2^{ml})$, $Row_j(\cdot)$ indicates the j^{th} row in the block, $(PU)_{uv}^2$ is the block in the

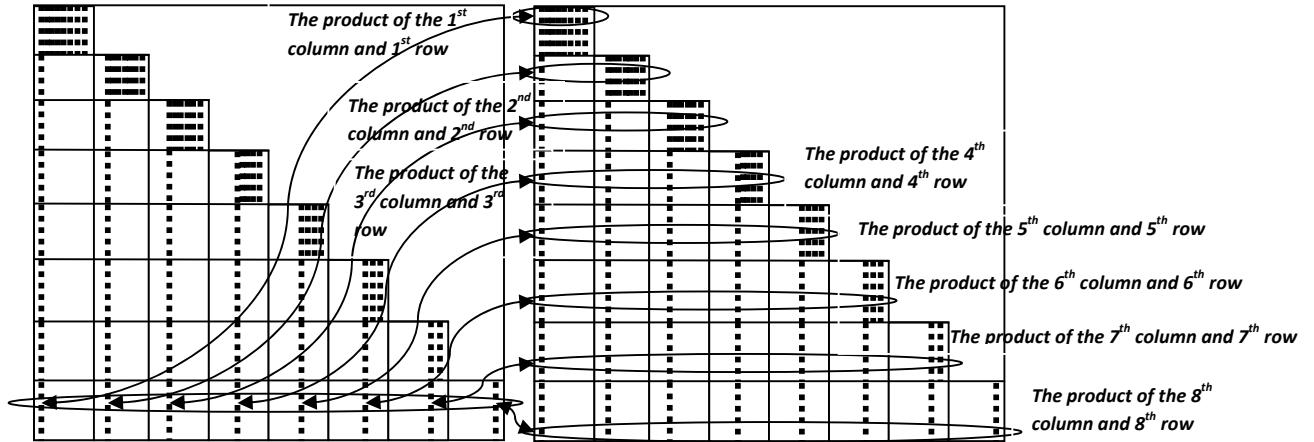


Figure 5.1 An illustration of the Product $PU \cdot PU$

u^{th} row and v^{th} column of $(PU)^2$, and $h = \lceil \frac{i}{2^{ml}} \rceil$. From Equation (5.1), it can be obtained that with respect to each row of blocks of $(PU)^2$, if all of the blocks are summed, the result is equal to $(CMS)^2$. In other words, the statement holds for $n = 2$.

Suppose for $n = k$, the statement holds. That is, with respect to each row of blocks of $(PU)^k$, if all of the blocks are summed, the result is equal to $(CMS)^k$.

For $n = k + 1$, the product $PU \cdot (PU)^k$ needs to be computed. Let $(PU)_{uv}^k$ be the block in the u^{th} row and v^{th} column of $(PU)^k$. For any row i , $1 \leq i \leq 2^{(m+1)l}$, of PU , assume $C_1, C_2, \dots, C_{2^{ml}}$ are the aforementioned nonzero entries, $j = (i \bmod 2^{ml})$, and $h = \lceil \frac{i}{2^{ml}} \rceil$. Then, based on the rule for the product of two matrices, it is derived that

$$\begin{aligned} Row_j((PU)_{hw}^{k+1}) &= \sum_{z=0}^{h-w-1} C_{w+z} Row_{w+z}((PU)_{(w+z)w}^k) \\ &\quad + \sum_{x=h}^{2^{ml}} C_x Row_x((PU)_{hw}^k), \end{aligned}$$

if $w \leq h$, and

$$Row_j((PU)_{hw}^{k+1}) = (0, \dots, 0)_{2^{ml}},$$

if $w > h$. Hence, it is obtained that

$$\begin{aligned}
\sum_{w=1}^{2^l} \text{Row}_j((PU)_{hw}^{k+1}) &= \sum_{w=1}^h \text{Row}_j((PU)_{hw}^{k+1}) \\
&= C_1 \text{Row}_1((PU)_{11}^k) + C_2 \text{Row}_2((PU)_{21}^k + (PU)_{22}^k) + \\
&\quad C_3 \text{Row}_3((PU)_{31}^k + (PU)_{32}^k + (PU)_{33}^k) + \cdots + \\
&\quad C_{h-1} \text{Row}_{h-1}((PU)_{(h-1)1}^k + \cdots + (PU)_{(h-1)(h-1)}^k) + \\
&\quad C_h \text{Row}_h((PU)_{h1}^k + \cdots + (PU)_{hh}^k) + \\
&\quad C_{h+1} \text{Row}_{h+1}((PU)_{h1}^k + \cdots + (PU)_{hh}^k) + \\
&\quad \cdots + C_{2^m l} \text{Row}_{2^m l}((PU)_{h1}^k + \cdots + (PU)_{hh}^k).
\end{aligned} \tag{5.2}$$

Since it is already known that with respect to each row of blocks of $(PU)^k$, if all of the blocks are summed, the result is equal to $(CMS)^k$ (i.e., the statement holds for $n = k$), it can be derived that the right hand side (RHS) of Equation (5.2) is equal to $\text{Row}_j((CMS)^{k+1})$. Hence, the statement holds for $n = k + 1$. That is, with respect to each row of blocks of $(PU)^{k+1}$, if all of the blocks are summed, the result is equal to $(CMS)^{k+1}$.

Secondly, the empirical convergence of $(PU)^n$ with respect to ϵ needs to be proved. Let $u_{ij}^{(n)}$ be the element in the i^{th} row and j^{th} column of $(PU)^n$.

By Definition 8 and the fact that $\{(CMS)^n\}$ converges empirically at $n = n_1$ and $\{A^n\}$ converges empirically at $n = n_2$ with respect to the same $\frac{\epsilon}{2\sqrt{2(2^l-1)}} > 0$, Inequality (5.3) and Inequality (5.4) are derived. For $\frac{\epsilon}{2\sqrt{2(2^l-1)}} > 0$ and the constant n_1 ,

$$\left(\sum_{i,j=1}^{2^{ml}} |v_{ij}^{(n)} - v_j^*|^2 \right)^{1/2} < \frac{\epsilon}{2\sqrt{2(2^l-1)}}, \forall n \geq n_1, \tag{5.3}$$

where $v_{ij}^{(n)}$ is the element in the i^{th} row and j^{th} column of $(CMS)^n$, and $e(v^*)^T$ is the limit of $(CMS)^n$ (see Theorem 2). In addition, for $\frac{\epsilon}{2\sqrt{2(2^l-1)}} > 0$ and the constant n_2 ,

$$\left(\sum_{i,j=1}^{2^{(m+1)l}-2^{ml}} |a_{ij}^{(n)} - 0|^2 \right)^{1/2} < \frac{\epsilon}{2\sqrt{2(2^l-1)}}, \forall n \geq n_2, \tag{5.4}$$

where $a_{ij}^{(n)}$ is the element in the i^{th} row and j^{th} column of A^n , and 0 is the limit of A^n . Then

$\exists n_3 = \max\{n_1, n_2\}$ such that

$$\begin{aligned} \sum_{i=2^{m+1}+1}^{2^{(m+1)l}} \sum_{j=1}^{2^{ml}} |u_{ij}^{(n)} - v_j^*|^2 &= \sum_{i=2^{m+1}+1}^{2^{(m+1)l}} \sum_{j=1}^{2^{ml}} |u_{ij}^{(n)} + \sum_{k=0}^{2^l-2} a_{i,k \cdot 2^{ml}+j}^{(n)} - v_j^* - \sum_{k=0}^{2^l-2} a_{i,k \cdot 2^{ml}+j}^{(n)}|^2 \\ &= \sum_{i=2^{m+1}+1}^{2^{(m+1)l}} \sum_{j=1}^{2^{ml}} |v_{ij}^{(n)} - v_j^* - \sum_{k=0}^{2^l-2} a_{i,k \cdot 2^{ml}+j}^{(n)}|^2 \\ &\leq \sum_{i=2^{m+1}+1}^{2^{(m+1)l}} \sum_{j=1}^{2^{ml}} (1^2 + (-1)^2) (|v_{ij}^{(n)} - v_j^*|^2 + |\sum_{k=0}^{2^l-2} a_{i,k \cdot 2^{ml}+j}^{(n)}|^2) \end{aligned}$$

(by Cauchy-Schwarz Inequality)

$$\leq \sum_{i=2^{m+1}+1}^{2^{(m+1)l}} \sum_{j=1}^{2^{ml}} 2 \cdot (|v_{ij}^{(n)} - v_j^*|^2 + (2^l - 1) \cdot \sum_{k=0}^{2^l-2} |a_{i,k \cdot 2^{ml}+j}^{(n)}|^2)$$

(by Cauchy-Schwarz Inequality)

$$\begin{aligned} &\leq 2 \sum_{i=2^{m+1}+1}^{2^{(m+1)l}} \sum_{j=1}^{2^{ml}} |v_{ij}^{(n)} - v_j^*|^2 + 2(2^l - 1) \sum_{i=2^{m+1}+1}^{2^{(m+1)l}} \sum_{j=1}^{2^{ml}} \sum_{k=0}^{2^l-2} |a_{i,k \cdot 2^{ml}+j}^{(n)}|^2 \\ &< 2 \cdot (2^l - 1) \cdot \frac{\epsilon^2}{8(2^l - 1)} + 2 \cdot (2^l - 1) \cdot \frac{\epsilon^2}{8(2^l - 1)} = \frac{\epsilon^2}{2}, \forall n \geq n_3. \end{aligned}$$

Then by Inequality (5.3) and Inequality (5.4),

$$\begin{aligned} \sum_{i,j=1}^{2^{(m+1)l}} |u_{ij}^{(n)} - (pu)_{ij}^*|^2 &< \frac{\epsilon^2}{8(2^l - 1)} + \frac{\epsilon^2}{8(2^l - 1)} + \frac{\epsilon^2}{2} \\ &< \frac{\epsilon^2}{4} + \frac{\epsilon^2}{4} + \frac{\epsilon^2}{2} \quad (1 \geq 1) \\ &= \epsilon^2, \end{aligned}$$

where $(pu)_{ij}^*$ is the element in the i^{th} row and j^{th} column of the limit of $(PU)^n$.

That is, from the statement of Mathematical Induction and the fact that $\{(CMS)^n\}$ converges empirically at $n = n_1$ and $\{A^n\}$ converges empirically at $n = n_2$ with respect to the same $\frac{\epsilon}{2\sqrt{2(2^l-1)}}$, the empirical convergence of $\{(PU)^n\}$ at $n = k$, $k \geq \max\{n_1, n_2\}$, with respect to ϵ is proved. \square

Based on Inequality (3.4) and Inequality (4.12), it is implied that the convergence of $\{(CMS)^n\}$ and $\{A^n\}$ depend on $|\lambda_{2,CMS}|$ (the second largest eigenvalue of CMS in absolute

value) and $|\lambda_2|$, respectively. Since CMS is one of the diagonal blocks of PU , $|\lambda_{2,CMS}| \leq |\lambda_2|$. If $0 < \lambda_{2,CMS} < 1$, sufficiently small $\lambda_{2,CMS}^n$ implies that n is large enough. Hence, $|\lambda_2|^n$ is small enough for the empirical convergence of A^n . Suppose after $(n-1)$ generations, the elements in A^{n-1} satisfy

$$\left(\sum_{i,j=1}^{2^{(m+1)l}-2^{ml}} |a_{ij}^{(n-1)} - 0|^2 \right)^{1/2} < \frac{\epsilon}{2^{(m+1)l}-2^{ml}}, \text{ for some } \epsilon. \quad (5.5)$$

Then regardless of the initial distribution,

$$Pr\{\text{The } n^{\text{th}} \text{ Generation is an Optimal State}\} > 1 - \epsilon. \quad (5.6)$$

The proof of that is as follows. Suppose the initial distribution is $p_0 = (p_{0,1}, p_{0,2}, p_{0,3}, \dots, p_{0,2^{(m+1)l}})$, where $p_{0,1}, \dots, p_{0,2^{ml}}$ are corresponding to the absorbing states (related to the global optimal solution). Since p_0 is a probability distribution, it can be derived that $\sum_{i=2^{ml}+1}^{2^{(m+1)l}} p_{0,i} \leq 1$. According to Equation (3.5) and Equation (3.10), we get

$$\begin{aligned} & p_0(PU)^{n-1} \\ &= (p_{0,1}, \dots, p_{0,2^{ml}}, p_{0,2^{ml}+1}, \dots, p_{0,2^{(m+1)l}}) \begin{pmatrix} (CMSU_{11})^{n-1} & 0 \\ \sum_{i=0}^{n-2} A^{n-2-i} R(CMSU_{11})^i & A^{n-1} \end{pmatrix}. \end{aligned} \quad (5.7)$$

From Inequality (5.5), it is derived that $a_{ij} < \frac{\epsilon}{2^{(m+1)l}-2^{ml}}$ for all $i, j \in \{1, 2, \dots, 2^{(m+1)l}-2^{ml}\}$. With Equation (5.7), we obtain that the probability for the n^{th} generation (state) to be an transient state is

$$\begin{aligned} & p_{0,2^{ml}+1} \cdot (a_{11} + a_{12} + \dots + a_{1,2^{(m+1)l}-2^{ml}}) \\ &+ \dots \\ &+ p_{0,2^{(m+1)l}} \cdot (a_{2^{(m+1)l}-2^{ml},1} + \dots + a_{2^{(m+1)l}-2^{ml},2^{(m+1)l}-2^{ml}}) \\ &< p_{0,2^{ml}+1} \cdot \epsilon + \dots + p_{0,2^{(m+1)l}} \cdot \epsilon \\ &< 1 \cdot \epsilon = \epsilon. \end{aligned}$$

Hence, the probability for the n^{th} generation (state) to be an optimal (absorbing) state is greater than $1 - \epsilon$.

5.2 The Proposed Evaluation Metric

5.2.1 Overview of the Methodology

This section proposes an evaluation metric, which derives the applicability of a CGA to a problem from the estimated trace of the corresponding transition matrix. Through this method, a degree of convergence can be determined for each CGA run. According to that, researchers and engineers will be able to obtain a comprehensive view of the applicability of CGA and know how good the solutions generated by CGA, so that more correct decisions can be made. The general methodology diagram is illustrated in Figure 1.1.

5.2.2 Evaluating the Applicability of GAs from Run Time Data

The major concerns for the applicability of GAs to real-world problems are the computation time and the quality of the obtained solutions. Actually, there are tradeoffs between them. For instance, typically the longer the computation time, the better the solution is. Since they are tightly related, without loss of generality, we can fix one and utilize another to estimate the applicability of GAs to problems. Assumption 3 forces us to fix the computation time, whereas the quality of the obtained solution will be used to measure the problem applicability of GAs. Again, the discussion will be focused on CGA.

Due to the lack of information on the landscapes of fitness functions and encodings, it is difficult to analyze the quality of the obtained solution itself; however, the quality of the obtained solution can be measured indirectly through the degree of convergence under certain evaluation metrics from run time data. Theorem 8 functions as a support for the evaluation metric on degrees of convergence. The reason for employing the trace of $(CMS)^{n'}$, where n' is fixed, to estimate degrees of convergence is because it quantifies degrees of convergence. With it, a number on the degree of convergence of CGA will be given at the end of a CGA run. Based on that, one can determine whether or not the CGA run converges within n' generations.

Theorem 8 *The trace of $(CMS)^n$ converges to 1 as $n \rightarrow \infty$.*

Proof. By Theorem 1, 5, and Lemma 1, it is obtained that CMS has only one eigenvalue which is equal to 1 and others are less than 1.

Moreover, by Theorem 6, every square matrix is unitarily equivalent to a triangular matrix whose diagonal entries are the eigenvalues of the matrix. Hence, $(CMS)^n$ can be written as

$$(CMS)^n = (STS^*)^n = ST^n S^*,$$

where S is a unitary matrix and T is upper triangular. The diagonal entries of T are the eigenvalues of CMS . Let the eigenvalues of CMS be denoted as $\lambda_1, \dots, \lambda_{2^{ml}}$ with $1 = \lambda_1 > |\lambda_2| \geq \dots \geq |\lambda_{2^{ml}}|$. Then

$$\text{Trace}((CMS)^n) = \text{Trace}((ST^n)S^*) = \text{Trace}(S^*ST^n) = \text{Trace}(T^n) = \sum_{i=1}^{2^{ml}} \lambda_i^n. \quad (5.8)$$

From Equation (5.8), the result, the trace of $(CMS)^n$ converges to 1 as $n \rightarrow \infty$, is derived.

□

In fact, Theorem 8 coincides with a result from Theorem 2. In Theorem 2, since CMS is regular, it converges to a positive stable stochastic matrix $\Pi = ev^T$ as $n \rightarrow \infty$, where $e = (1, 1, \dots, 1)^T$ is a $2^{ml} \times 1$ column vector in which all elements are of the value 1, and $v^T = (v_1, v_2, \dots, v_r)$ is a 1×2^{ml} probability row vector with non-null entries. Based on that, the result, $\lim_{n \rightarrow \infty} \text{Trace}((CMS)^n) = 1$, can be obtained.

5.2.3 Theoretical Framework for Approximating the Applicability of CGAs

This subsection mainly discusses the methodology adopted to evaluate the applicability of CGAs based on Theorem 8.

The overview of the idea includes:

1. Applying a Monte-Carlo-like simulation (Empirical Probability) [Leemis, L. H. and Park, S. K. (2005)] to estimate $\text{Trace}((CMS)^{n'})$, $n' \in N$, from states visited by a CGA run. Note that n' should be chosen carefully so that $n - n'$ can be a large number, where n is the number of generations.

2. Comparing the obtained $Trace((CMS)^{n'})$ with the “expected” value 1, and then deriving a value for the degree of convergence with respect to that CGA run.

In Assumption 3, n , the number of generations, is a reasonably large number and it is fixed. After a CGA run, there will be n generations in total. Each generation corresponds to a state from the state space of the Markov transition matrix CMS . During a CGA run, a state representing the next generation will be generated after every selection operation. If all of the n generations (states) are recorded, a list of n states sorted chronologically will be generated at the end of each CGA run. With that list, a number $n' \in N$, where $n' < n$ and $n - n'$ is a large number, is chosen. The starting states and the ending states are determined accordingly to estimate the trace of $(CMS)^{n'}$. Figure 5.2 illustrates this concept; it shows that for each generation (state) i , $1 \leq i \leq (n - n')$, given as a starting state, the $(i + n')^{th}$ generation (state) is the corresponding ending state. After the matching, there are $(n - n')$ pairs of (Starting State, Ending State). Based on the results, one can compute the frequency, or probability, of any starting state which goes back to itself. If all the obtained frequencies are summed up, the obtained value is the approximation of $Trace((CMS)^{n'})$. Note that an ending state of a pair can also be a starting state of another pair.

According to my empirical investigation, the transition matrix CMS constructed by CGA has the property that its eigenvalues, which have large absolute values, are all real and positive if p_m is small (e.g. $p_m < 0.45$). Only a few eigenvalues of CMS , whose absolute values are close to 0, are complex numbers. Intuitively, the transition matrix M constructed by a small mutation probability p_m is symmetric and strictly diagonally dominant (i.e., M is a positive definite matrix). It has eigenvalues which are all real and positive. The sparse matrices C and S do not significantly impact the eigenvalues of M . Therefore, the eigenvalues of CMS primarily depend on the eigenvalues of M . This fact implies that $Trace((CMS)^{n'})$ decreases to 1 as n' increases.

Theorem 9 *Any square matrix with identical rows has only one nonzero eigenvalue, which is equal to the sum of any row of the matrix.*

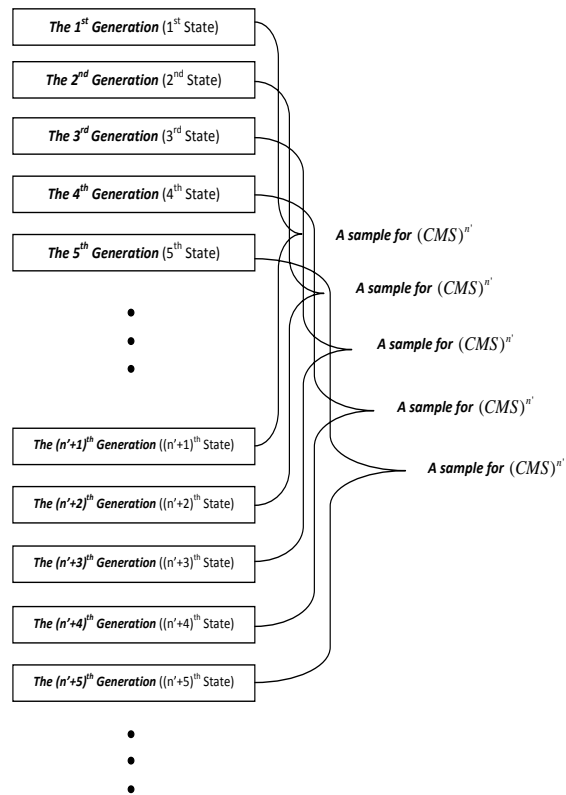


Figure 5.2 The Estimation on the Trace of $(CMS)^{n'}$

Proof. Suppose that A is an arbitrary square matrix of order r with identical rows. First, the claim that 0 is an eigenvalue of A is going to be proved. Based on the definition of eigenvalues [Horn, R. A. and Johnson, C. R. (1985)], λ is an eigenvalue of A if and only if $\det(A - \lambda I_r) = 0$. Let $\lambda = 0$ and we have $\det(A - 0 \cdot I_r) = \det(A) = 0$ because A has identical rows. Hence, 0 is an eigenvalue of A .

Secondly, it is claimed that the eigenspace of A corresponding to 0 (i.e., the (algebraic) multiplicity of 0) is $r - 1$. To compute that, the number of linearly independent eigenvectors corresponding to 0 should be calculated. Consider the equation

$$Ax = 0 \cdot x, \quad x \neq 0.$$

Since A has identical rows, there is only one constraint for the r -dimensional vector x . Therefore, there are $r - 1$ linearly independent eigenvectors corresponding to the eigenvalue 0. That is, the eigenspace of A corresponding to 0 is $r - 1$.

Now, let us prove that the only nonzero eigenvalue of A is the sum of any row of A . Let x be an $r \times 1$ vector $e = (1, 1, \dots, 1)^T$. Then

$$Ax = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ a_{11} & a_{12} & \cdots & a_{1r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{11} & a_{12} & \cdots & a_{1r} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = (a_{11} + a_{12} + \cdots + a_{1r}) \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Hence, $(a_{11} + a_{12} + \cdots + a_{1r})$ is an eigenvalue of A . \square

Theorem 9 tells us that any stochastic stationary state matrix has eigenvalues $1, 0, 0, \dots, 0$. Based on the discussion above, Theorem 8, and Theorem 9, to determine the degree of convergence one simply needs to compare the obtained value with 1. More specifically, all the diagonal blocks of A are principal submatrices of CMS and M is a positive definite matrix. By the theoretical result from the current state of the art in mathematics [Hwang, S. (2004)] and empirical study, the second largest eigenvalue of PU , λ_2 , is a little bit larger than, but very close to $\lambda_{2_{CMS}}$. Hence, it can be derived that the trace of $(CMS)^{n'}$, where $n' \in N$, needs to

be close to 1 first (i.e., $|\text{Obtained Value} - 1|$ close to 0), and then the global convergence occurs. That is, $(CMS)^{n'}$ has to be first close to its stationary state matrix, and then $A^{(n'+k)}$, where $k \in N$, is going to be close to 0. Therefore, if the value (or criterion) $|\text{Obtained Value} - 1|$ is close to 0, the degree of convergence is high.

The merit of this approach is that for every CGA run, a degree of convergence can be derived from run time data. With n' being fixed, one can compare different encodings of the same problem, or obtain the values on how good the solutions are for several CGA runs with respect to different problems. Researchers can evaluate the best solution obtained by a CGA based on the degree of convergence.

5.2.4 Foundation of the Methodology

According to Theorem 8 and the discussion in previous subsection, it can be derived that the trace of $(CMS)^n$ determines the degree of convergence. In other words, to obtain the degree of convergence, one can calculate the trace of $(CMS)^n$ first. A traditional way to compute it is first to compute the transition matrix CMS ; however, as mentioned, the construction time of CMS takes more than the total computation time of all feasible solutions. Therefore, the traditional way is considered to be impractical due to the large computation time.

My methodology to approximate the trace of $(CMS)^{n'}$ is based on the following. In Figure 5.2, the first generation (state) is chosen at random from all feasible solutions. After the first generation is determined, the second generation can be generated by the crossover, mutation, and selection operators of a CGA accordingly. In the perspective of Markov chains, the distribution of the second generation is generated based on the product of the distribution of the first generation and the transition matrix CMS . Similarly, the distribution of the third generation is generated based on the product of the distribution of the first generation and the transition matrix $(CMS)^2$, and so on. In general, for $i \in N$, if the distribution of the i^{th} generation is determined, the distribution of the $(i + n')^{th}$ generation can be generated based on the product of the distribution of the i^{th} generation and the transition matrix $(CMS)^{n'}$. To estimate the trace of $(CMS)^{n'}$, the i^{th} generation and the $(i + n')^{th}$ generation need to

be coupled ($i, n' \in N$). Now consider the physical meaning of the trace of $(CMS)^{n'}$. The diagonal elements of $(CMS)^{n'}$ represent the probabilities of the fix points (states) from the i^{th} generation to the $(i+n')^{th}$ generation for a CGA. Therefore, the summation of the frequencies (probabilities) of any starting state going back to itself approximates the trace of $(CMS)^{n'}$.

A question about the distribution of samples may be raised. Besides the first generation, the i^{th} generation ($i \geq 2$) is dependent on the $(i-1)^{th}$ generation. That is, after the first generation, the starting state is not chosen uniformly. According to my investigation, this dependency will not cause any problem. On the other hand, it helps improve the accuracy of the approximation. Suppose the transition matrix

$$(CMS)^{n'} = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1,2^{ml}} \\ t_{21} & t_{22} & \vdots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ t_{2^{ml},1} & \cdots & \cdots & t_{2^{ml},2^{ml}} \end{pmatrix}. \quad (5.9)$$

For the purpose of illustration, we may assume that t_{11} has a large value, say 0.99, and $t_{2^{ml},2^{ml}}$ has a small value, say 0.01. If the first state (i.e., the state corresponds to the element t_{11} in Equation (5.9)) is generated as the i^{th} generation ($i \leq n$), in the $(i+n')^{th}$ generation, it has a high probability (0.99) to be chosen again. Other states have lower probability to become the $(i+n')^{th}$ generation. Similarly, if the last state is generated as the i^{th} generation, in the $(i+n')^{th}$ generation, it has merely 0.01 probability to be chosen again. Other states have a large probability to become the $(i+n')^{th}$ generation. Accordingly, a state with a large diagonal value has a high probability to become a starting state, and more samples will be collected for estimating it. A state with a small diagonal value may be ignored from the estimation. Compared with the samples collected uniformly, it increases the accuracy of the approximation since the number of samples is fixed (by n and n') and the large diagonal values dominate the trace. If more samples with the starting states being the states with small diagonal values are gathered, fewer samples with starting states being the states with large diagonal values will be collected. In that case, the estimations of large diagonal values as well as the approximation of the trace may not be so accurate.

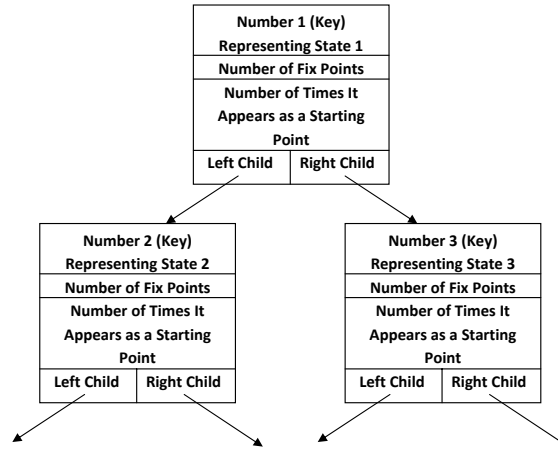


Figure 5.3 A possible data structure for the proposed methodology

5.2.5 The Implementation and Data Structure of the Proposed Methodology

Many data structures, such as binary search trees, hash tables, and linked lists, can be applied to implement the proposed methodology. For the purpose of demonstration, binary search trees are applied (see Figure 5.3). Other data structures are also encouraged to be used to speed up the computation time.

For each node in the binary search tree, there are four fields. The first field is a number, the key of the node, representing the starting state converted from a binary string. The second field represents the number of ending states which are the same as the starting state in the first field, and the third field represents the total number of occurrences of the starting state in the first field. The fourth field represents the addresses for the roots of sub-trees.

The following shows how the proposed methodology is implemented:

1. Set the root of the tree to be NULL.
2. Record the first n' states as starting states using an array.
3. Start from the $n' + 1$ state, do 4) to 7) until the stop condition is satisfied.
4. Search for the corresponding starting state in the tree. If it is not found, insert the

corresponding starting state in the tree.

5. Compare the current state with the corresponding starting state. If it is the same as the starting state, add 1 to the second field.
6. Add 1 to the third field.
7. Replace the corresponding starting state with the current state in the array.

Since for each generation the selection operator selects individuals for the entire state, all of the states can be recorded at the same time as the selection. The recording will not increase the complexity of the computation time. In the next subsection, detailed information on the space and time complexity is calculated.

5.2.6 Complexity Analysis

5.2.6.1 Space Complexity

Based on the implementation in the previous section, an array and a binary search tree must be maintained during a CGA run. Each element in the array needs to record a binary string with length ml . Since there are n' states preserved in the array, the space complexity of the array is $O(n'ml)$. Moreover, there are $(n - n')$ pairs of (Starting State, Ending State). Therefore, the tree has to maintain at most $(n - n')$ nodes. Similar to the array, each node in the tree needs to record a binary string with length ml , so the space complexity of the tree is $O((n - n')ml)$. Hence, the total space complexity is $O(nml)$.

5.2.6.2 Time Complexity

The extra time needed for the estimation is mainly from the time to search (or insert) the corresponding starting states in the tree (Step 4 in the previous section). Since there are in total $(n - n')$ starting states, the tree has at most $(n - n')$ nodes. The time to search each starting state (or insert a node) in the tree is $O(\log(n - n'))$ in the average case and $O((n - n'))$ in the worst case. That is, the total time complexity is $O(n \cdot \log(n))$ in the average case and

$O(n^2)$ in the worst case. However, a form of self balancing binary search tree, such as red-black trees, can be used for improving the computation time. With that form, insertions and searches are $O(\log(n))$ in the worst case. In other words, the total time complexity becomes $O(n \cdot \log(n))$ in the worst case.

5.3 Experimental Validation of the Proposed Evaluation Metric

My goal for the experiments is to investigate the usability of the proposed evaluation metric. To facilitate the experimental comparisons in the following sections, we can fix the value of some of the CGA parameters. The fixed parameters are:

- the crossover probability p_c is set to 0.5
- the mutation probability p_m is set to 0.01 (a small value)

In addition, for the purpose of validation, several fitness functions, including 1–dimensional and 3–dimensional functions, with their known optimal solutions are selected as test functions.

5.3.1 One Dimensional Fitness Functions

Eight 1–dimensional fitness functions with different degrees of difficulty for CGAs are investigated and analyzed. The eight fitness function types are: a linear function, a quadratic function, a periodic function, a fraction function, two functions with isolation optimal points (called *needle-in-a-haystack* (NIAH) [Horn, J. and Goldberg, D. E. (1995); Horn, J. (1995)]), a function consisting of unequal spaced peaks of uniform heights, and a function consisting of unequal spaced peaks of nonuniform heights [Sareni, B. and Krahenbuhl, L. (1998); Petrowski, A. (1996)]. The fitness functions and their optimal fitness values are presented in Table 5.1 (the default value for the domains is $[0, 600]$).

5.3.2 Discussion of the Sizes of Samples

One may argue how we can be sure that the size of the samples is sufficient for deriving a proper approximation. To answer this question, experiments of sample sizes with respect to

Table 5.1 The Selected Fitness Functions

Fitness Function	Optimal Value
$f_1(x) = x$	600
$f_2(x) = 360000 - x^2$	360000
$f_3(x) = 1 + \sin(x)$	2
$f_4(x) = \frac{1}{\sqrt{x+1}}$	1
$f_5(x) = \begin{cases} -x^2 + 90000, & x < 300 \\ 100000, & x = 300 \\ -(x - 600)^2 + 90000, & x > 300 \end{cases}$	100000
$f_6(x) = \begin{cases} 100000, & x = 0 \\ -(x - 300)^2 + 90000, & x \neq 0 \end{cases}$	100000
$f_7(x) = \sin^6(5\pi[x^{3/4} - 0.05])$	1
$f_8(x) = e^{-2(\ln 2)(\frac{(x/600)-0.08}{0.854})^2} \sin^6(5\pi[(x/600)^{3/4} - 0.05])$	1

encoding sizes for a problem are conducted.

For the following observations, I apply the theoretical truth, if the power n' is fixed, the value $Trace((CMS)^{n'})$ is fixed. With the setting on both the number of individuals in a generation $m = 2$ and the length of feasible solutions $l = 2$, CMS has a total of $2^4 = 16$ states. Table 5.2 lists the average results for fitness function f_2 from 20 trials of CGA runs for the number of generations ranging from 5000 to 10000 with the actual traces compared. Note that similar results are also derived for other fitness functions.

Table 5.2 The Estimated $Trace((CMS)^{n'})$ w.r.t. the Number of Generations and Power n'

No. of Gens. \ Power n'	10	12	14	16	18	20
5000	2.7331	2.5614	2.4094	2.2540	2.1660	2.0517
6000	2.7343	2.5584	2.4156	2.2564	2.1737	2.0527
7000	2.7379	2.5620	2.4104	2.2631	2.1715	2.0532
8000	2.7333	2.5692	2.4143	2.2613	2.1742	2.0565
9000	2.7397	2.5672	2.4070	2.2584	2.1716	2.0539
10000	2.7351	2.5558	2.4105	2.2708	2.1671	2.0548
Actual Traces	3.3043	3.1863	3.0771	2.9737	2.8763	2.7843

As we can see, with the same n' , no matter how many generations exist, the results for estimated $Trace((CMS)^{n'})$ are almost the same for each one, and are close to actual traces. Therefore, it is concluded that those numbers of generations are large enough for $l = 2$ and $m = 2$.

Now let the length l be set to 16. Table 5.3 shows the average results for fitness functions

f_1 and f_2 from 20 trials of CGA runs for the number of generations ranging from 5000 to 10000. Note that similar results are also derived for other fitness functions.

Table 5.3 The Estimated $Trace((CMS)^{n'})$ With Respect To the Number of Generations and Power n'

No. of Gens. \ Power n'	Fitness Function f_1				Fitness Function f_2			
	10	15	20	25	10	15	20	25
5000	7.0202	1.9653	1.2016	1.0667	6.5991	1.9553	1.3250	1.2053
6000	8.1490	2.1476	1.3326	1.0867	7.6623	1.9990	1.3516	1.1179
7000	9.7629	2.4942	1.4066	1.1031	9.2037	2.4565	1.3858	1.2133
8000	10.8016	2.7857	1.3546	1.1576	10.3304	2.6599	1.4224	1.2306
9000	12.5045	3.2925	1.4917	1.0618	11.8056	3.0480	1.5589	1.0657
10000	13.8309	3.6587	1.5685	1.2171	13.2582	3.3984	1.4045	1.1884

Table 5.3 shows that the results are different from previous results since as the number of generations increases, the estimated $Trace((CMS)^{n'})$ increases. This occurs because the numbers of samples for the 2^{32} states are relatively small. Some diagonal elements of $(CMS)^{n'}$ will not be calculated by the small numbers of samples; however, based on the data from Table 5.3, we observe that both of the $Trace((CMS)^{n'})$ converge to 1 as n' increases. A conclusion can be drawn that although the number of samples sometimes may not be large enough for large l 's, for some n' , the estimated $Trace((CMS)^{n'})$ is still referenceable. The setting of the n' depends on the setting of l and experiences.

As mentioned before, it is believed that large diagonal elements are easily estimated by the proposed metric. To verify, I have run several experiments. The result for f_2 is presented in Figure 5.5. Similar results can be derived for other fitness functions. As shown, the estimated $Trace((CMS)^{n'})$ increases rapidly before a large enough number of generations (e.g., $n = 5000$) and after, it increases asymptotically to its actual value. If we choose the fixed number of generations close to the turning point ($n = 5000$), a certain amount (percentage) of $Trace((CMS)^{n'})$ can be derived. The detailed data from Figure 5.5 is shown in Table 5.4.

Table 5.4 implies that when a state space has $2^9 = 512$ states and the (fixed) number of generations is set to 2000, more than 50% of $Trace((CMS)^{n'})$ can be estimated if n' is between 10 and 26. In addition, small n' (e.g., $n' = 10$) has better estimated precision than large n' (e.g., $n' = 26$). However, the power n' cannot be set too small since with small n' , relatively small eigenvalues are still not very close to zero, which causes some noise if the purpose is to

(a) Power $n' = 10$ (b) Power $n' = 26$

Figure 5.4 The relationship between estimated trace values, actual trace values, and numbers of generations with $m = 3$, $l = 3$

predict a degree of convergence. In other words, $(CMS)^{n'}$ with too small n' has large number of diagonal elements being not very close to zero, which makes the estimation difficult for the proposed metric because the sample size is relatively small with respect to the number of states in the state space.

5.3.3 Discussion of the Precision of the Estimation

Now, let us compare the estimated $Trace((CMS)^{n'})$ of fitness functions and their actual values. In order to accomplish this comparison, I have run the experiments on a High-Performance-Computer (HPC: hpc-class.iastate.edu). The maximal order of the transition matrices (square matrices) estimated for the actual trace values is 2^{10} . In this case, we can either set $(m, l) = (2, 5)$, or $(m, l) = (5, 2)$. Both of them draw the same conclusion. Table 5.5 presents the estimated and actual values of $Trace((CMS)^{n'})$ for $(m, l) = (2, 5)$ with the (fixed) number of generations being set to 10000. Other settings with $ml \leq 10$ have similar results.

In Table 5.5, if we compare the estimated $Trace((CMS)^{n'})$ with n' being fixed, we get that all of the estimated $Trace((CMS)^{n'})$ have close percentages of accuracy. The rates for

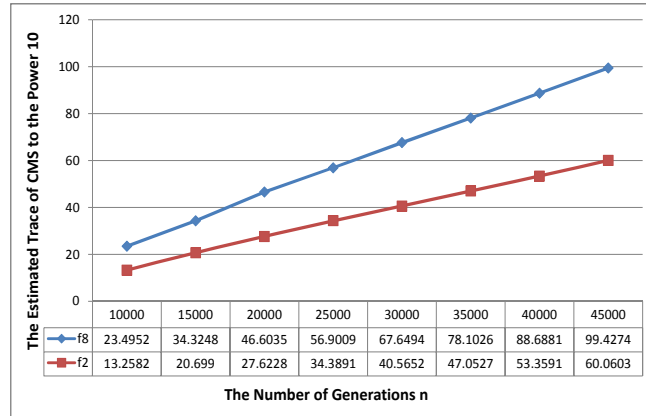


Figure 5.5 The relationship between estimated traces and numbers of generations

$\frac{\text{The Estimated Value}}{\text{The Actual Value}}$ are between 50% and 60%. As we can see, those rates are larger than the percentages of visited states for most n' . This coincides with the earlier conclusion that the states with large diagonal elements tend to be visited first. With this feature, we can be sure that if a fitness function has larger trace value for a certain n' , the proposed evaluation metric will not underestimate its $\text{Trace}((\text{CMS})^{n'})$ compared to the estimated traces of other fitness functions. For the same reason, small n' has better estimated precision than large n' .

However, a concern may be raised that the discussion and observation above are based on the condition that the number of samples is greater than the number of states. The relation between estimated $\text{Trace}((\text{CMS})^{n'})$ with a fixed n' and a number of generations n with n smaller than a predefined number of states is not specified. My empirical investigation shows that the estimated $\text{Trace}((\text{CMS})^{n'})$ and the number of generations n are linearly related if n is less than the number of states. Figure 5.5 illustrates their relationship with respect to f_2 and f_8 when the setting is $(n', m, l) = (10, 2, 16)$. That means, the estimated traces with respect to fitness functions should have close percentages of accuracy with n smaller than the number of states, if they have close percentages of accuracy with n larger than or equal to the number of states.

5.3.4 Discussion of the Convergence Trend

Another feature in the data shown in Table 5.3 is that with the number of generations being fixed, the estimated $Trace((CMS)^{n'})$ decreases as n' increases. Therefore, the convergence trend of the estimated $Trace((CMS)^{n'})$ remains the same as the convergence trend of the theoretical value of $Trace((CMS)^{n'})$. This feature can be applied to evaluate the degrees of convergence asserted earlier in this dissertation.

To analyze the convergence trends of the estimated $Trace((CMS)^{n'})$ with respect to the fitness functions, the number of individuals within a generation m is set to 2, the length of an individual (a feasible solution) l is set to 16, the number of generations is fixed to 10000, and the power n' of $(CMS)^{n'}$ is set from 10 to 24, respectively. The derived result, shown in Figure 5.6, is the average over 40 trials of CGA runs. For the purpose of verification, a set of numerical data representing the range for numbers of generations at which the CGA with best solution maintained over time has probability of 80% to reach the global (or near) optimal solution, shown in Table 5.6, is derived from 1000 trials.

Figure 5.6 shows the relationship between n' and the estimated $Trace((CMS)^{n'})$, and demonstrates that no matter what type the fitness function, as n' increases, the estimated $Trace((CMS)^{n'})$ decreases. All of the estimated values of $Trace((CMS)^{n'})$ approach to a value between 0 and 1 with increasing n' . This result coincides with Theorem 8 and discussions in both Section 5.3.2 and Section 5.3.3. Moreover, the fitness functions are clustered based on their difficulties for CGAs. Depending on different scales for clustering, two or more clusters can be identified in the figure. In my opinion, fitness functions f_1, f_2, f_4, f_5 , and f_6 can be considered to be in the same cluster, fitness functions f_3 and f_8 are in another, and f_7 is in the other. Compared with Table 5.6, we obtain that except for the fitness function f_4 , the clustering in Figure 5.6 is consistent with the set of numerical data in Table 5.6. Since the landscape of the fitness function f_4 is almost flat, the fitness value is not sensitive with respect to the modifications of the value x . With this type of fitness function, proportional selection cannot discriminate “good” solutions from “bad” ones. Therefore, during a CGA run, most states have large probabilities of remaining the same in the next generation. In other words,

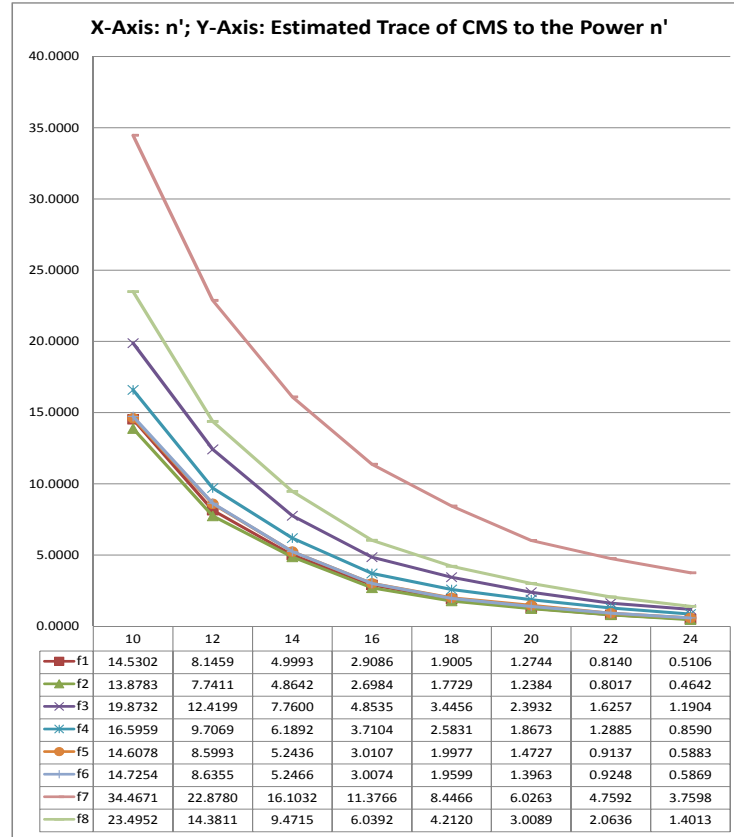


Figure 5.6 The convergence trends of $Trace((CMS)^{n'})$

this type of fitness function results in a transition matrix which has a large number of diagonal entries which are not close to 0 and have values close to each other. This feature will cause problems for the proposed methodology since there may not be enough samples to reach all of the states and accurately estimate $Trace((CMS)^{n'})$, $n' \in N$. However, besides f_4 , the clustering shows that the proposed evaluation metric has the ability to distinguish the degrees of difficulty of the CGA.

5.3.5 Discussions on the Confidence Intervals

Further investigation on the relationship between n' and the estimated value $Trace((CMS)^{n'})$ generated by the proposed evaluation metric is conducted. The distributions for the estimated

traces generated by the metric with respect to f_1 to f_8 are illustrated in Figure 5.7 to Figure 5.14, respectively. The settings remain the same as in the previous discussion on the convergence trend. The 95% standard confidence intervals (SCIs) are computed for all of the fitness functions f_1 to f_8 (see Figure 5.15 to Figure 5.22). Note that the evaluation metric cannot derive 100% of true trace values (see Section 5.3.3). The 95% SCIs merely can tell us something about the true mean value (i.e., the mean of all possible outcomes) generated by the metric, which is certain percentage of the true trace value.

The SCI tends to shrink as n' increases. Among f_1 to f_8 , f_7 has the largest confidence intervals for $n' = 10$ and $n' = 24$, whose lengths are around 2.02 and 0.60 (less than 1), respectively. In Figure 5.23, the 95% SCIs for f_1 , f_3 , and f_7 representing the three clusters mentioned in the previous discussion are compared. The SCIs for f_1 do not overlap with the SCIs for f_3 . Similarly, the SCIs for f_3 do not overlap with the SCIs for f_7 . It confirms that the clustering of fitness functions with respect to the estimated values of $Trace((CMS)^{n'})$ derived by the evaluation metric is valid and has a strong discrimination. In reality, except f_4 , f_7 is the most difficult fitness function among f_1 to f_8 for the CGA with the setting $(m, l) = (2, 16)$ (see Table 5.6).

In the literature, researchers who investigate confidence intervals have asserted that the actual coverage probability of the SCI may sometimes not be equal to the nominal level claimed, especially when the sample is small [Brown, L. D. and Cai, T. T. and DasGupta, A. (2001)]. Several alternative intervals are recommended. The Wilson interval (WI) is one of them and used to compare with the SCI here (see Figure 5.24 to Figure 5.31). As we can see, the WI also tends to shrink as n' increases. Most of the mean values and the lengths of WIs are slightly smaller than those of SCIs. Figure 5.32 shows the mean values of WI and SCI for the fitness functions f_1 and f_8 , respectively. Carefully comparing the WIs and SCIs for the eight fitness functions, we can get that when $n' \geq 14$, the two intervals overlap each other. No matter which interval (WI or SCI) is used for the comparison, the clusters formed earlier are still valid.

Table 5.4 The Data of Figure 5.4(a) and 5.4(b)

No. of Gens	Power $n' = 10$			Power $n' = 26$		
	Estimated Value	Actual Value	%	Estimated Value	Actual Value	%
1000	3.1428	5.8952	53.31	1.6304	3.8432	42.42
1500	3.4209	5.8952	58.03	1.8348	3.8432	47.74
2000	3.6386	5.8952	61.72	1.9571	3.8432	50.92
2500	3.7500	5.8952	63.61	1.9859	3.8432	51.67
3000	3.8105	5.8952	64.64	2.0407	3.8432	53.10
3500	3.8867	5.8952	65.93	2.0509	3.8432	53.36
4000	3.8913	5.8952	66.01	2.0675	3.8432	53.80
4500	3.9689	5.8952	67.32	2.1166	3.8432	55.07
5000	4.0181	5.8952	68.15	2.1479	3.8432	55.89
5500	4.0554	5.8952	68.79	2.1935	3.8432	57.08
6000	4.0788	5.8952	69.19	2.1977	3.8432	57.18
6500	4.1034	5.8952	69.61	2.2439	3.8432	58.39
7000	4.1206	5.8952	69.90	2.2226	3.8432	57.83
7500	4.1298	5.8952	70.05	2.2292	3.8432	58.00
8000	4.1457	5.8952	70.32	2.2528	3.8432	58.62
8500	4.1307	5.8952	70.07	2.2546	3.8432	58.66
9000	4.1564	5.8952	70.50	2.2398	3.8432	58.28
9500	4.1780	5.8952	70.87	2.2620	3.8432	58.86
10000	4.1642	5.8952	70.64	2.2591	3.8432	58.78

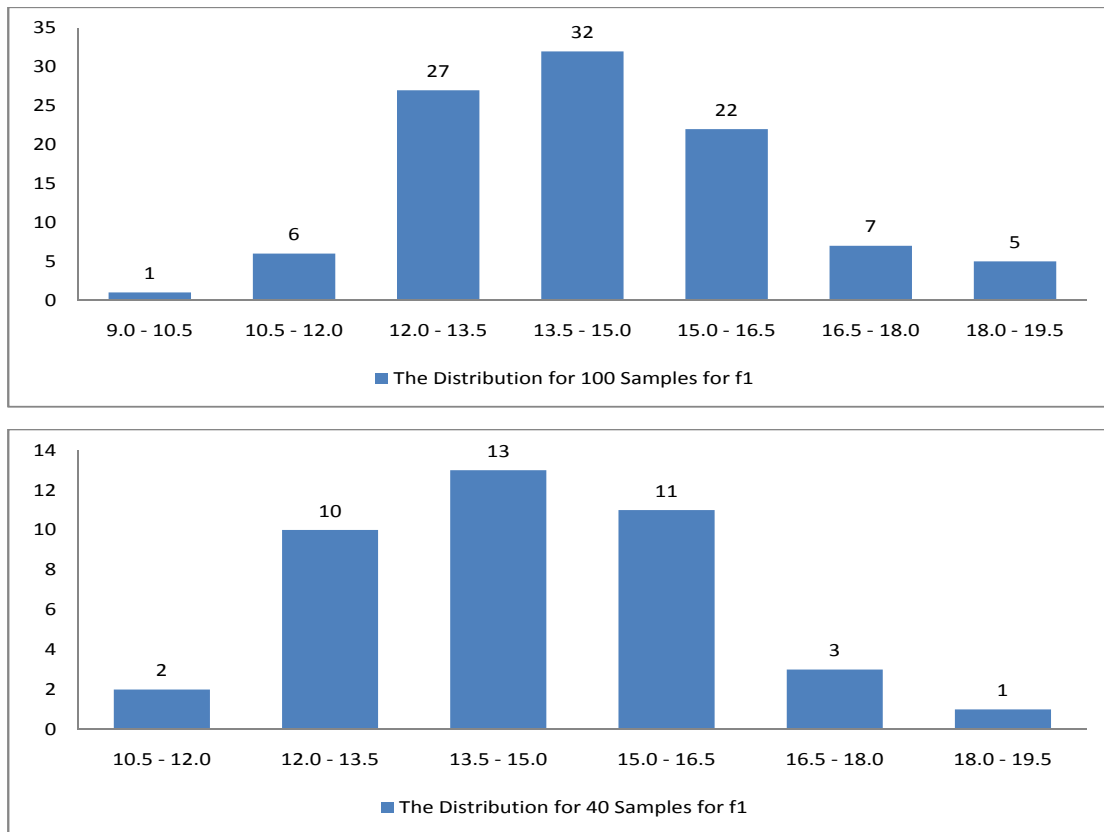
Figure 5.7 The Distribution with respect to f_1

Table 5.5 Comparison with Actual Trace Values

The Power n'	Fitness f_2			Fitness f_3		
	Estimated Value	Actual Value	%	Estimated Value	Actual Value	%
10	11.2222	19.8676	56.48	10.7942	19.9873	54.00
12	9.6180	18.1384	53.03	9.3883	18.3296	51.22
14	8.2574	16.6158	49.70	8.3272	16.8629	49.38
16	7.1721	15.2567	47.01	7.2876	15.5563	46.85
18	6.2690	14.0383	44.66	6.5492	14.3887	45.52
20	5.4732	12.9436	42.29	5.9391	13.3429	44.51
22	4.9405	11.9583	41.31	5.3703	12.4042	43.29
24	4.3463	11.0701	39.26	4.8289	11.5599	41.77
26	3.9340	10.2680	38.31	4.4362	10.7989	41.08
Percentages of Visited States (%)	42.29			35.25		
The Power n'	Fitness f_4			Fitness f_5		
	Estimated Value	Actual Value	%	Estimated Value	Actual Value	%
10	10.9152	19.8817	54.90	11.7743	19.9793	58.93
12	9.0834	18.1525	50.04	10.0742	18.2940	55.07
14	7.7609	16.6331	46.66	8.7976	16.8133	52.33
16	6.7049	15.2781	43.89	7.6219	15.4948	49.19
18	5.8291	14.0643	41.45	6.7025	14.3155	46.82
20	5.0959	12.9746	39.28	5.3872	13.2577	40.63
22	4.5871	11.9946	38.24	5.1549	12.3069	41.89
24	4.0685	11.1118	36.61	4.8047	11.4504	41.96
26	3.6454	10.3154	35.34	4.4481	10.6772	41.66
Percentages of Visited States (%)	35.44			42.87		
The Power n'	Fitness f_7			Fitness f_8		
	Estimated Value	Actual Value	%	Estimated Value	Actual Value	%
10	10.3222	20.2025	51.09	10.9513	20.2646	54.04
12	9.1415	18.6256	49.08	9.7161	18.7078	51.94
14	8.2737	17.2356	48.00	8.7522	17.3387	50.48
16	7.5026	16.0042	46.88	7.9443	16.1280	49.26
18	6.9316	14.9097	46.49	7.1592	15.0536	47.56
20	6.2590	13.9342	44.92	6.6624	14.0971	47.26
22	5.7717	13.0621	44.19	6.2506	13.2431	47.20
24	5.4395	12.2804	44.29	5.7019	12.4782	45.69
26	5.1078	11.5777	44.11	5.4555	11.7910	46.27
Percentages of Visited States (%)	30.37			31.74		

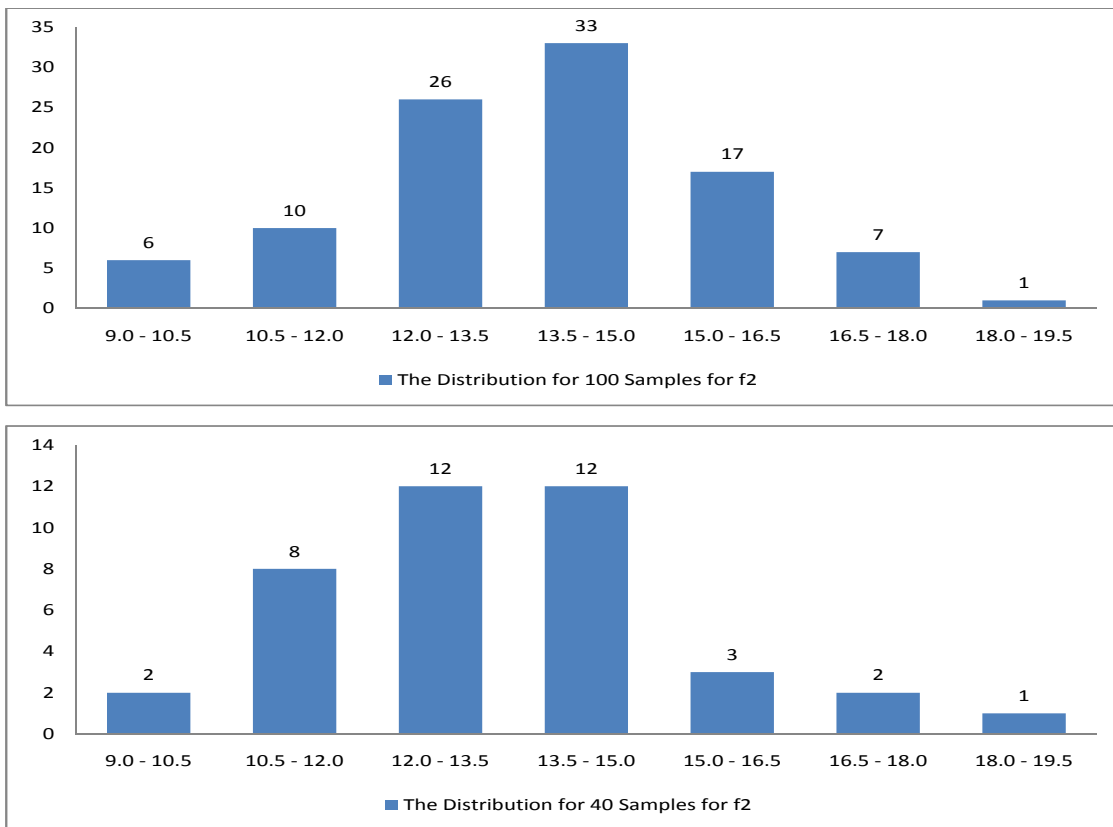


Figure 5.8 The Distribution with respect to f_2

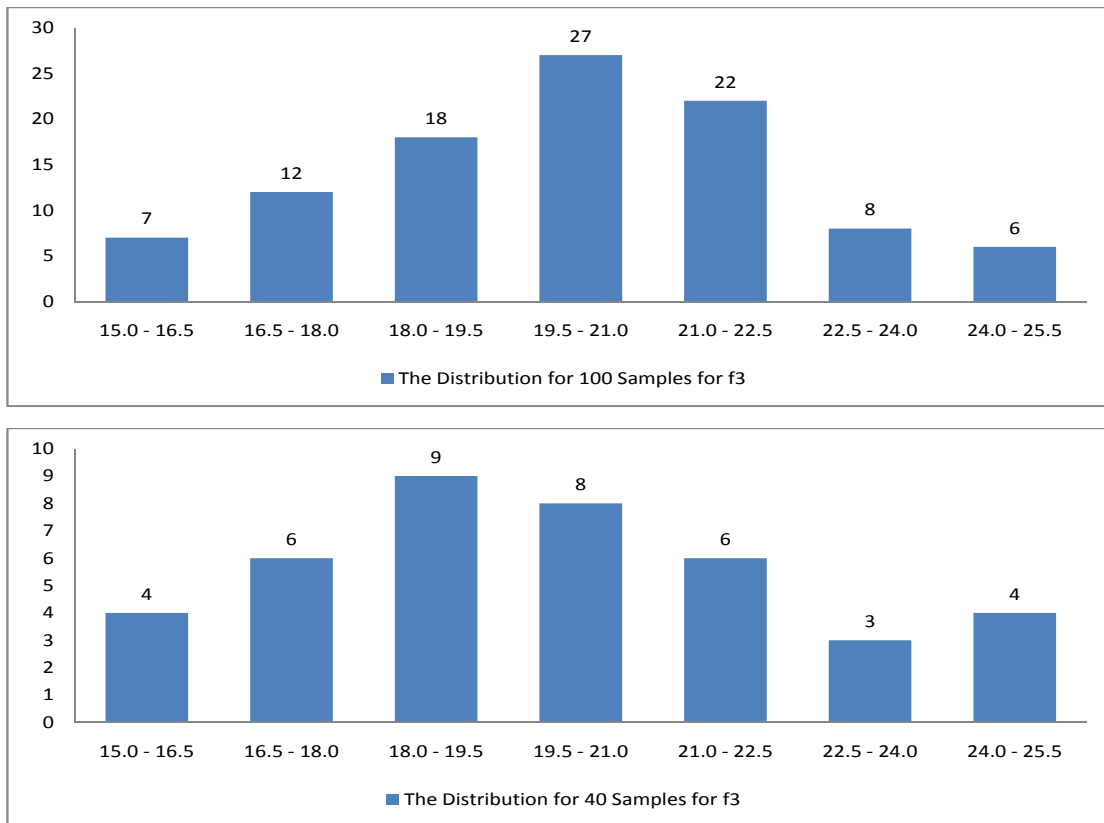


Figure 5.9 The Distribution with respect to f_3

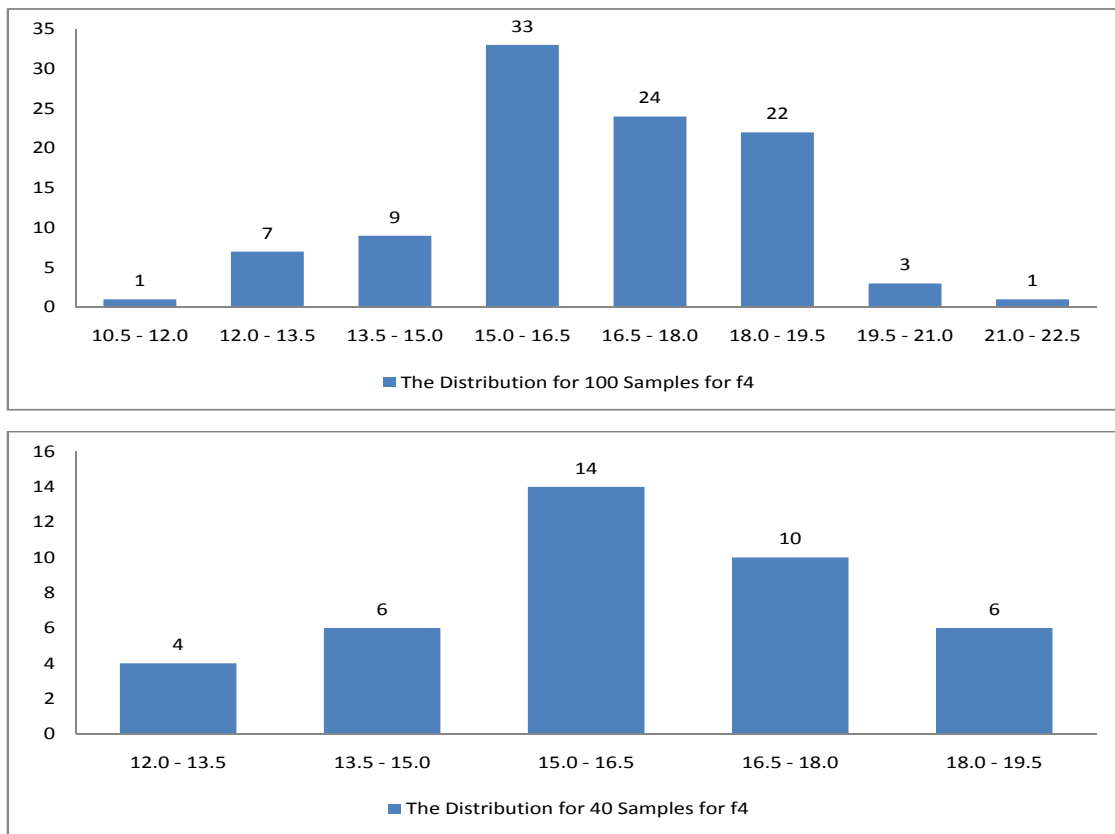


Figure 5.10 The Distribution with respect to f_4

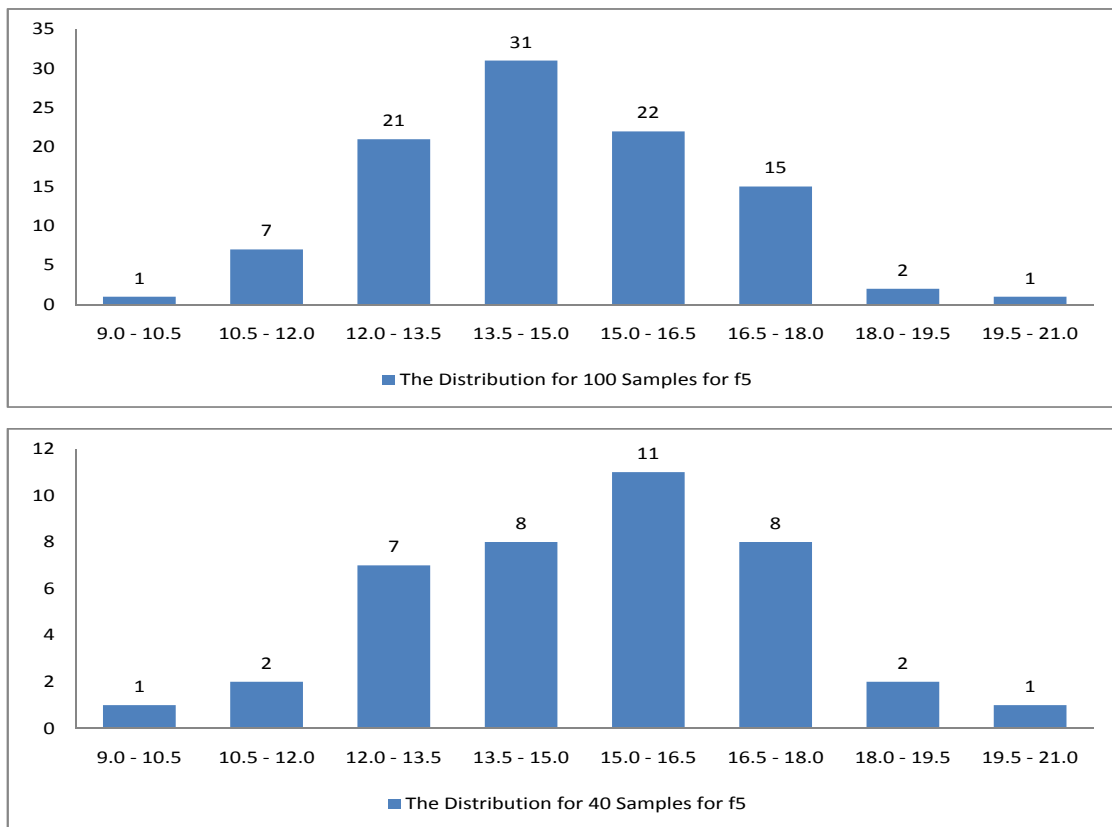


Figure 5.11 The Distribution with respect to f_5

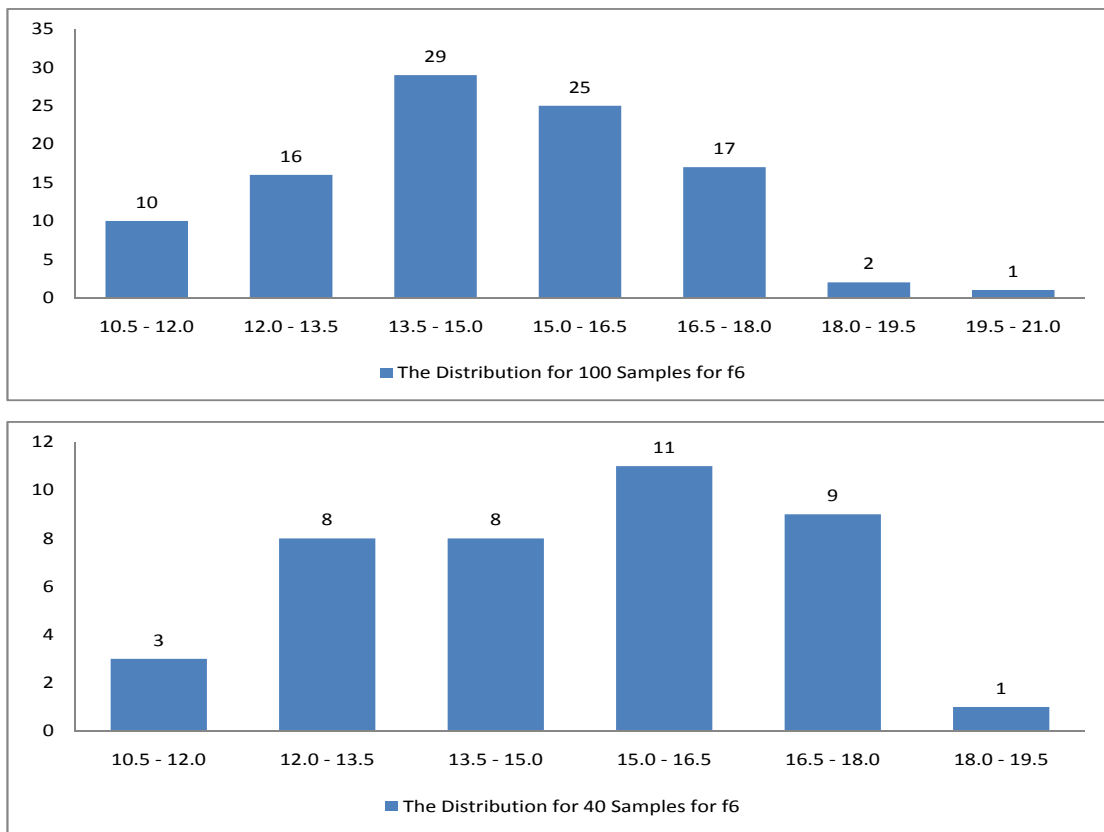


Figure 5.12 The Distribution with respect to f_6

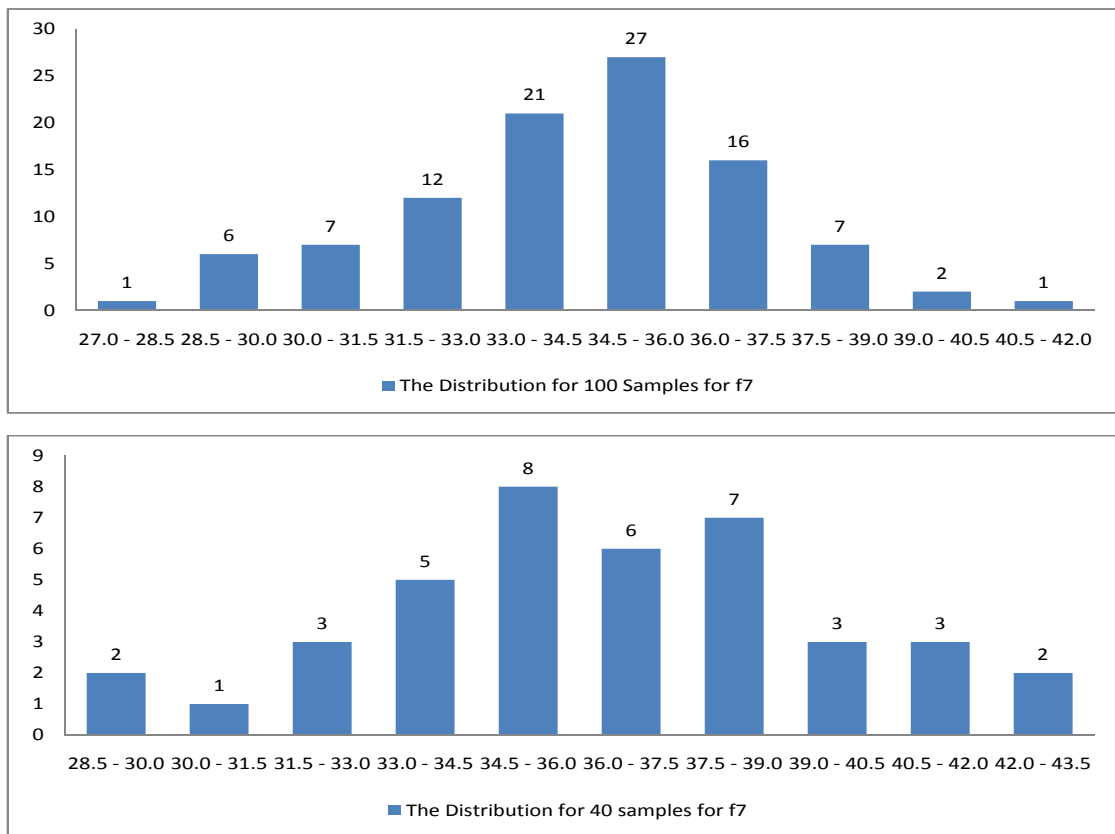


Figure 5.13 The Distribution with respect to f_7

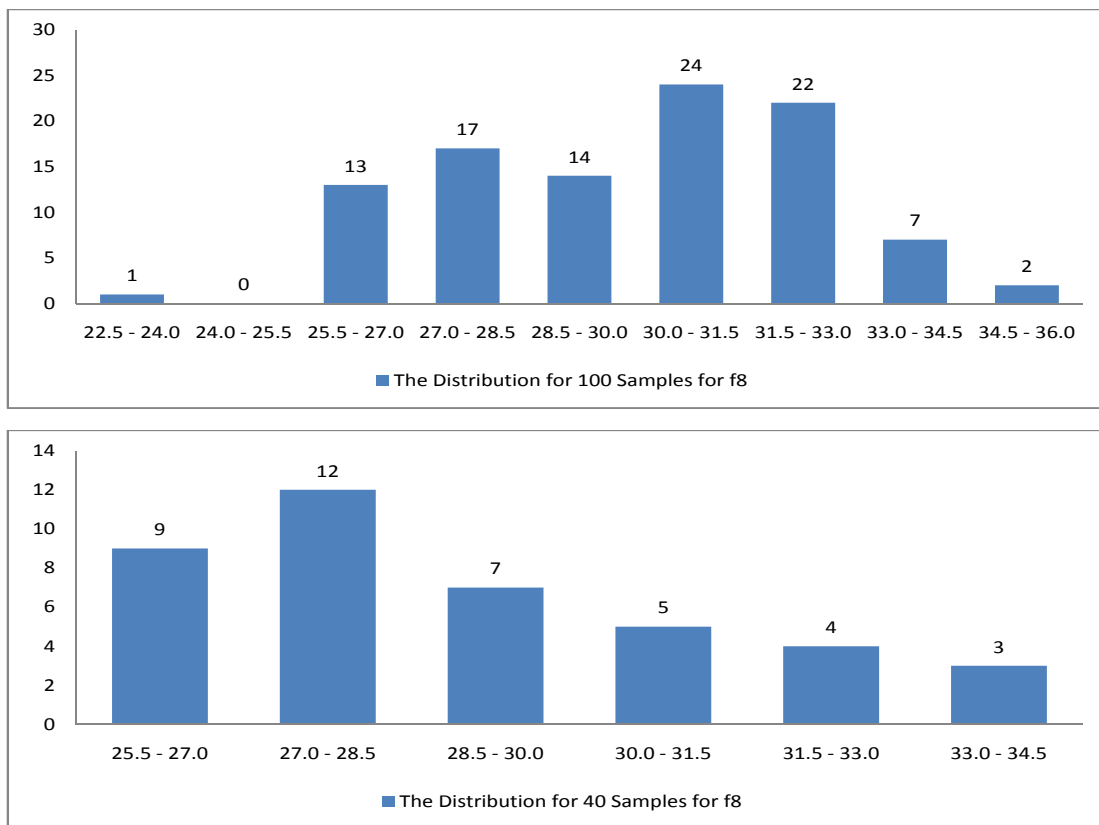


Figure 5.14 The Distribution with respect to f_8

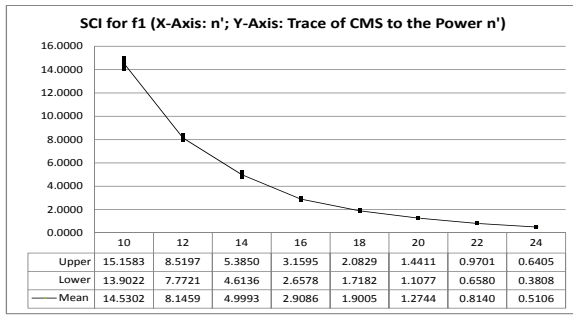


Figure 5.15 95% SCI for f_1

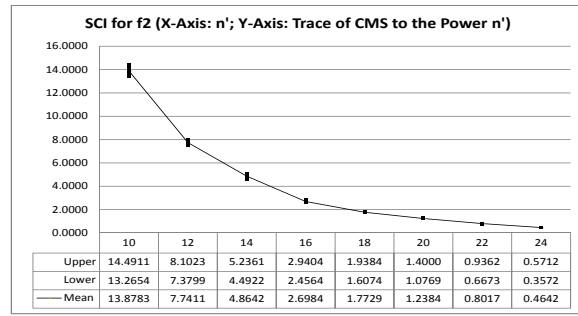


Figure 5.16 95% SCI for f_2

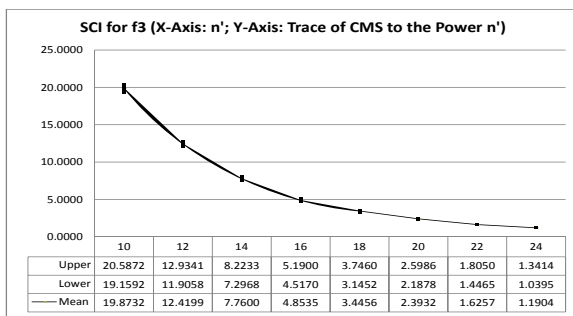


Figure 5.17 95% SCI for f_3

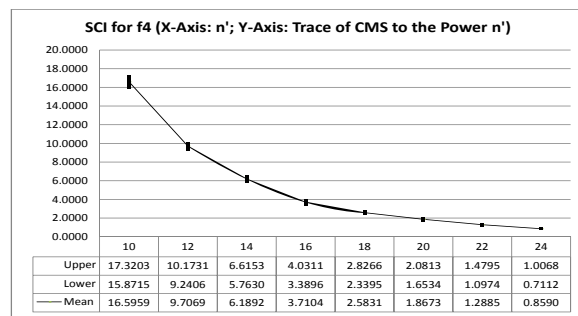


Figure 5.18 95% SCI for f_4

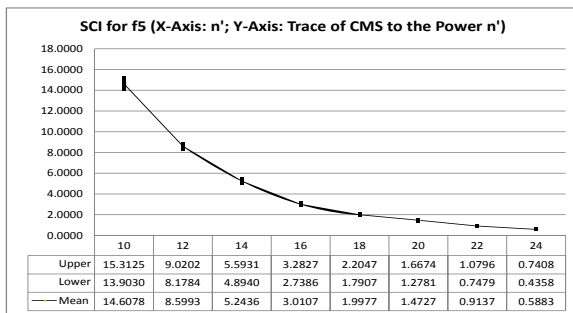


Figure 5.19 95% SCI for f_5

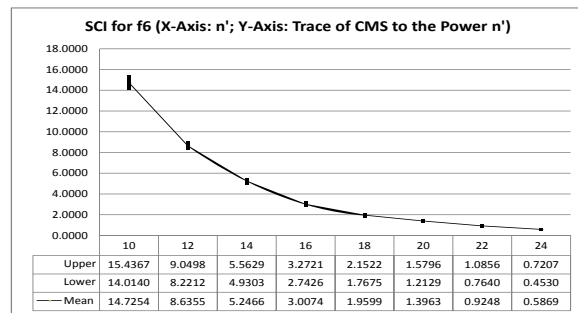


Figure 5.20 95% SCI for f_6

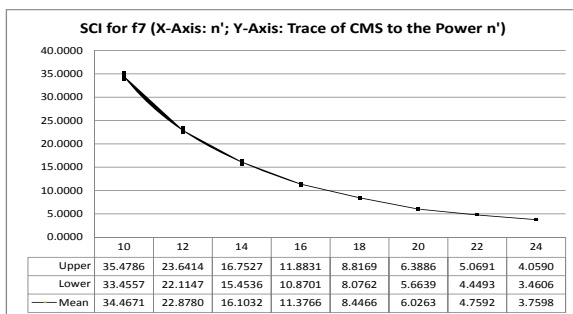


Figure 5.21 95% SCI for f_7

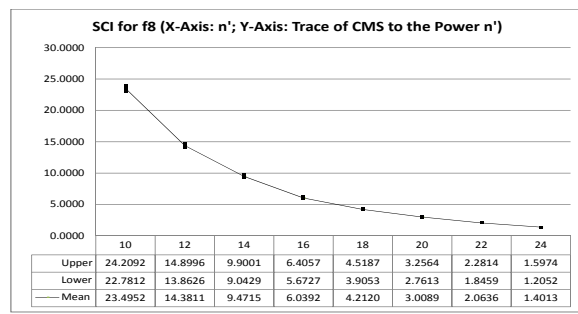


Figure 5.22 95% SCI for f_8

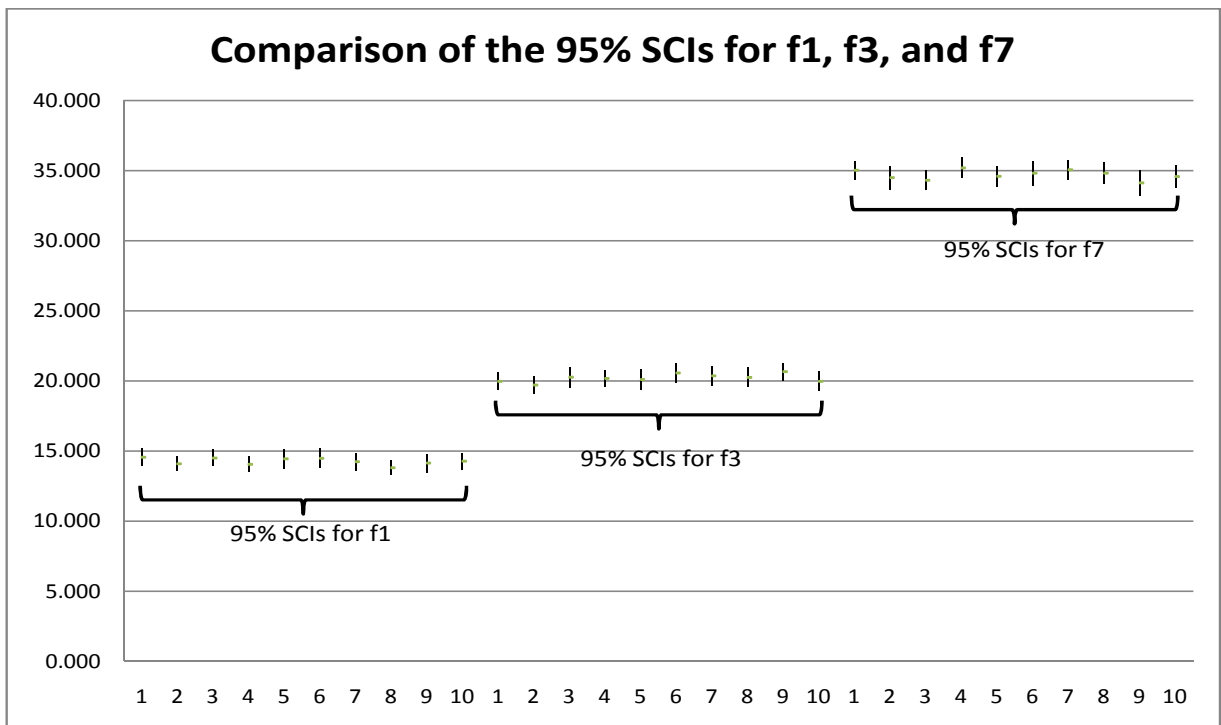


Figure 5.23 The Comparison of 95% SCIs for the Estimated Trace Values Generated by the Proposed Evaluation Metric

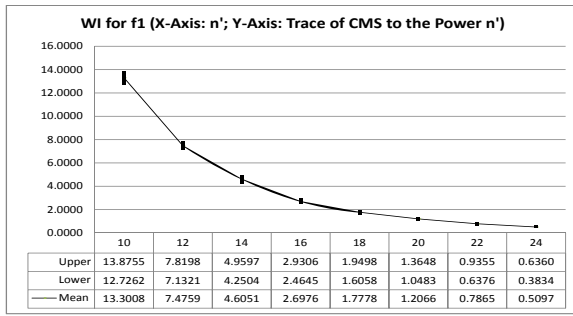


Figure 5.24 95% WI for f_1

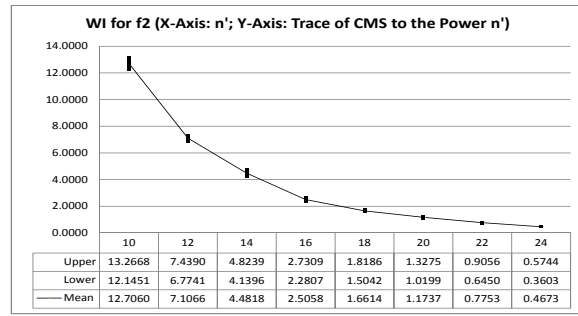


Figure 5.25 95% WI for f_2

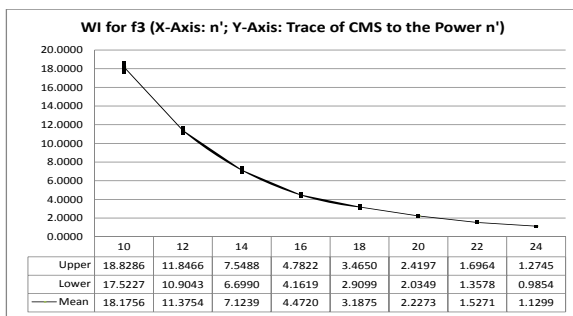


Figure 5.26 95% WI for f_3

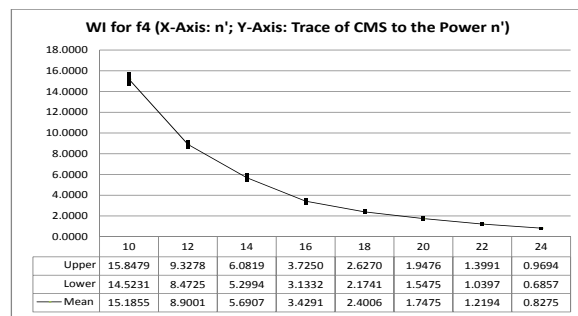


Figure 5.27 95% WI for f_4

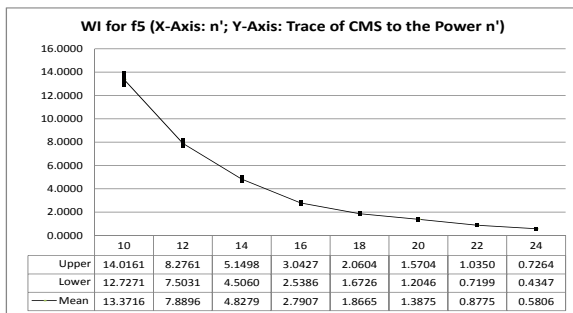


Figure 5.28 95% WI for f_5

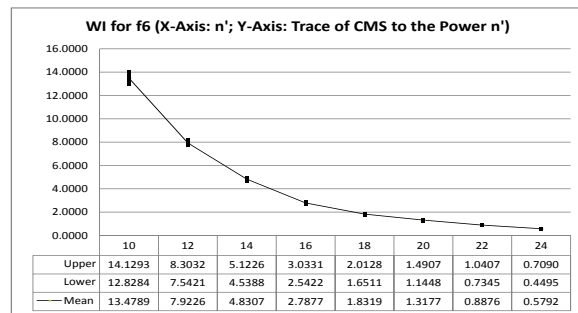


Figure 5.29 95% WI for f_6

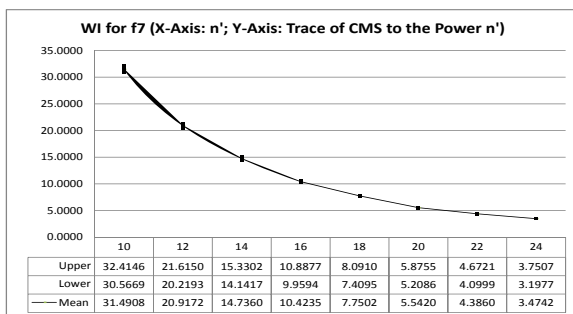


Figure 5.30 95% WI for f_7

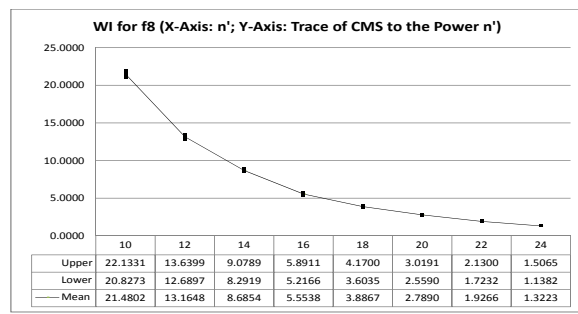


Figure 5.31 95% WI for f_8

Table 5.6 The Range for Numbers of Generations at which the CGA with Best Solution Maintained Over Time Has Probability of 80% (out of 1000 Trials) to Reach the Global (or Near) Optimal Solution

Fitness Function	The Range
f_1	15500 – 16000
f_2	14500 – 15000
f_3	24000 – 24500
f_4	> 70000
f_5	16000 – 16500
f_6	16500 – 17000
f_7	57000 – 57500
f_8	44000 – 44500

5.3.6 Discussions on the Smallest Number of Samples Needed for the Estimation

Through above discussions, we know that the proposed evaluation metric has the ability to determine the order of fitness functions sorted by the difficulty levels for CGAs. This subsection simply discusses the smallest number of samples needed for the estimation. Similar as before, the fitness functions, except f_4 , with the setting $(n', m, l) = (10, 2, 16)$ are investigated. Assume that for the same n' , using a larger number of samples derives the estimated $Trace((CMS)^{n'})$ closer to the actual trace value. The order of fitness functions when the number of generations $n = 20000$ is compared with smaller numbers of generations. When $n = 20000$, the order of the fitness functions from the most difficult one (i.e., the one has the largest $Trace((CMS)^{10})$) to the easiest one is

$$f_7 \rightarrow f_8 \rightarrow f_3 \rightarrow f_6 \rightarrow f_5 \rightarrow f_1 \rightarrow f_2$$

Note that when $n = 10000$, the order of fitness functions is exactly the same as the above order. Figure 5.33 shows the results obtained from using smaller numbers of generations.

As we can see, when the number of generations increases, the estimated traces increase linearly. This result is similar as the result in Figure 5.5. From $n = 500$ to $n = 8500$, the two lines representing the trace values for f_5 and f_6 are tangled. This is also the case when n is around 20000. When $n = 500$, the traces for f_7 and f_8 already have larger values than

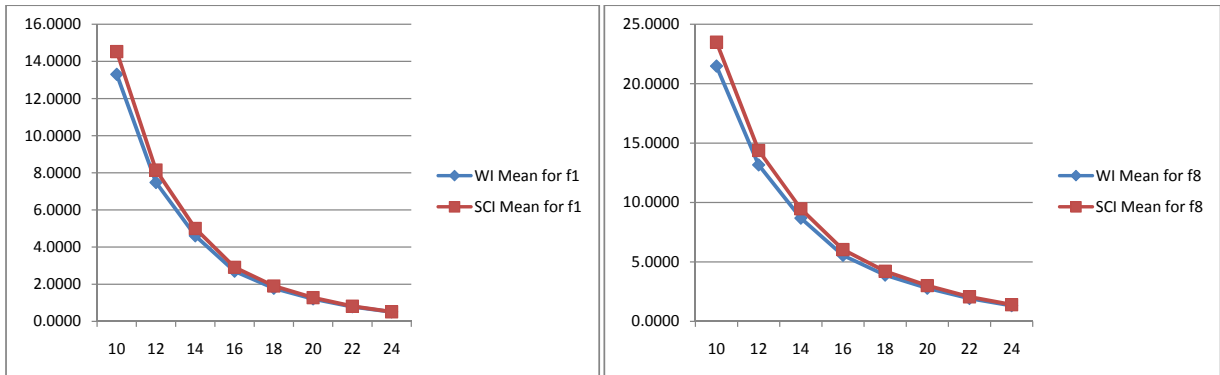


Figure 5.32 The means of WI and SCI for fitness functions f_1 and f_8 , respectively

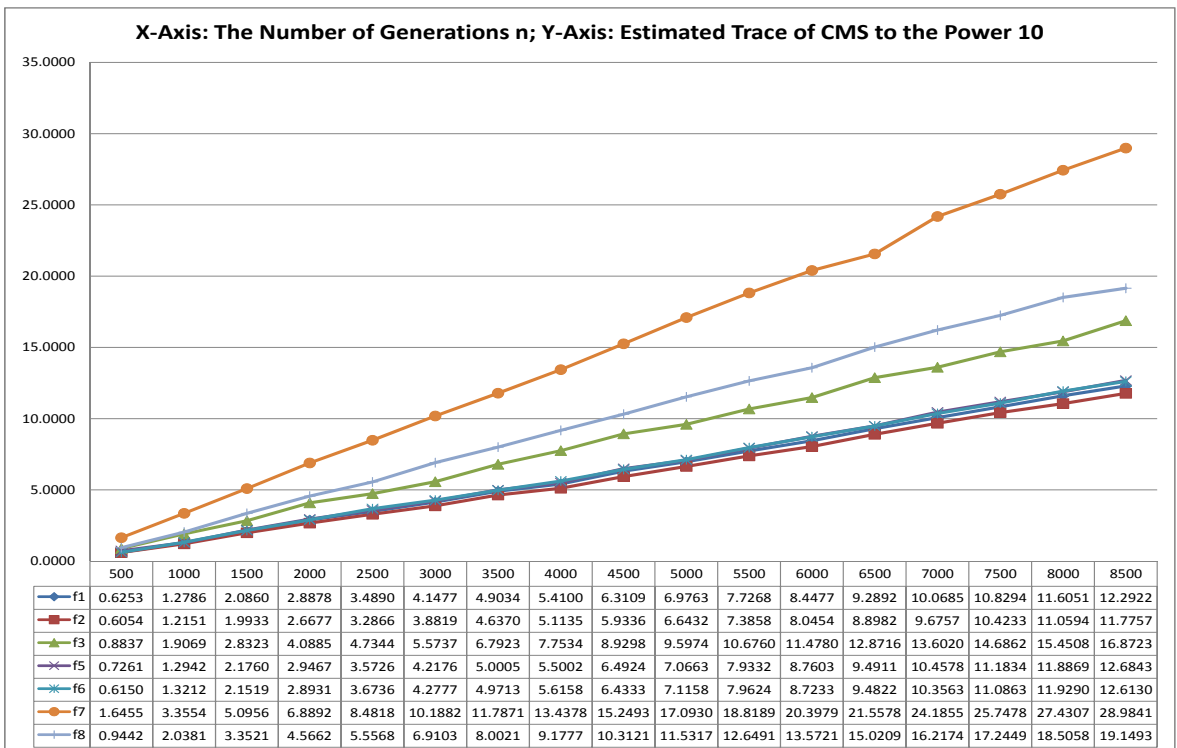


Figure 5.33 The order of fitness functions w.r.t. smaller numbers of generations

the traces for other fitness functions; however, the difference of the traces for any two fitness functions is very small, which may cause problems. For instance, we cannot know whether the fitness function which has a smaller trace is actually the easier one or not. When $n = 1000$, we get the exact same order of fitness functions presented above. This shows that the smallest number of samples needed for the setting $(n', m, l) = (10, 2, 16)$ can be around 1000. Compared with the number of states (2^{32}) and the number of feasible solutions in the solution space (2^{16}), it is a small number.

5.3.7 Multi-Dimensional Fitness Functions

In order to present more evidence for verifying my proposed methodology, five 3-dimensional fitness functions are selected, including Ackley's, Rastrigin's, and Sphere functions, with different degrees of difficulty for CGAs [Tang, K. and Yao, X. and Suganthan, P. N. and MacNish, C. and Chen, Y. P. and Chen, C. M and Yang, Z. (2007)]. Ackley's, Rastrigin's, and Sphere functions are designed to be minimized. The selection technique utilized in all experiments in this dissertation, roulette wheel selection, enables a CGA to maximize fitness functions. Hence, those fitness functions are slightly adjusted in this experiment. The five fitness functions investigated are in Table 5.7. Intuitively, the fitness functions f_9 and f_{10} are relatively simple

Table 5.7 The Selected 3-Dimensional Fitness Functions and Their Features

Fitness Function	Optimal Value
$f_9(\mathbf{x}) = (\sum_{i=1}^3 x_i) + 16$, where $x_1, x_2, x_3 \in [-5, 5]$ (Sphere Function - Unimodal and Separable)	31
$f_{10}(\mathbf{x}) = -(\sum_{i=1}^3 x_i^2) + 76$, where $x_1, x_2, x_3 \in [-5, 5]$ (Ackley's Function 1 - Multimodal, and Non-separable)	76
$f_{11}(\mathbf{x}) = -(-20e^{-0.2\sqrt{\frac{1}{3}\sum_{i=1}^3 x_i^2}} - e^{\frac{1}{3}\sum_{i=1}^3 \cos(2\pi x_i)} + 20 + e) + 15$, where $x_1, x_2, x_3 \in [-5, 5]$ (Ackley's Function 2 - Multimodal, and Non-separable)	15
$f_{12}(\mathbf{x}) = -(-20e^{-0.2\sqrt{\frac{1}{3}\sum_{i=1}^3 x_i^2}} - e^{\frac{1}{3}\sum_{i=1}^3 \cos(2\pi x_i)} + 20 + e) + 23$, where $x_1, x_2, x_3 \in [-32.768, 32.768]$ (Rastrigin's Function - Multimodal, and Separable)	23
$f_{13}(\mathbf{x}) = -(\sum_{i=1}^3 (x_i^2 - 10\cos(2\pi x_i) + 10)) + 135$, where $x_1, x_2, x_3 \in [-5, 5]$	135

among the five, since both of them only have one optimal solution. Comparing the features of the rest of the fitness functions, Ackley's function is more difficult for a CGA than Rastrigin's

function, since Ackley’s function is not only multimodal, but also non-separable (i.e., it cannot be rewritten as a sum of several functions of just one variable) [Hadley, G. (1964)]. Separable functions can be optimized for each variable in turn. Non-separable functions are more difficult to optimize as the accurate search direction depends on two or more genes. The result of this experiment, shown in Figure 5.34, is the average from 40 trials of CGA runs. The confidence intervals with respect to the settings of n' are similar to the results derived for 1–dimensional fitness functions. From Figure 5.34, the estimated trace derived by f_{12} is larger than that of

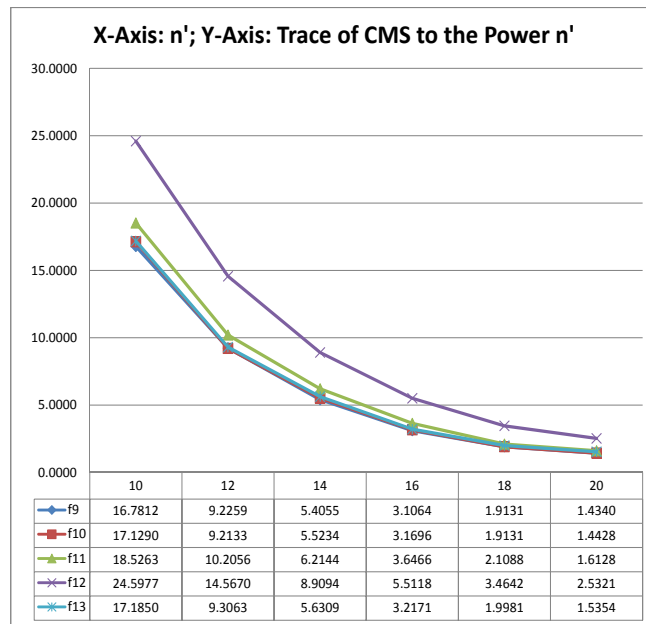


Figure 5.34 The convergence trends of $Trace((CMS)^{n'})$ for multi-dimensional fitness functions

f_{11} since the range $[-32.768, 32.768]$ includes more peaks than the range $[-5, 5]$. The degree of difficulty of f_{12} is higher than that of f_{11} for a CGA. Moreover, when fitness functions are multi-dimensional, the traces estimated by the proposed evaluation metric still coincide with the earlier assertion that more difficult fitness functions, with respect to a CGA, result in higher estimated traces. The proposed evaluation metric has the ability to determine the best fitness function for a CGA among a set of fitness functions.

5.3.7.1 Observations on the Experiments

Due to the fact that usually the sample size is relatively small compared to the state space size, the estimated $Trace((CMS)^{n'})$ is only a small percentage of the actual $Trace((CMS)^{n'})$. The criterion (i.e., $|\text{Obtained Value} - 1|$) discussed in Section 5.2.3 for determining a degree of convergence should be slightly adjusted. Since the actual $Trace((CMS)^{n'})$ is greater than or equal to 0 for all $n \in N$, any percentage of $Trace((CMS)^{n'})$ should be greater than or equal to 0. The criterion can be modified to $|\text{Obtained Value} - 0|$ (i.e., Obtained Value) to reflect that the smaller the estimated $Trace((CMS)^{n'})$, the higher the degree of convergence.

From the experiments, the probability that multiple encoded points have equal fitness values is low, even for periodic functions. That is, in the real world, there is usually one global optimal solution in the search space of CGAs. Different settings of m and l may cause different orders of fitness functions sorted by difficulty to a certain CGA configuration. The reason is that the number of points encoded in a search space for discovering the landscape of a fitness function is based on l . Small l implies a small number of points encoded in a search space. In that case, a non-smooth fitness function may become a smooth one due to the lack of points. The setting of m impacts probabilities of selecting individuals. Another observation is that different domains of a fitness function encoded in a search space have a high possibility to result in different degrees of difficulties for a CGA, which is because more of the landscape of the fitness function, such as peaks, and valleys, is included in a larger domain.

In addition, before running a CGA with the evaluation metric, one has to check:

1. Sensitivity of the fitness function;
2. Fitness values of the encoding 0000...0 and 1111...1;
3. If the isolation points are known, the fitness values of those points should be computed in advance.

Checking the sensitivity of the fitness function helps researchers determine whether or not the defined fitness function has a low discrimination for “good” and “bad” solutions. In real-world applications, CGAs with non-sensitive fitness functions are rare. Practitioners apply

CGAs to obtain (near) optimal solutions. If the fitness function is almost flat for a CGA, it is meaningless to spend time searching for the global optimal solution. Besides, the low discrimination fitness function can always be reformulated to improve the performance of CGAs. Through the second step, the isolation points of the fitness function in the extreme locations can be known beforehand. This step is essential because the isolation points other than $0000\dots 0$ and $1111\dots 1$ are relatively easy to produce by chance through crossover and mutation operators. Checking the extreme cases first helps CGAs to have a better chance to find the global optimal solution.

CHAPTER 6. The Estimation of Global Convergence

This chapter presents an approach to using the proposed evaluation metric in Chapter 5 for estimating the number of generations needed for the best solution generated by a CGA to converge to the global optima of a fitness function.

6.1 The Overview of the Estimation

The estimation includes the following three phases (see Figure 6.1):

- the empirical estimation on $Trace((CMS)^{n'})$ and $Trace((CMS)^{n'+k})$, for some $n', k \in N$,
- the generation of a fitting curve based on $Trace((CMS)^{n'})$ and $Trace((CMS)^{n'+k})$, and
- the derivation of the least number of generations needed (for global convergence) from the fitting curve and a small fixed ϵ .

Each phase is introduced below.

6.1.1 The Empirical Estimation on the Traces and the Fitting Curve

In the experiments of Chapter 5, estimated traces derived by the proposed evaluation metric are merely certain proportions of the actual ones. Without any other information, the exact actual trace is difficult to determine from the estimated trace. To closely estimate a trace, one has to know the number of states visited by a CGA run, which can be computed by directly counting the number of nodes in the linked list in Figure 5.3. Being aware of the number of states visited by a CGA run, the proportion of the visited states over the total number of states is derived, which is used to estimate the actual trace below.

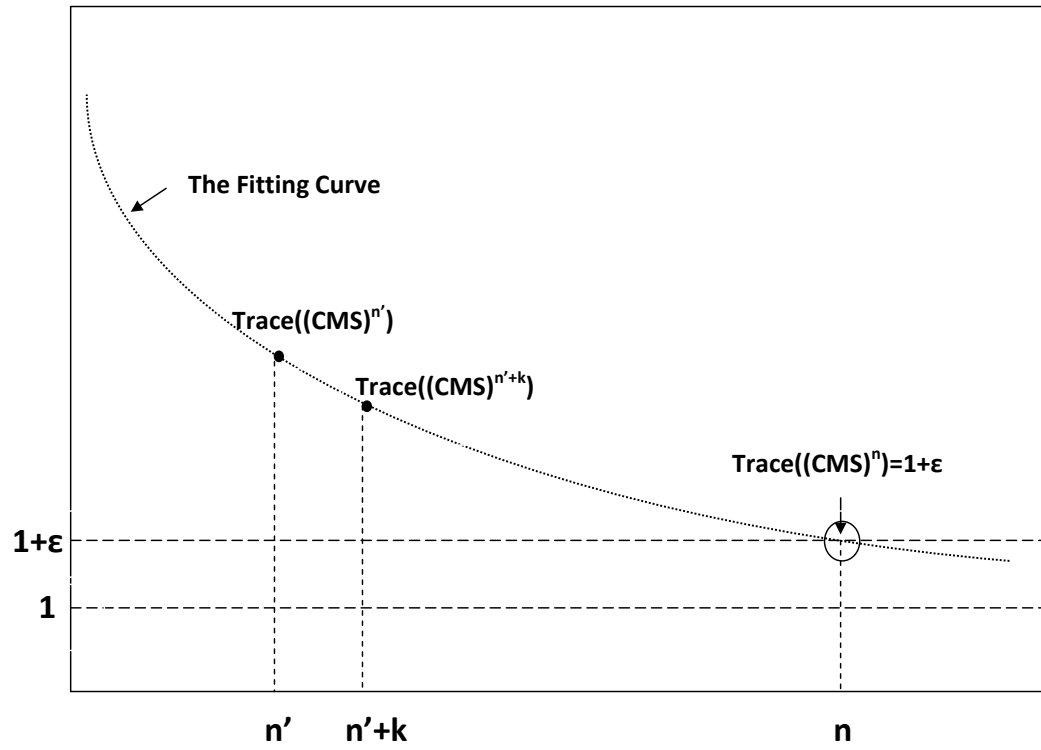


Figure 6.1 The Overview of the Estimation

Due to the limitation of the computer capability, the maximal order of the transition matrices (square matrices) estimated for the actual traces is 2^{10} . For this reason, the setting $m = 2$ is used to develop the entire estimation framework as an illustration. Similar frameworks can be derived by the same way for other settings of m . Table 6.1 and Table 6.2 show the relationships among the estimated trace, the proportion of the visited states over the total number of states, and the actual trace with respect to the fitness function $f_1(x) = x$. Note that the actual traces for the settings $l > 5$ are guessed by fitting a curve to the previous four points (i.e., $(m, l) = (2, 2), (2, 3), (2, 4)$ and $(2, 5)$).

Fitness Function: $f_1(x) = x, n' = 10$							
Settings	$m = 2, l = 2$	$m = 2, l = 3$	$m = 2, l = 4$	$m = 2, l = 5$	$m = 2, l = 6$	$m = 2, l = 7$	$m = 2, l = 8$
Number of Generations: 500							
(Average) Number of States Visited	10	27	54	89	121	157	186
Percentages of Visited States (%)	62.50	42.19	21.09	8.69	2.95	0.96	0.28
Estimated Trace by the Proposed Metric	1.9325	2.7694	3.6092	4.1131	4.4298	3.8777	3.5605
(Estimated Trace/Percentages of Visited States) (*500)	3.0919	6.5644	17.1103	47.3240	149.9540	404.6681	1254.5194
(*500)/(The Actual Trace)	0.9357	1.0933	1.5674	2.3842	4.1543	6.1644	10.5078
Number of Generations: 1000							
(Average) Number of States Visited	13	35	77	141	208	284	352
Percentages of Visited States (%)	81.25	54.69	30.08	13.77	5.08	1.73	0.54
Estimated Trace by the Proposed Metric	2.2072	3.0862	4.6629	6.0658	6.8859	6.9667	6.5767
(Estimated Trace/Percentages of Visited States) (*1000)	2.7166	5.6433	15.5027	44.0521	135.6001	401.9119	1224.4665
(*1000)/(The Actual Trace)	0.8221	0.9399	1.4202	2.2193	3.7566	6.1224	10.2561
Number of Generations: 1500							
(Average) Number of States Visited	13	39	90	176	279	398	507
Percentages of Visited States (%)	81.25	60.94	35.16	17.19	6.81	2.43	0.77
Estimated Trace by the Proposed Metric	2.2457	3.4820	5.2121	7.2691	8.6713	9.3192	9.3039
(Estimated Trace/Percentages of Visited States) (*1500)	2.7640	5.7140	14.8255	42.2932	127.3033	383.6316	1202.6409
(*1500)/(The Actual Trace)	0.8365	0.9516	1.3581	2.1307	3.5268	5.8439	10.0733
Number of Generations: 2000							
(Average) Number of States Visited	14	42	100	203	335	498	647
Percentages of Visited States (%)	87.50	65.63	39.06	19.82	8.18	3.04	0.99
Estimated Trace by the Proposed Metric	2.3166	3.5612	5.5706	7.7670	10.0028	11.1726	11.5273
(Estimated Trace/Percentages of Visited States) (*2000)	2.6475	5.4265	14.2607	39.1792	122.3023	367.5734	1167.6285
(*2000)/(The Actual Trace)	0.8012	0.9038	1.3064	1.9738	3.3882	5.5993	9.7800
Number of Generations: 5000							
(Average) Number of States Visited	15	51	139	306	570	933	1342
Percentages of Visited States (%)	93.75	79.69	54.30	29.88	13.92	5.69	2.05
Estimated Trace by the Proposed Metric	2.4505	4.0357	6.4425	9.8537	14.0102	17.9755	20.9455
(Estimated Trace/Percentages of Visited States) (*5000)	2.6139	5.0644	11.8654	32.9744	100.6768	315.6592	1022.8634
(*5000)/(The Actual Trace)	0.7911	0.8434	1.0870	1.6612	2.7891	4.8085	8.5675
The Actual Trace	3.3043	6.0044	10.9161	19.8492	36.0964 (Guess)	65.6460 (Guess)	119.3893 (Guess)

Table 6.1 The Relationships among the Estimated Trace, the Proportion of the Visited States over the Total Number of States ($n' = 10$)

As shown in both Table 6.1 and Table 6.2, the value, derived by dividing the estimated trace by the proportion of the visited states over the total number of states, is greater than the actual trace when setting $m = 2$ and $l \geq 4$. Moreover, the relationship between the value and

Fitness Function: $f_1(x) = x$, $n' = 12$							
Settings	$m = 2, l = 2$	$m = 2, l = 3$	$m = 2, l = 4$	$m = 2, l = 5$	$m = 2, l = 6$	$m = 2, l = 7$	$m = 2, l = 8$
Number of Generations: 500							
(Average) Number of States Visited	10	27	54	88	121	156	185
Percentages of Visited States (%)	62.50	42.19	21.09	8.59	2.95	0.95	0.28
Estimated Trace by the Proposed Metric	1.8528	2.5426	3.1317	3.2154	3.2348	3.1077	2.6261
(Estimated Trace/Percentages of Visited States) (*500)	2.9644	6.0269	14.8466	37.4151	109.5020	326.3875	930.2841
(*500)/(The Actual Trace)	0.9303	1.0595	1.4620	2.0641	3.3843	5.6512	9.0237
Number of Generations: 1000							
(Average) Number of States Visited	13	35	77	140	207	284	351
Percentages of Visited States (%)	81.25	54.69	30.08	13.67	5.05	1.73	0.54
Estimated Trace by the Proposed Metric	2.0982	2.8219	4.0908	4.9186	5.4002	5.4528	5.0476
(Estimated Trace/Percentages of Visited States) (*1000)	2.5824	5.1601	13.6007	35.9760	106.8554	314.5709	942.4512
(*1000)/(The Actual Trace)	0.8104	0.9071	1.3393	1.9847	3.3025	5.4466	9.1417
Number of Generations: 1500							
(Average) Number of States Visited	13	39	90	176	279	397	506
Percentages of Visited States (%)	81.25	60.94	35.16	17.19	6.81	2.42	0.77
Estimated Trace by the Proposed Metric	2.1311	3.1494	4.6002	5.9594	6.8408	7.3889	7.0296
(Estimated Trace/Percentages of Visited States) (*1500)	2.6228	5.1682	13.0850	34.6728	100.4298	304.9358	910.4647
(*1500)/(The Actual Trace)	0.8231	0.9085	1.2886	1.9128	3.1039	5.2798	8.8315
Number of Generations: 2000							
(Average) Number of States Visited	14	42	100	203	334	497	647
Percentages of Visited States (%)	87.50	65.63	39.06	19.82	8.15	3.03	0.99
Estimated Trace by the Proposed Metric	2.1711	3.2489	4.8517	6.3587	7.9047	8.7813	8.7114
(Estimated Trace/Percentages of Visited States) (*2000)	2.4812	4.9507	12.4203	32.0755	96.9392	289.4821	882.3972
(*2000)/(The Actual Trace)	0.7786	0.8703	1.2231	1.7695	2.9960	5.0122	8.5592
Number of Generations: 5000							
(Average) Number of States Visited	15	51	139	306	570	933	1342
Percentages of Visited States (%)	93.75	79.69	54.30	29.88	13.92	5.69	2.05
Estimated Trace by the Proposed Metric	2.3152	3.7109	5.6716	8.3251	11.1602	13.9268	16.0126
(Estimated Trace/Percentages of Visited States) (*5000)	2.4696	4.6569	10.4456	27.8591	80.1966	244.5628	781.9699
(*5000)/(The Actual Trace)	0.7750	0.8186	1.0286	1.5369	2.4786	4.2344	7.5851
The Actual Trace	3.1866	5.6887	10.1548	18.1266	32.3562 (Guess)	57.7557 (Guess)	103.0934 (Guess)

Table 6.2 The Relationships among the Estimated Trace, the Proportion of the Visited States over the Total Number of States ($n' = 12$)

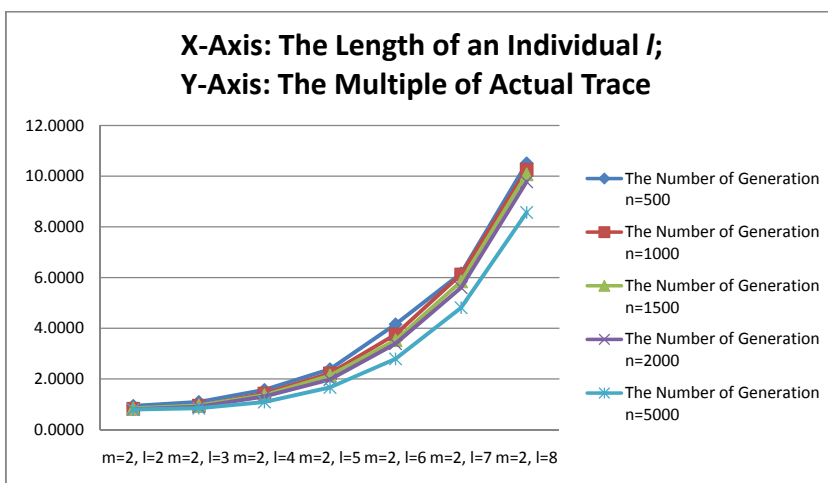


Figure 6.2 The relationship between the value and the setting of l

the setting of l is shown in Figure 6.2, which is based on the results from the setting $n' = 10$. Figure 6.3 shows the difference between the setting $n' = 10$ (the power of *CMS*) and $n' = 12$ when the number of generations is set to 2000. The observations on other fitness functions have similar results, although the values are slightly different.

Based on my investigation, different fitness functions, with fixed numbers of generations, have different multiple functions (as shown in Figure 6.2) for actual traces, but all of them can be fitted by the curve $y = \alpha^{l-3}$ when $l \geq 4$, where $\alpha \geq 1$. Note that different settings on the number of generations have different values for α . For each fitness function, we can always compute its own multiple functions to derive the (estimated) actual traces for estimating the number of generations for the empirical global convergence; however, that idea has some drawbacks. We have to compute everything for each fitness function, which costs a lot of computation time. In addition, computation errors of the multiple functions may sometimes result in the situation that the estimated λ_2 (i.e., the second largest eigenvalue in absolute value), denoted as $\hat{\lambda}_2$, is greater than 1. In order to conquer the above problems and provide a robust method for the estimation, the entire estimation framework is analyzed and simplified. The following explains how the framework works.

Let us discuss the equation planned to be used as the fitting curve in Figure 6.1 first.

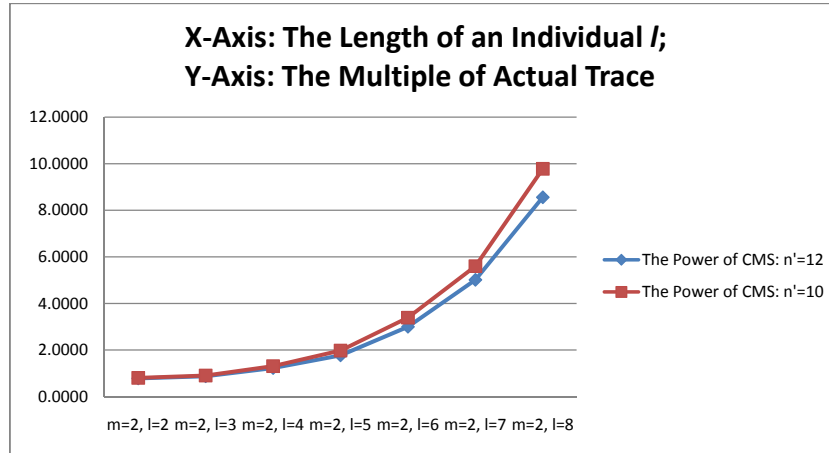


Figure 6.3 The relationship between the value and the setting of l

Theoretically, the curve should be $y = f(x) = 1 + m_2\lambda_2^x + m_3\lambda_3^x + \dots + m_q\lambda_q^x$, where m_2, \dots, m_q and $\lambda_2, \dots, \lambda_q$ are defined in Theorem 4 in Chapter 3; however, to include all of the parameters (m_2, \dots, m_q and $\lambda_2, \dots, \lambda_q$) costs a lot of computation overhead. As it is commonly known that $\lambda_2, \dots, \lambda_q$ are all less than 1, for a large enough x , the terms $m_3\lambda_3^x, \dots, m_q\lambda_q^x$ can be ignored. The equation $y = f(x) = 1 + m_2\lambda_2^x$ is thusly applied to fit the curve. In this case, merely two points are needed for the fitting. Note that in order to obtain more precise results, higher dimensional fitting curves using the existing fitting techniques, such as least squares fitting, can be applied here, but more points are needed for the fitting.

With the two points version, a proper set of fitness functions with different difficulty levels is chosen for estimating λ_2 . With an arbitrary setting of m and l , the estimations of λ_2 derived by three pairs of points, $(n' = 10, n' = 12)$, $(n' = 10, n' = 20)$, and $(n' = 20, n' = 22)$, are compared. Based on the results, it is considered true that $\hat{\lambda}_2$ computed based on the pair $(n' = 20, n' = 22)$ is always the largest. For instance, with respect to the fitness function $y = f_3(x) = 1 + \sin(x)$ with the setting m equal to 2 and l equal to 5, $\hat{\lambda}_2$ derived by the pair $(n' = 10, n' = 12)$ is equal to 0.9554, $\hat{\lambda}_2$ derived by the pair $(n' = 10, n' = 20)$ is equal to 0.9578, and $\hat{\lambda}_2$ derived by the pair $(n' = 20, n' = 22)$ is equal to 0.9612. The reason is that, at the point $x = 10$, $m_3\lambda_3^x + \dots + m_q\lambda_q^x$ is not really close to 0, which causes some noises for the estimation. Although the $\hat{\lambda}_2$ derived by the pair $(n' = 20, n' = 22)$ is the best among

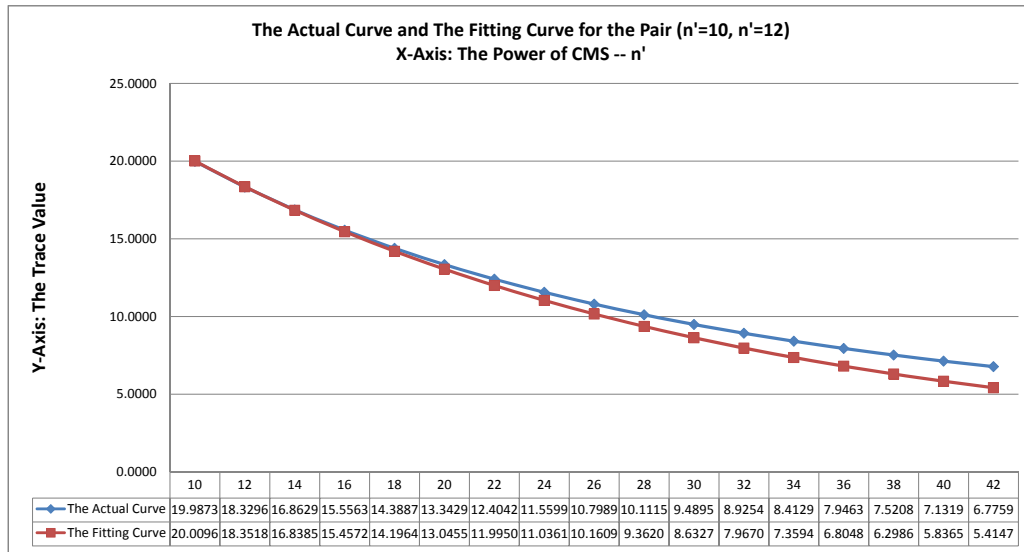


Figure 6.4 The relationship between the actual curve and the curve fitted by the pair ($n' = 10, n' = 12$)

the three, the noises caused by $m_3\lambda_3^x + \dots + m_q\lambda_q^x$ still exist, which makes the estimated λ_2 slightly smaller than the actual λ_2 . The fitting curves based on those three $\hat{\lambda}_2$'s and their corresponding \hat{m}_2 's are drawn in Figure 6.4, Figure 6.5, and Figure 6.6, respectively, where \hat{m}_2 represents the estimated m_2 . Comparing three of them, the curve generated by the pair ($n' = 20, n' = 22$) is the best fit for the actual curve. Note that the n' cannot be too large when it is applied to estimate λ_2 for any fitness function. Too large n' may cause some errors due to precision problems.

This paragraph mainly discusses and analyzes the approach adopted for estimating traces with certain powers n' and the corresponding λ_2 . Consider an arbitrary fitness function. With respect to this fitness function, we can compute two multiple functions for both $n' = 20$ and $n' = 22$. Suppose the value of the multiple function for $n' = 20$ at $l = 16$ is $Mul_{20}(16)$, and the value of the multiple function for $n' = 22$ at $l = 16$ is $Mul_{22}(16)$. The traces for $n' = 20$ and $n' = 22$ are $\frac{\text{Estimated Trace } (n'=20)}{\text{Percentages of Visited States}} / Mul_{20}(16)$ and $\frac{\text{Estimated Trace } (n'=22)}{\text{Percentages of Visited States}} / Mul_{22}(16)$, respectively. Note that the fraction $\frac{\text{Estimated Trace}}{\text{Percentages of Visited States}}$ is used for estimating traces because the proportion of visited states roughly presents the percentages of traces discovered by the

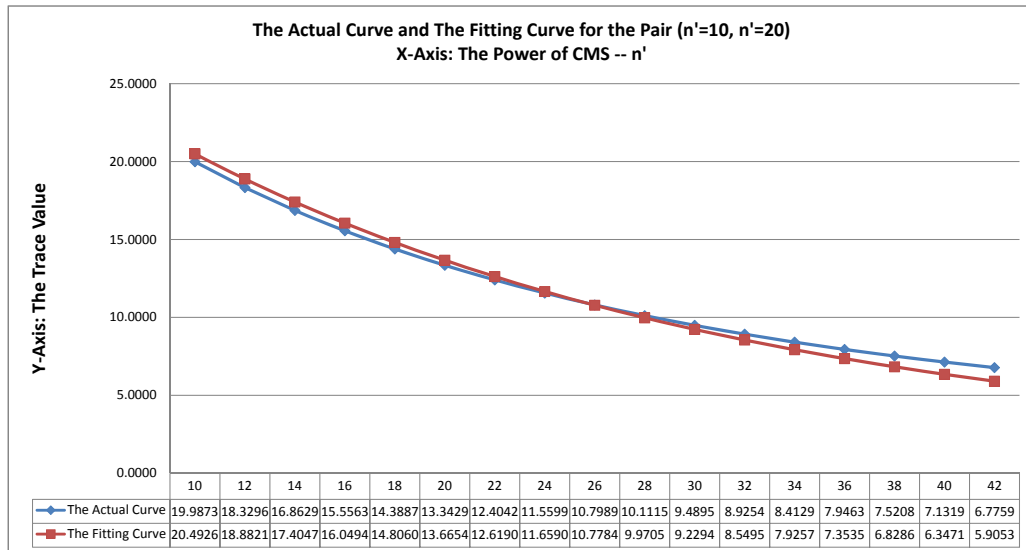


Figure 6.5 The relationship between the actual curve and the curve fitted by the pair ($n' = 10, n' = 20$)

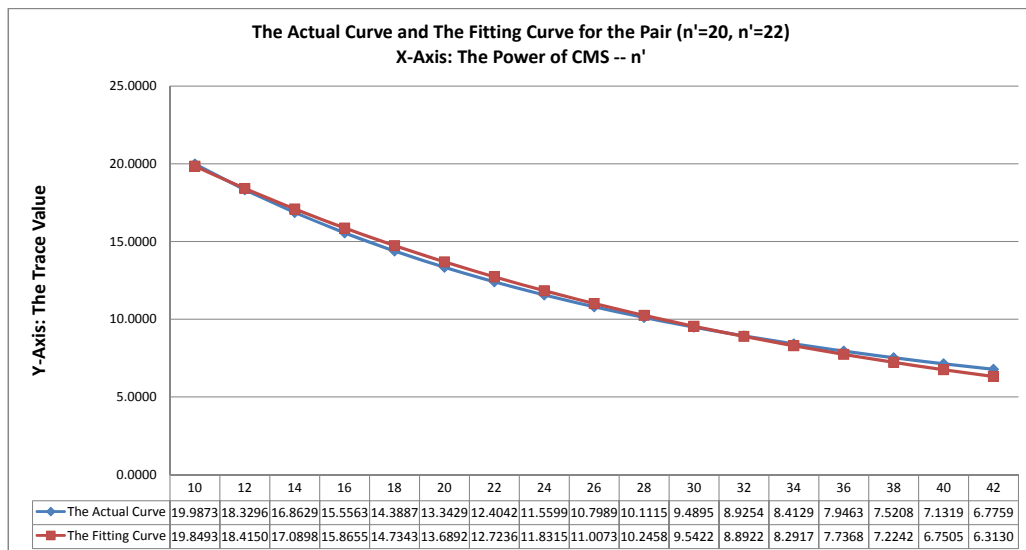


Figure 6.6 The relationship between the actual curve and the curve fitted by the pair ($n' = 20, n' = 22$)

evaluation metric. From Table 6.1 and Table 6.2, we know that the fraction overestimates the traces by some multiples when $l \geq 4$. The multiple functions are leveraged to solve the over-estimated problem. The reason for the overestimation is that usually the sum of the relatively larger diagonal elements is discovered and computed by the evaluation metric. The amount is already included in the estimated trace. Estimating the sum of smaller diagonal elements using the sum of the relatively larger diagonal elements causes the problem. After deriving the estimated traces for $n' = 20$ and $n' = 22$, the equations

$$\begin{cases} 1 + m_2\lambda_2^{20} = \text{Trace}((CMS)^{20}) \\ 1 + m_2\lambda_2^{22} = \text{Trace}((CMS)^{22}) \end{cases}$$

are applied to estimate λ_2 . From the equations, we obtain

$$\begin{aligned} \hat{\lambda}_2^2 &= \frac{\text{Trace}((CMS)^{22}) - 1}{\text{Trace}((CMS)^{20}) - 1} \\ &= \frac{\text{Trace}((CMS)^{22})}{\text{Trace}((CMS)^{20})} + \left(\frac{\text{Trace}((CMS)^{22}) - 1}{\text{Trace}((CMS)^{20}) - 1} - \frac{\text{Trace}((CMS)^{22})}{\text{Trace}((CMS)^{20})} \right) \\ &= \frac{\text{Trace}((CMS)^{22})}{\text{Trace}((CMS)^{20})} - \left(\frac{1}{\text{Trace}((CMS)^{20}) - 1} - \frac{\text{Trace}((CMS)^{22})}{\text{Trace}((CMS)^{20})(\text{Trace}((CMS)^{20}) - 1)} \right). \end{aligned} \quad (6.1)$$

Note that the second term of the last line in Equation (6.1) can be ignored if $\text{Trace}((CMS)^{20})$ is large enough (e.g., $\text{Trace}((CMS)^{20}) \geq 1000$). We use a function $\text{Correction}(\text{Trace}((CMS)^{20}), \text{Trace}((CMS)^{22}))$ to denote it. Combined with the computations mentioned earlier, Equation (6.1) becomes

$$\begin{aligned} \hat{\lambda}_2^2 &= \left(\frac{\text{Estimated Trace } (n' = 22)}{\text{Percentages of Visited States}} / \text{Mul}_{22}(l) \right) / \left(\frac{\text{Estimated Trace } (n' = 20)}{\text{Percentages of Visited States}} / \text{Mul}_{20}(l) \right) \\ &\quad - \text{Correction}(\text{Trace}((CMS)^{20}), \text{Trace}((CMS)^{22})), \end{aligned} \quad (6.2)$$

which can be re-written as

$$\begin{aligned} \hat{\lambda}_2^2 &= \frac{\text{Mul}_{20}(l)}{\text{Mul}_{22}(l)} \left(\left(\frac{\text{Estimated Trace } (n' = 22)}{\text{Percentages of Visited States}} \right) / \left(\frac{\text{Estimated Trace } (n' = 20)}{\text{Percentages of Visited States}} \right) \right) \\ &\quad - \text{Correction}(\text{Trace}((CMS)^{20}), \text{Trace}((CMS)^{22})). \end{aligned} \quad (6.3)$$

If both of $Trace((CMS)^{20})$ and $Trace((CMS)^{22})$ are estimated by the proposed evaluation metric at the same time (i.e., they are computed using the same set of random numbers), Equation (6.3) can be simplified and re-written as

$$\hat{\lambda}_2^2 = \frac{Mul_{20}(l)}{Mul_{22}(l)} \left(\frac{\text{Estimated Trace } (n' = 22)}{\text{Estimated Trace } (n' = 20)} \right) - Correction(Trace((CMS)^{20}), Trace((CMS)^{22})), \quad (6.4)$$

since both of them have the same value for *Percentages of Visited States*.

Now let us carefully consider an approximation equation for Equation (6.4) using the limited known information. As mentioned before, it is time consuming to compute the multiple functions for each fitness function. In order to get more precise results, one may choose to perform that. My goal is to simplify the computation process. Hence, an approximation approach is suggested. Based on my observation, the results show that with respect to the same setting of l , the rates $\frac{Mul_{20}(l)}{Mul_{22}(l)}$ of different fitness functions are close; however, fitness functions which have smaller numbers of states visited by the CGA tend to have slightly larger rates. The fitness function $f_7(x) = \sin^6(5\pi[x^{3/4} - 0.05])$ is chosen as the referenced fitness function. (Note that other fitness functions can also be used as the referenced fitness function.) For different l ($l \geq 4$), the rate $\frac{Mul_{20}(l)}{Mul_{22}(l)}$ of f_7 (when the number of generations n is set to 2000) is approximate to

$$\frac{Mul_{20}(l)}{Mul_{22}(l)} = (1.0072)^{l-3}.$$

Some adjusted coefficients need to be applied for other fitness functions. The one suggested is the fraction $\frac{\text{The Number of States Visited for } f_7}{\text{The Number of States Visited for the Fitness Function}}$ for a specific fitness function. The approximation equation for Equation (6.4) can be

$$\hat{\lambda}_2^2 \approx (1.0072)^{l-3} (\text{The Adjusted Coefficient}) \left(\frac{\text{Estimated Trace } (n' = 22)}{\text{Estimated Trace } (n' = 20)} \right), \quad (6.5)$$

where $l \geq 4$. The $Correction(Trace((CMS)^{20}), Trace((CMS)^{22}))$ is ignored because for most of the cases, we need Equation (6.5) to estimate λ_2 for large-scale problems; in which case, l is greater than 14. When $l \geq 14$ and $n' = 20$, the traces are around or greater than 1000.

6.2 The Number of Generations for the Empirical Global Convergence

After obtaining $\hat{\lambda}_2$, the equation $1 + m_2\lambda_2^{20} = \text{Trace}((CMS)^{20})$ is used to estimate m_2 . Before that, $\text{Trace}((CMS)^{20})$ needs to be computed first. By the computation in the previous section, for each fitness function, $\text{Trace}((CMS)^{20})$ is equal to the fraction $\frac{\text{Estimated Trace } (n'=20)}{\text{Percentages of Visited States}}$ divided by a number $Mul_{20}(l)$ corresponding to the fitness function. According to the results on the values of $Mul_{20}(l)$ for fitness functions in the observed set, the largest $Mul_{20}(l)$ is around ten times of the smallest one. As mentioned before, it is not very efficient to compute the multiple functions for each fitness function. Hence, using one of the multiple functions instead of all of them is suggested. Note that the selection of the value $Mul_{20}(l)$ only impacts \hat{m}_2 , which is employed in computing n from the equation $1 + m_2\lambda_2^n = 1 + \epsilon$ (as in Figure 6.1), in the entire estimation framework. An equivalent equation for $1 + m_2\lambda_2^n = 1 + \epsilon$ is

$$n = \frac{\log(\frac{\epsilon}{m_2})}{\log(\lambda_2)}. \quad (6.6)$$

From Equation (6.6), we can get that λ_2 is more sensitive than m_2 . Moreover, Figure 6.6 shows that the fitting curve computed by the proposed estimation method is eventually lower than the curve drawn by the actual traces after n exceeds a value. That means the number of generations for the global convergence estimated by the proposed method is smaller than the actual one. Hence, smaller $Mul_{20}(l)$ is suggested to be leveraged so that larger \hat{m}_2 can be used to solve n in Equation (6.6).

6.3 The Verification of the Proposed Estimation Framework

The one-dimensional fitness functions f_1 to f_8 in Chapter 5, with the setting $(m, l) = (2, 16)$, are used for the verification. Table 6.3 shows the pair $(\hat{\lambda}_2, \hat{m}_2)$ with respect to each fitness function computed by the proposed methodology. Note that $Mul_{20}(16) = 158.5566 \approx 159$ (from the observed set) is applied to derive \hat{m}_2 .

Figure 6.7 to Figure 6.10 shows the cumulative distribution function (CDF) for fitness functions f_1 to f_8 , respectively. It is estimated by observing the convergence trends of the fitness functions from 1000 trials. More specifically, 1000 trials of the CGA (with best solution

Table 6.3 The Pair $(\hat{\lambda}_2, \hat{m}_2)$ with respect to Each Fitness Function

Fitness Function	$\hat{\lambda}_2$	\hat{m}_2
f_1	0.8490	102814
f_2	0.8434	150613
f_3	0.8890	76333
f_4	0.9339	26663
f_5	0.8646	87450
f_6	0.8699	84670
f_7	0.9439	85533
f_8	0.9165	12535

maintained) are run for each fitness function. The CDFs with the length of the bin being 500 are generated. For the non-periodic fitness functions, the trials, in which the CGA reach the global optimal solutions, are counted. Note that the isolation points of the fitness functions f_5 and f_6 are ignored here. That is, as long as the best solutions maintained reach 90000.0, the trials of the CGA are considered converging to the global optimal solutions. Sometimes, they can reach 100000.0. However, based on my observation, the best solutions have certain difficulty to reach 100000.0 for all trials even for a very large n . For the periodic fitness functions f_3 and f_7 , it is almost impossible for their best solutions to reach only a single point since they have more than one global optimal solutions. Therefore, for f_3 and f_7 , with the best solutions maintained over time, when for the first time x (best solutions) satisfy

$$|f_i(x) - \text{Global Optimal Value}| < 10^{-6},$$

the trials are considered converging to the global optimal solutions. For those figures, the maximum number of generations is set to 70000. In other words, the trials which run more than 70000 generations are not shown in the figures.

In order to compare the CDFs of the CGA with respect to those fitness functions, the bin, representing a range for numbers of generations, which first reaches probability of 80% in the CDF for each fitness function is listed in Table 6.4. (The second column of Table 6.4 repeatedly shows the $\hat{\lambda}_2$ column in Table 6.3.) In Table 6.4, the second column which is derived using my

proposed methodology and the third column show similar results. That is, the difficulty of the CGA with respect to the fitness functions f_2 , f_1 , f_5 , and f_6 is the least. The difficulty of the CGA with respect to f_7 and f_4 is the greatest. This implies that the proposed methodology can derive close values for the corresponding λ_2 .

Table 6.4 The Bins Which First Reach 80% in CDFs

Fitness Function	$\hat{\lambda}_2$	80% in CDFs
f_1	0.8490	15500 – 16000
f_2	0.8434	14500 – 15000
f_3	0.8890	24000 – 24500
f_4	0.9339	> 70000
f_5	0.8646	16000 – 16500
f_6	0.8699	16500 – 17000
f_7	0.9439	57000 – 57500
f_8	0.9165	44000 – 44500

Table 6.5 The Empirical Global Convergence for Each Fitness Function

Fitness Function	Condition (99% of the Global Optima)	$\epsilon = 0.003$		$\epsilon = 0.000003$	
		Derived Number of Generations	Probability (Out of 40)	Derived Number of Generations	Probability (Out of 40)
f_1	Best Value ≥ 594	106	0.8250	149	0.9500
f_2	Best Value ≥ 35640	105	1.0000	145	1.0000
f_3	Best Value ≥ 1.98	145	1.0000	204	1.0000
f_4	Best Value ≥ 0.99	234	1.0000	335	1.0000
f_5	Best Value ≥ 89100	119	1.0000	166	1.0000
f_6	Best Value ≥ 89100	124	0.9000	172	0.9750
f_7	Best Value ≥ 0.99	298	1.0000	417	1.0000
f_8	Best Value ≥ 0.99	175	0.9250	254	0.9750

Table 6.5 shows the results derived by the proposed methodology. With respect to different settings of ϵ , the probabilities of satisfying the condition that the best solution maintained over time is greater than or equal to 0.99 multiple of the global optimal solution are different. The result from small ϵ has higher probability to satisfy the condition. Comparing the numbers of generations derived by the proposed method and the upper bounds of the third column in Table 6.4, we can get that the latter ones are in certain multiples of the former ones. Except

f_4 (i.e., the unknown one), the multiples are in between 138 and 255 for $\epsilon = 0.003$, and in between 99 and 176 for $\epsilon = 0.000003$, respectively. Hence, the number of generations derived by the proposed method is useful for providing some information about the empirical global convergence if a certain multiple greater than 255 (e.g., 260) is applied to it and ϵ is set to be less than 0.003.

Another important result derived from Table 6.3 and Table 6.5 is that, the convergence mainly depends on the value λ_2 , while m_2 , related to the constant in Inequality (4.3), does not have significant impact on it. For instance, the \hat{m}_2 with respect to f_2 is much larger than the \hat{m}_2 with respect to f_1 , but in the *Derived Number of Generations* columns in Table 6.5 for both $\epsilon = 0.003$ and $\epsilon = 0.000003$, the *Derived Number of Generations* for f_2 is a little bit less than that for f_1 . The values of $\hat{\lambda}_2$ mainly control the order. Equation (6.6) explains this result. Further, this result coincides with my earlier result (see Inequality (4.3)) and many work in the literature that the second largest eigenvalue represents the convergence rate of a Markov chain (see Definition 7 for convergence rate).

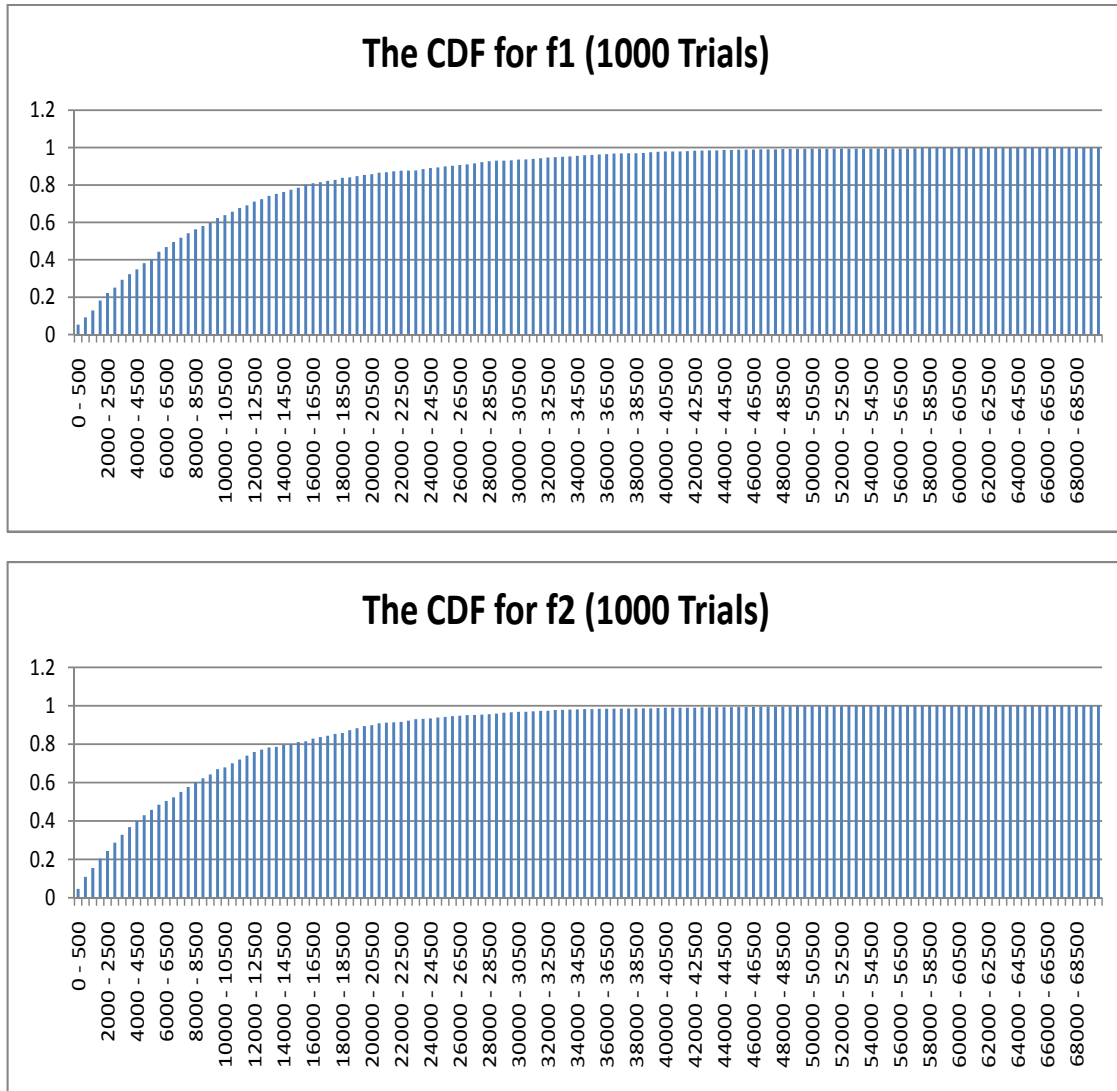


Figure 6.7 The CDF for Number of Trials to Find the Global Optimal Solution with respect to f_1 and f_2 (X-Axis represents the number of generations; Y-axis represents the cumulated number of trials to find the global optimal solution (out of 1000 trials))

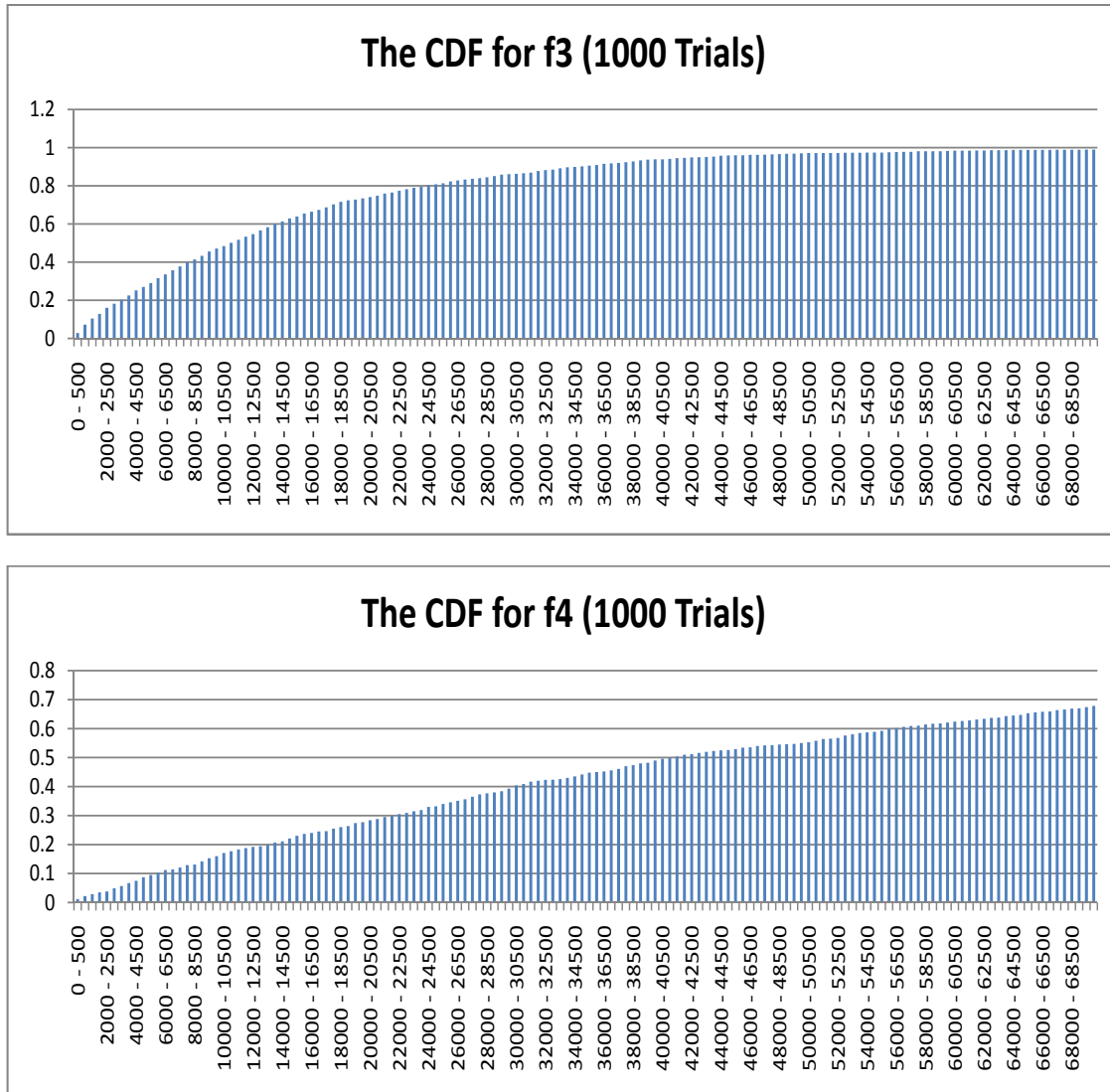


Figure 6.8 The CDF for Number of Trials to Find the Global Optimal Solution with respect to f_3 and f_4 (X-Axis represents the number of generations; Y-Axis represents the cumulated number of trials to find the global optimal solution (out of 1000 trials))

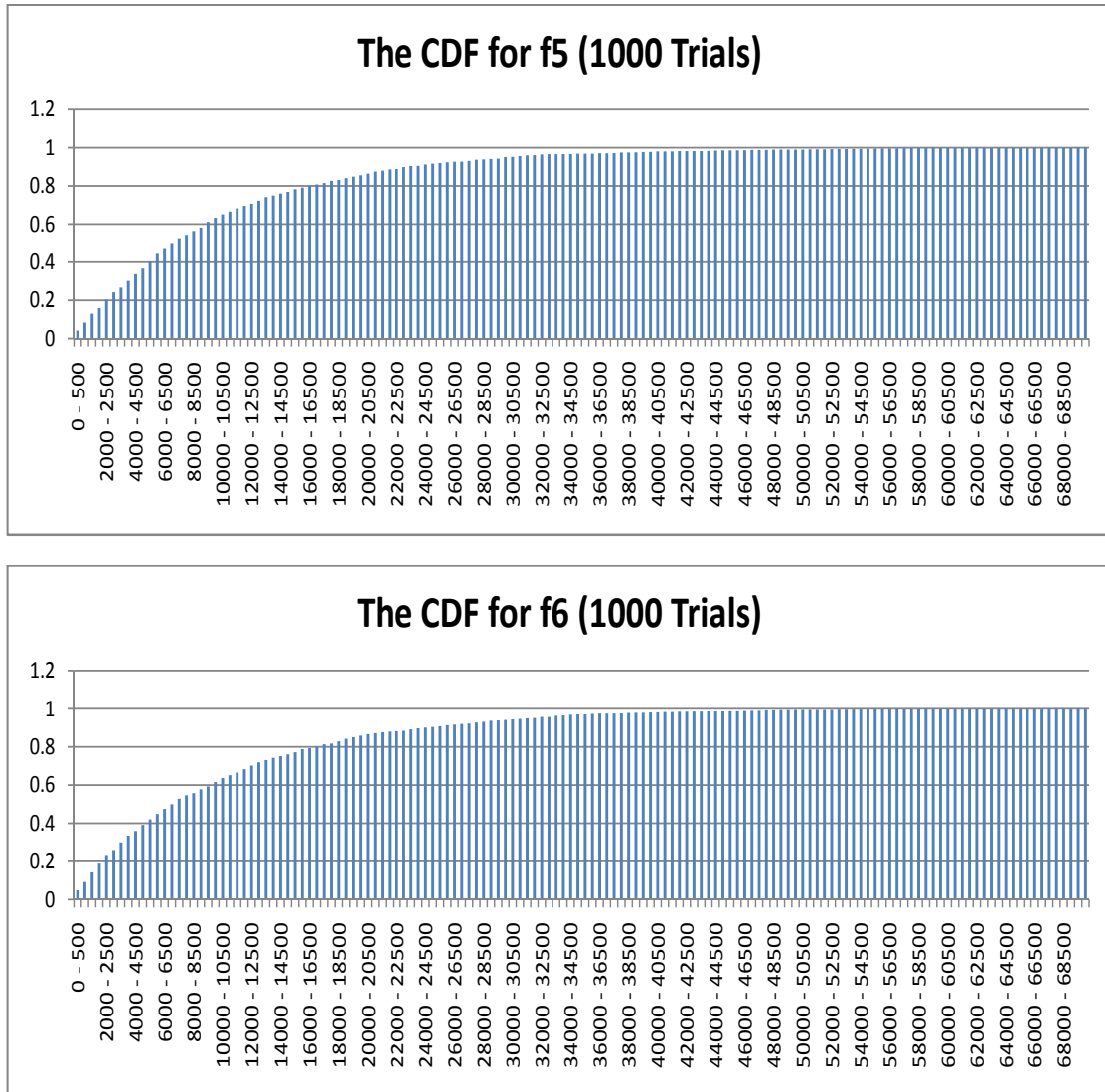


Figure 6.9 The CDF for Number of Trials to Find the Global Optimal Solution with respect to f_5 and f_6 (X-Axis represents the number of generations; Y-Axis represents the cumulated number of trials to find the global optimal solution (out of 1000 trials))

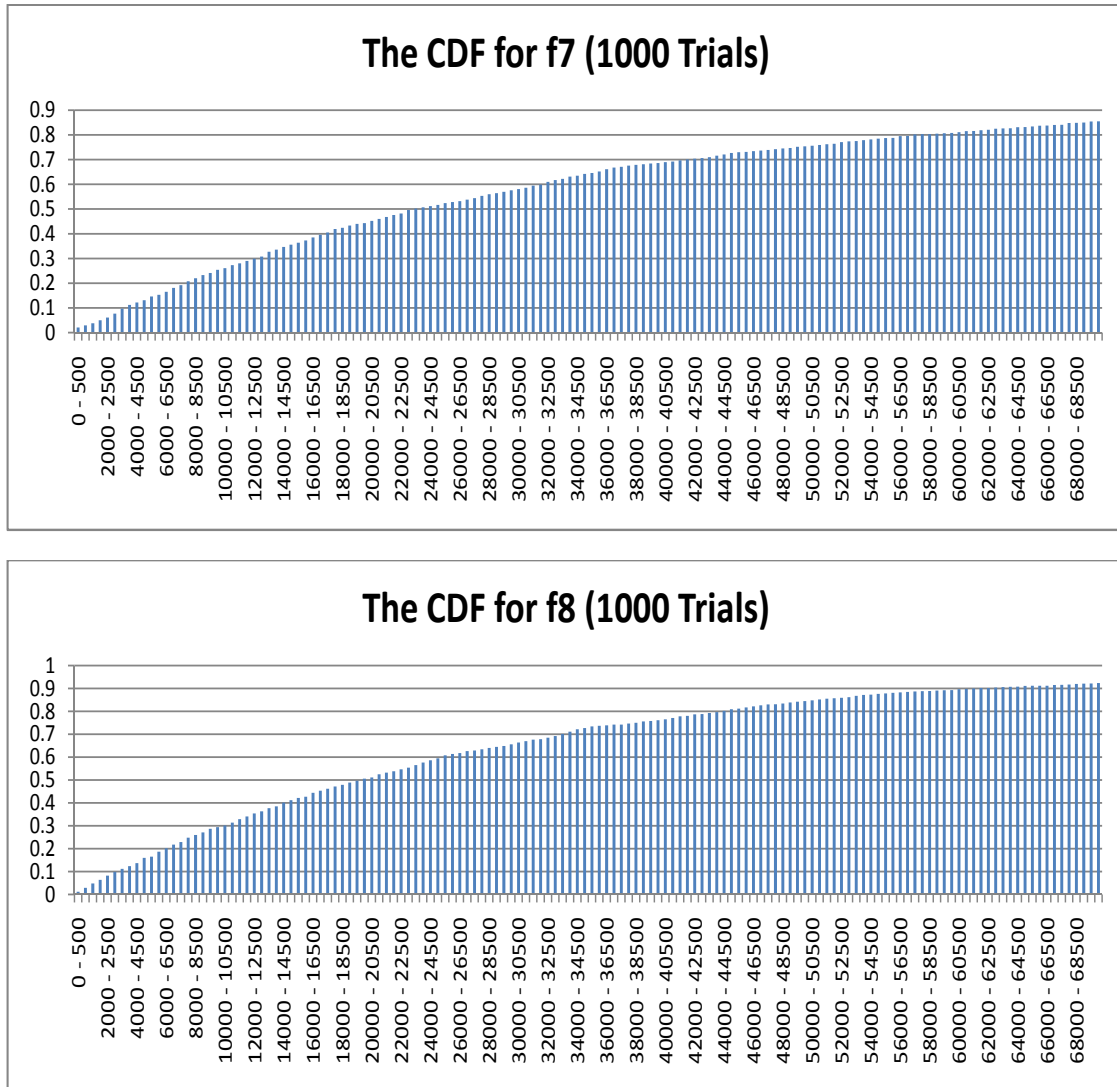


Figure 6.10 The CDF for Number of Trials to Find the Global Optimal Solution with respect to f_7 and f_8 (X-Axis represents the number of generations; Y-Axis represents the cumulated number of trials to find the global optimal solution (out of 1000 trials))

CHAPTER 7. A Case Study - Evolutionary Testing

7.1 Overview of Evolutionary Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and coding [Pressman, R. S. (2005)]. Due to its importance, at least 50% of the total cost (including human effort) involved in a software development project is typically consumed by testing [Beizer, B. (1990); Tracey, N. and Clark, J. and Mander, K. (1998)]. Although a lot of effort has been allocated to the task of testing, complete testing is usually impossible in practice because of the vast amount of possible input situations [Wegener, J. (2005)]. For the same reason, effective methods for automatic test data generation are in great demand. Many techniques focusing on that purpose have been proposed and developed. Evolutionary Testing (ET) is one of them and it gradually forms an important branch of the research area in automatic test data generation.

ET exploits evolutionary algorithms (EAs), such as GA, to generate test data. According to the objectives of testing [Pressman, R. S. (2005)], it is believed that ET has its potential to become a major technique for automatically generating test data since it possesses the following features:

- Like Random Testing (RT), it has the ability to search feasible solutions in the search space. However, unlike RT, it can guide solutions to the desired ones through an evaluation criterion. With guidance, ET is capable of producing effective solutions even for complex and poorly understood search spaces with many dimensions. Without it, ET behaves exactly the same as RT. Hence, ET is more powerful than RT.
- ET is able to recombine the input data based on the higher fitness values. That is, it has

the capability to adjust the input data from the testing history, and create the desired ones.

A better encoding of EAs is one that can explore more possible solutions within a fixed time duration. In other words, a good design for ET can cover lots of combinations of inputs and discover errors with a high probability.

Among all the software testing activities (i.e., test case design, test execution, monitoring, test evaluation, test planning, test organization, and test documentation), test case design is essential. Systematic generation of test cases based on the test design is indispensable to the quality of software. However, for most of the test objectives, it is difficult to automate the generation of test cases due to the reasons listed below [Sthamer, H. and Wegener, J. and Baresel, A. (2002)]:

- The generation of test cases for functional testing is usually impossible because in general no formal specifications can be applied in industrial practice.
- Structural testing is difficult to automate due to the limits of symbolic execution.
- No specialized methods and tools exist for testing the temporal behavior of systems.
- A generation of test cases for testing safety constraints is generally impossible.

Hence, test cases have to be defined manually, which is not efficient. Therefore, the researchers have sought other techniques, e.g., ET, to effectively solve these problems.

According to the literature review, ET has been applied for functional testing (black box testing), structural testing (white box testing), and real-time testing [Last, M. and Eyal, S. and Kandel, A. (2006); Michael, C. C. and McGraw, G. and Schatz, M. A. (2001); Baresel, A. and Pohlheim, H. and Sadeghipour, S. (2003); Baresel, A. and Binkley, D. and Harman, M. and Korel, B. (2004); McMinn, P. (2004); Wegener, J. (2005)] to automatically generate test data. Among all of the existing work, structural testing is the method that has been investigated and developed the most.

In structural testing, various methods to analyze the coverage of program structures have been proposed. Depending on their test aims, fitness functions of ET are formulated. The fitness functions summarized below are commonly seen in the literature.

The test data generation problem for structural testing is to find a set of program inputs that achieves the desired coverage [Tracey, N. and Clark, J. and Mander, K. (1998)]. This type of problems has been converted to optimization problems since the test criteria are specifically known. Most of the work defines the fitness function based on the test criterions. The methods to construct fitness functions can be categorized in Figure 7.1 [McMinn, P. (2004)].

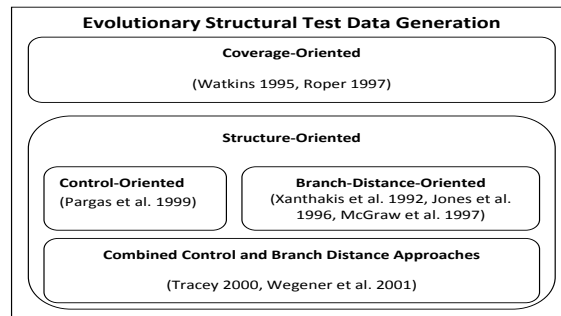


Figure 7.1 Classification of Dynamic Structural Test Data Generation Techniques Using EAs [McMinn, P. (2004)]

From Figure 7.1, we can see that researchers started to apply EAs for software testing around two decades ago. They first set their goal in terms of coverage. For instance, the fitness functions are designed to penalize the individuals that follow already covered paths. Later, they found that it is difficult for the coverage-oriented approach to discover some specific paths. It gradually changed to structure-oriented approach. In this approach, researchers employed different information to form the fitness functions. The most updated one has combined control and branch distance together. The branch distance is usually normalized to the range $[0, 1)$ [McMinn, P. (2004); McMinn, P. and Binkley, D. and Harman, M. (2005)]. With it (fractional part), one may evaluate two feasible solutions with the same fitness value on the approach level (integer part) [Levin, S. and Yehudai, A. (2007)]. The basic form of that to be minimized is

$$Fitness = Approach_Level + Branch_Distance, \quad (7.1)$$

where the *Approach_Level* is the number of target condition statements which are not executed by the given test data, and *Branch_Distance* is the distance from the test data to the input values of the target branch, and it can be computed in various ways. The formula to calculate both depend on the test goal, i.e., which types of coverage are needed [McMinn, P. and Holcombe, M. (2003); McMinn, P. (2004); Wegener, J. (2005)].

7.2 The Flag Problem

Many works have applied EA for structural testing using fitness functions to guide the feasible solutions to form sets of test data for the desired branches. However, problems exist that inhibit the search and have not been perfectly solved yet. The flag problem is one of them. Typically, it occurs when the source code has flag variables. A flag variable is any variable that takes on one of two or more (finite) discrete values [Baresel, A. and Binkley, D. and Harman, M. and Korel, B. (2004)]. For instance, boolean variables are flag variables. In the following, an example is given to show how the flag variable influences the performance of the search.

Suppose Figure 7.2 is part of the tested source code.

1. flag = false;
2. ... /*Nothing related to 'flag'*/
3. if (switch == 5) flag = true;
4. ... /*Nothing related to 'flag'*/
5. if (flag) /* Test aim */

Figure 7.2 An Example: The Source Code

In Figure 7.2, there is a boolean variable called *flag*. The test aim is in line 5, in which *flag* should be equal to 1 (TRUE). In that case, part of the fitness function related to the *flag* variable should be of some formula like $|flag - 1|$, i.e., the absolute value of $(flag - 1)$. Now,

given two or more sets of input test data automatically generated by EA, if all of them do not go to the branch in which the variable *switch* is equal to 5 before line 3, all of the fitness values related to the flag variable are equal to $|0 - 1|$. That is, the fitness values related to the flag variable are the same for all sets of test data with *flag* not being TRUE in line 5. Hence, the fitness function does not provide any guidance for reaching the test aim as the flag can be always FALSE and cannot become TRUE. The search degenerates to random search.

7.3 Discussion

The reason of selecting ET as a case study is that

- There is a difficult flag problem in the context of ET so that the difficult problems can be compared with easier problems (i.e., the problems dealing with testing of flag-free programs);
- From the final fitness value, we can always get information about the quality of solutions. The *Approach_Level* (i.e., integer part) represents which branch the generated test case is located in.

7.4 Experiment Settings

The flag problem is a difficult problem for ET. In order to demonstrate that the proposed evaluation metric has the ability to discern the applicability of CGAs for real-world optimization problems, two types of programs are designed (see Figure 7.3 and Figure 7.4). A corresponding fitness function is formulated for each program (see Table 7.1). CGA with best solution maintained over time is applied to generate test cases for those types of programs with respect to the fitness functions.

The first type of the program (called P_1) has flags which are discrete values. For this type of program, the fitness function formulated using Equation 7.1 has isolation points. The second type of the program (called P_2) is a flag-free program.

Input (a, b, c, d)

```

1.   if (a==18.75)
2.   {
3.       if (b==37.5)
4.       {
5.           if (c==56.25)
6.           {
7.               if (d==75)
8.               {
9.                   Test Aim
10.              }
11.          }
12.      }
13.  }
```

Figure 7.3 Program with Flags

Input (a, b, c, d)

```

1.   if (a>=1)
2.   {
3.       if (b>=2)
4.       {
5.           if (c>=3)
6.           {
7.               if (d>=4)
8.               {
9.                   Test Aim
10.              }
11.          }
12.      }
13.  }
```

Figure 7.4 Program without Flags

f_{14} and f_{15} in Table 7.1 are the corresponding fitness functions for P_1 and P_2 , respectively. The function *Approach_Level* in Table 7.1 is in Figure 7.5.

The CGA with best solution maintained is applied to generate test cases corresponding to P_1 and P_2 with their fitness functions, respectively. With the setting $n = 10000$, $l = 16$, and $m = 2$, the result in Section 7.5 is derived.

In order to further verify that the proposed methodology in Chapter 6 is valid, λ_2 and m_2 are computed for both fitness functions. The result is also presented in Section 7.5.

7.5 The Result of the Case Study

Table 7.2 presents the average values of $Trace(CMS)^{n'}$ derived from the proposed evaluation metric from 40 trials. f_{15} with P_2 has lower estimated $Trace(CMS)^{n'}$ for all n' because it is easier than the other for ET. The results coincide the conclusions made from Chapter 5. From the results, we can also know that the isolation points occasionally can be visited under the condition that they are always encoded as points in the search space of CGAs.

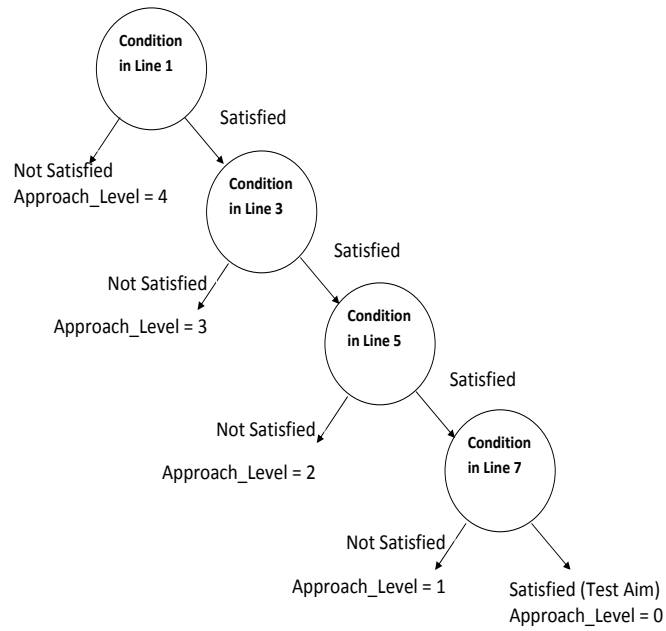
Table 7.3 shows the values of λ_2 , m_2 , and the corresponding information for both fitness functions. λ_2 for the fitness function f_{14} is much greater than λ_2 for the fitness function f_{15} .

Table 7.1 The Fitness Functions of P_1 and P_2

Program	Fitness Function
P_1	$f_{14} = 5 - (Approach_Level + \frac{1}{1101}(a - 18.75 + b - 37.5 + c - 56.25 + d - 75))$
P_2	$f_{15} = 5 - (Approach_Level + \frac{1}{1190}(f_{a'}(a) + f_{b'}(b) + f_{c'}(c) + f_{d'}(d)))$, where $f_{a'}(a) = \begin{cases} a - 1 , & \text{if } a \geq 1 \\ a - 1 + 180, & \text{if } a < 1 \end{cases}$ $f_{b'}(b) = \begin{cases} b - 2 , & \text{if } b \geq 2 \\ b - 2 + 180, & \text{if } b < 2 \end{cases}$ $f_{c'}(c) = \begin{cases} c - 3 , & \text{if } c \geq 3 \\ c - 3 + 180, & \text{if } c < 3 \end{cases}$ $f_{d'}(d) = \begin{cases} d - 4 , & \text{if } d \geq 4 \\ d - 4 + 180, & \text{if } d < 4 \end{cases}$

Therefore, the number of generations derived from the proposed approximation method with respect to f_{14} is much greater than that with respect to f_{15} with $\epsilon = 0.000003$. The results here are also consistent with the conclusions made from previous two chapters. Since the fitness functions f_{14} and f_{15} are multi-dimensional, it may not be appropriate to compare the results derived in Table 6.3, Table 6.5, and their corresponding data.

In order to verify the number of generations derived in Table 7.3, the PDFs and CDFs with respect to both fitness functions are investigated. Figure 7.6 shows the number of trials needed (and the corresponding CDF) to reach the global optimal solution with respect to f_{14} out of 1000 trials. When the number of generations equals to 70000, the CDF in Figure 7.6 merely reaches the probability around 50%. Comparing to the PDF and CDF with respect to f_{15} (not shown here), f_{15} is much easier for the CGA since all of the 1000 trials reach the global optimal solution (i.e., the branch of the test aim) within 500 generations. This result is consistent with the derived numbers of generations in Table 7.3.

Figure 7.5 The Control Flow Graph for P_1 and P_2 Table 7.2 The Results of the Estimated $Trace((CMS)^{n'})$

n'	f_{14} corresponding to P_1	f_{15} corresponding to P_2
10	14.3495	13.9687
12	8.4146	7.9792
14	4.9971	4.7871
16	2.9591	2.7624
18	2.1697	1.8531
20	1.6540	1.5847
22	1.5015	1.3598
24	1.3851	1.2680
26	1.2795	1.1730
Average Best Value	4.2517	4.9296

Table 7.3 The Corresponding Information for Fitness Functions f_{14} and f_{15}

Fitness Function with Information	λ_2	m_2	The Number of Generations Derived from the Proposed Approximation Method $\epsilon = 0.000003$
f_{14} Trace Values for $n = 2000$: $n' = 20$: 0.3180 $n' = 22$: 0.3033 The Number of States Visited: 1329	0.9819	9313	1197
f_{15} Trace Values for $n = 2000$: $n' = 20$: 0.2476 $n' = 22$: 0.2134 The Number of States Visited: 1333	0.9313	5016	299

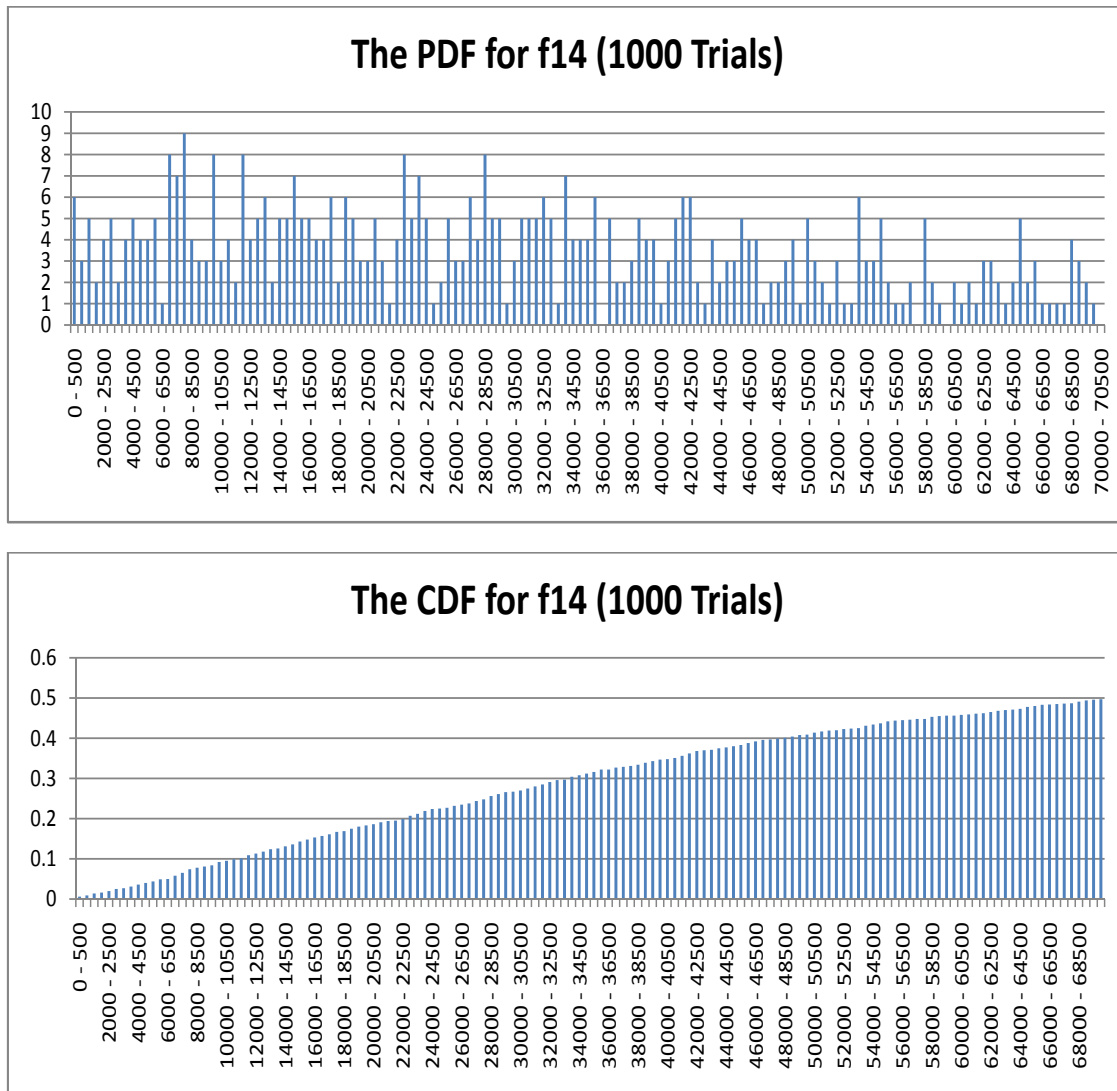


Figure 7.6 The PDF and CDF for Number of Trials to Find the Global Optimal Solution with respect to f_{14} (X-Axis represents the number of generations; Y-Axis represents the cumulated number of trials to find the global optimal solution (out of 1000 trials))

CHAPTER 8. The Generalization of the Proposed Methodology

As aforementioned, the proposed evaluation metric is based on the condition that CMS is regular, which is obtained from the assumption that CGA is applied to solve the optimization problems. In this chapter, I am going to prove that in addition to CGA, the evaluation metric can be applied to any type of selection, and crossover operators, as long as bit mutation is performed after selection and crossover operators.

Theorem 10 *Regardless of the types of selection and crossover operators, the transition matrix SCM is regular if and only if bit mutation is performed after selection and crossover.*

Proof. Since S and C are stochastic matrices, SC is a stochastic matrix (i.e., sum of elements in a row is equal to 1 and each element of SC is nonnegative). Moreover, because bit mutation is performed, for all i and j , the element m_{ij} in M is

$$m_{ij} = \prod_{k=1}^m p_m^{H(\pi_k(i), \pi_k(j))} (1 - p_m)^{l - H(\pi_k(i), \pi_k(j))},$$

where $H(\cdot, \cdot)$ is the Hamming distance of the strings (chromosomes). Hence, m_{ij} is positive for all i and j . That means M is positive.

Since SC is a stochastic matrix and M is positive, SCM is positive. Therefore, SCM is regular. \square

With the condition that SCM is regular, the evaluation metric can be generally applied — not limited to certain types of selection and crossover operators. The corresponding properties investigated and checked in previous chapters should also be further investigated if different selection or crossover operators are used.

Theorem 11 *The transition matrix A constructed by Rudolph's version can be converted to a transition matrix B in Vose's version. Moreover, the statement that A is positive implies that B is positive.*

Proof. Let us consider an arbitrary state, $State_i$, of the transition matrix A constructed by Rudolph's version. Suppose the number of different individuals in $State_i$ is r , each individual is assigned a distinct number between 1 and r , and m_j , $j \in \{1, 2, \dots, r\}$, is the corresponding number of occurrences for *Individual $_j$* (in State i). Note that $m_1 + m_2 + \dots + m_r = m$.

By Rudolph's state representation, there are $\frac{m!}{m_1!m_2!\dots m_r!}$ states which are a permutation of individuals in $State_i$ in the transition matrix A . In Vose's version, those states, including $State_i$, are all in one state since each state represents only the number of occurrences of the individuals. To convert the transition matrix A in Rudolph's version to a transition matrix B in Vose's version, one can simply add up the columns that correspond to the destination states (in Rudolph's version) considered as the same states in Vose's version. Similarly, the rows which correspond to the source states (in Rudolph's version) considered as the same states in Vose's version are added up. For the computation of rows, a multiple of $1/\frac{m!}{m_1!m_2!\dots m_r!}$ needs to be applied to the result. The reason is that given a state with r different individuals in Vose's version, there is $1/\frac{m!}{m_1!m_2!\dots m_r!}$ possibility that this state is permutation k , where $k \in \{1, 2, \dots, \frac{m!}{m_1!m_2!\dots m_r!}\}$, in Rudolph's version (see Figure 8.1).

Based on the computation above, we get that if A is positive, then B is positive since the adding and multiplying manipulations of positive numbers are still positive. \square

In Rudolph's paper [Rudolph, G. (1994)], it is shown that *CMS* constructed by CGA is positive. With Theorem 10, we know that *SCM* constructed by a GA, with bit mutation being performed after any selection and crossover operations, is also positive. The proposed evaluation metric can be generalized to a GA with bit mutation being performed after any selection and crossover operations using Vose's version for state representation, since the converted transition matrix in Vose's version is positive (also regular). More experiments on the performance of the proposed evaluation metric with states being represented by Vose's version need to be conducted. I believe that the estimated trace of a transition matrix in Vose's version

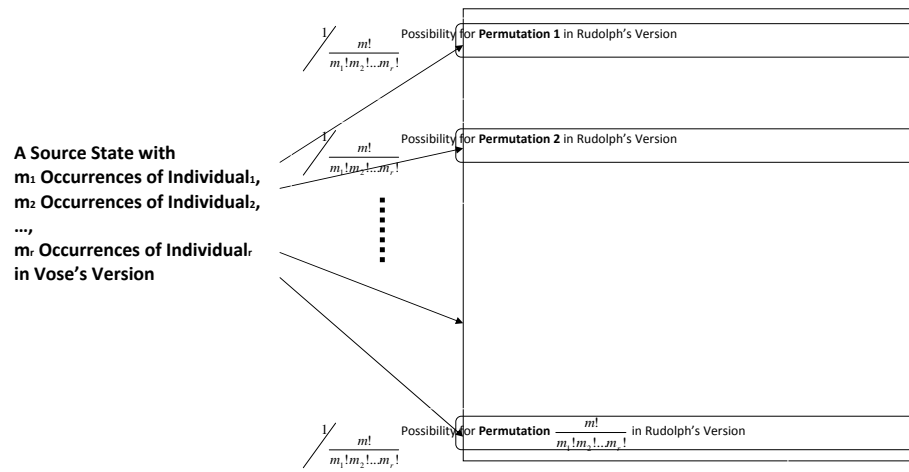


Figure 8.1 An Illustration of the Conversion

will be more accurate than the estimated trace of the transition matrix in Rudolph's version.

This will be one of the directions of my future work.

CHAPTER 9. Conclusion and Future Work

The conclusion of this dissertation includes three sections. The first section describes the contributions of this work; the second section discusses the limitations of the evaluation metric and the proposed approximation approach. In the final section, future work is suggested.

9.1 Contributions of this Work

The contributions of this work include the following:

- The convergence rate of PU (the extended transition matrix) from Perron's Formula is derived (see Chapter 4).
- A research framework is proposed for estimating the applicability of CGAs for real-world optimization problems. I have formulated several corresponding theorems, conducted experiments for the proposed framework, and analyzed the experimental results (see Chapter 5). I have also studied several cases in Software Testing to validate the evaluation metric (see Chapter 6).
- With my evaluation metric, researchers can decide a cluster of better fitness functions among a set of fitness functions by taking a set of fitness functions as input with the same GA configuration. This feature is very important for real-world optimization problems, since researchers or practitioners may consider using more than one fitness function when they attempt to solve optimization problems. The evaluation metric can rank the fitness functions from the best to the worst by sorting the fitness functions by degrees of convergence.

The traditional way (also the only existing computational and analyzable way) to discriminate the fitness functions with respect to a GA configuration is to compute their Markov transition matrices. With a number of 2^{ml} states, at least $2^{ml} \times 2^{ml} \times m$ fitness evaluations need to be performed, which costs a lot of time and is impractical for large-scale real-world applications. Another method, comparison of the fitness values after a fixed number of generations, is not a practical method, since the chosen fixed number may be too large, so that all of the transition matrices empirically converge to their limits.

- An approximation approach is presented for applying the proposed evaluation metric to estimate the number of generations for the global convergence of GAs (Chapter 7).

9.2 Limitations

Both the evaluation metric and the proposed approximation approach have limitations, discussed as follows:

- The proposed evaluation metric has the ability to rank the difficulties of fitness functions with respect to a configuration of GAs. However, it is difficult for the metric to derive a precise value of trace.
- A small n makes the metric arduous to predict the relative convergence behavior of a flat fitness function. This limitation can be solved by using large n . For a large enough n , a close numerical value related to the degree of convergence can be derived. However, the number n may be too large to make the metric practical.
- The proposed approximation approach for predicting the number of generations for the global convergence of GAs has difficulty in dealing with fitness functions that contain isolation points. The empirical results show that with respect to such a fitness function, when n is equal to the number of generations derived from the approximation, the best solution maintained over time reaches a global optimal solution of the fitness function

other than the isolation points in 40 continuous trials. Only a few of the trials obtain the best solutions equal to the isolation points.

Further investigation is needed to tackle the aforementioned limitations.

9.3 Future Work

My future work will be focused on improving the precision of the proposed evaluation metric. Currently, the metric is able to evaluate the applicability of CGA with best solution maintained over time, and is extended to a more general case in Chapter 7 (e.g., the applicability of EAs that perform bit mutation after selection and crossover operators). Further investigation on more classes of EAs is necessary. Chapter 8 also proves that the state representation using Rudolph's version can be converted to the state representation using Vose's version. I believe that this conversion is helpful for improving the precision of the trace estimated by the proposed evaluation metric, since certain states represented using Rudolph's version are combined into a state in Vose's version. The diagonal elements of the transition matrices constructed by Vose's version have larger values than those of the transition matrices constructed by Rudolph's version, and the state space in Vose's version is smaller than that in Rudolph's version. These reasons make the estimation easier in Vose's version. More research work and experiments need to be conducted along this line.

An alternative approach or adjustment to the approximation method is another future direction. The approximation method currently can predict the number of generations for the global convergence of GAs by multiplying the derived values with certain multiples; however, a solution to deal with its limitation is needed. I plan to investigate the Markov transition matrices constructed by the fitness functions with isolation points and solve the problem by either adjusting the proposed approximation or developing a novel approach.

The design of fitness functions of multi-objective problems is the other direction for future research. If the fitness functions cannot be well-formed for multi-objective problems, some of the objectives will not be considered during the GA runs. In the future, I plan to derive suitable rules to guide other researchers in dealing with this problem.

BIBLIOGRAPHY

- Baresel, A. and Pohlheim, H. and Sadeghipour, S. (2003). *Lecture Notes in Computer Science: Structural and Functional Sequence Test of Dynamic and State-Based Software with Evolutionary Algorithms*. Springer-Verlag Berlin Heidelberg.
- Baresel, A. and Binkley, D. and Harman, M. and Korel, B. (2004). Evolutionary testing in the presence of loop-assigned flags: a testability transformation approach. *Proceedings of the International Symposium on Software Testing and Analysis*, 108–118.
- Beizer, B. (1990). *Software Testing Techniques (second edition)*. International Thomson Computer Press.
- Berndt, D. and Fisher, J. and Johnson, L. and Pinglikar, J. and Watkins, A. (2003). Breeding Software Test Cases with Genetic Algorithms. *proceedings of Hawaii International Conference on System Sciences*, 338–347.
- Berndt, D. J. and Watkins, A. (2004). Investigating the Performance of Genetic Algorithm-Based Software Test Case Generation. *proceedings of IEEE International Symposium on High Assurance Systems Engineering*, 261–262.
- Berndt, D. J. and Watkins, A. (2005). High Volume Software Testing using Genetic Algorithms. *proceedings of Hawaii International Conference on System Sciences*, 318–326.
- Bethke, A. D. (1980). Genetic Algorithm as Function Optimizers. *Ph.D. Dissertation, University of Michigan*.

- Briand, L. C. and Labiche, Y. and Shousha, M. (2004). Performance Stress Testing of Real-Time Systems Using Genetic Algorithms. *Technical Report: SCE 03-23, Carleton University*.
- Briand, L. C. and Labiche, Y. and Shousha, M. (2005). Stress testing real-time systems with genetic algorithms. *proceedings of Conference on Genetic and Evolutionary Computation*, 1021-1028.
- Bridges, C. L. and Goldberg, D. E. (1991). *The Nonuniform Walsh-Schema Transform*. San Mateo, CA: Morgan Kaufmann.
- Brown, L. D. and Cai, T. T. and DasGupta, A. (2001). Interval Estimation for a Binomial Proportion. *Statistical Science*, 16(2), 101–117.
- Burden, R. L. and Faires, J. D. (2005). *Numerical Analysis, 8th Edition*. Brooks/Cole.
- Chang, C. K. and Christensen, M. J. (1999). A Net Practice for Software Project Management. *IEEE Software*, 16(6), 80–88.
- Chang, C. K. and Christensen, M. J. and and Zhang, T. (2001). Genetic Algorithms for Project Management. *Annals of Software Engineering 11*, 107–139.
- Chang, C. K. and Cleland-Haung, J. and Hua, S. and Kuntzmann-Combelles, A. (2001). Function-Class Decomposition: A Hybrid Software Engineering Method. *Computer*, 34(12), 87–93.
- Chao, C. (1995). SPMNET: A New Methodology For Software Management. *Ph.D. Dissertation, University of Illinois at Chicago*.
- Ching, W. K. and Ng, M. K. (2006). *Markov Chains: Models, Algorithms and Applications*. Springer Science + Business Media, Inc.

- Clark, J. and Dolado, J. J. and Harman, M. and Hierons, R. M. and Jones, B. and Lumkin, M. and Mitchell, B. and Mancoridis, S. and Rees, K. and Roper, M. and Shepperd, M. (2003). Reformulating Software Engineering as a Search Problem. *IEE Proceedings - software*, 150(3), 161–175.
- Coley, D. A. (1999). *An Introduction to Genetic Algorithms for Scientists and Engineers*. New Jersey: World Scientific Publishing Co. Pte. Ltd.
- Davis, T. E. and Principe, J. C. (1993). A Markov framework for the simple genetic algorithm. *Evolutionary Computation*, 1(3), 269–288.
- DeJong, K. A. and Spears, W. M. and Gordon, D. F. (1995). Using Markov Chains to Analyze GAFOs. *Foundations of Genetic Algorithms 3*, 115–137.
- Ding, L. and Yu, J. (2005). Some Theoretical Results About the Computation Time of Evolutionary Algorithms. *proceedings of Conference on Genetic and Evolutionary Computation*, 1409–1415.
- Droste, S. and Jansen, T. and Wegener, I. (1998). A rigorous complexity analysis of the (1+1) evolutionary algorithm for linear functions with Boolean inputs. *Proceedings of International Conference on Evolutionary Computation (ICEC)*, 499–504.
- Eiben, A. E. and Aarts, E. H. L. and Hee, K. M. V. (1991). Global convergence of genetic algorithms: A markov chain analysis. *Parallel Problem Solving from Nature*, 496, 3–12.
- Fogel, D. B. (1995). *Evolutionary Computation*. New York: IEEE Press.
- Forrest, S. and Mitchell, M. (1993). What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation. *Machine Learning*, 13, 285–319.
- Ge, Y. (2004). Capability-based Software Project Scheduling with System Dynamics and Heuristic Search. *Master's Thesis, Iowa State University*.

- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional.
- Grefenstette, J. J. (1992). *Foundations of Genetic Algorithms 2: Deception Considered Harmful*. San Mateo, CA: Morgan Kaufmann
- Hadley, G. (1964). *Nonlinear and Dynamics Programming*. Addison Wesley
- He, J. and Kang, L. (1999). On the convergence rates of genetic algorithms. *Theoretical Computer Science*, 229, 23–39.
- He, J. and Yao, X. (2001). Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1), 57–85.
- Holland, J. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press.
- Horn, R. A. and Johnson, C. R. (1985). *Matrix Analysis*. Cambridge University Press.
- Horn, J. and Goldberg, D. E. (1995). *Foundations of Genetic Algorithms 3: Genetic Algorithm Difficulty and the Modality of Fitness Landscapes*. San Francisco, CA: Morgan Kaufmann.
- Horn, J. (1995). Genetic Algorithms, Problem Difficulty, and the Modality of Fitness Landscapes. *Master's Thesis, University of Illinois at Urbana-Champaign*.
- Hwang, S. (2004). Cauchy's Interlace Theorem for Eigenvalues of Hermitian Matrices. *American Mathematical Monthly*, 111(2), 157–158.
- Iosifescu, M. (1980). *Finite Markov Processes and Their Applications*. John Wiley and Sons.
- Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007). A Foundational Study on the Applicability of Genetic Algorithm to Software Engineering Problems. *proceedings of Congress on Evolutionary Computation*, 2210–2219.

- Jiang, H. and Chang, C. K. (2008). Deriving Evaluation Metrics for Applicability of Genetic Algorithms to Optimization Problems. *proceedings of Conference on Genetic and Evolutionary Computation*, 1113–1114.
- Kosters, W. A. and Kok, J. N. and Leiden, P. F. (1999). Fourier Analysis of Genetic Algorithms. *Theoretical Computer Science*, 229, 143–175.
- Last, M. and Eyal, S. and Kandel, A. (2006). *Lecture Notes in Computer Science: Effective Black-Box Testing with Genetic Algorithms*. Springer-Verlag Berlin Heidelberg.
- Leemis, L. H. and Park, S. K. (2005). *Discrete-Event Simulation: A First Course*. Prentice Hall.
- Levin, S. and Yehudai, A. (2007). *Lecture Notes in Computer Science: Evolutionary Testing: A Case Study*. Springer Berlin/Heidelberg.
- McMinn, P. and Holcombe, M. (2003). The state problem for evolutionary testing. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03)*, 2488–2497.
- McMinn, P. (2004). Search-Based Software Test Data Generation: A Survey. *Software Testing, Verification and Reliability*, 14(2), 105–156.
- McMinn, P. and Binkley, D. and Harman, M. (2005). Testability Transformation for Efficient Automated Test Data Search in the Presence of Nesting. *Proceedings of the Third UK Software Testing Workshop (UKTest 2005)*, 165–182.
- Michael, C. C. and McGraw, G. and Schatz, M. A. (2001). Generating Software Test Data by Evolution. *IEEE Transactions on Software Engineering*, 27(12), 1085–1110.
- Mitchell, B. S. and Mancoridis, S. (2002). Using Heuristic Search Techniques To Extract Design Abstractions From Source Code. *proceedings of Conference on Genetic and Evolutionary Computation*, 1375–1382.
- Mitchell, M. (1996). *An Introduction to Genetic Algorithm*. Cambridge, MA: MIT Press.

- Naudts, B. and Kallel, L. (2000). A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 4, 1–15.
- Nix, A. E. and Vose, M. D. (1992). Modeling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, 5(1), 79–88.
- Petrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. *Proceedings of IEEE International Conference on Evolutionary Computation*, 798–803.
- Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach (6th edition)*. The McGraw-Hill Companies, Inc.
- Rudolph, G. (1994). Convergence Analysis of Canonical Genetic Algorithms. *IEEE Transactions on Neural Networks*, 5(1), 96–101.
- Rudolph, G. (1996). Convergence of Evolutionary Algorithms in General Search Spaces. *proceedings of Congress on Evolutionary Computation*, 50–54.
- Sareni, B. and Krahenbuhl, L. (1998). Fitness Sharing and Niching Methods Revisited. *IEEE Transactions on Evolutionary Computation*, 2(3), 97–106.
- Schoenauer, M. et al. (2007). Bridging the Gap between Theory and Practice. *MIT Press Journals - Evolutionary Computation*, 15(4), iii–v.
- Setnes, M. and Roubos, H. (2000). GA-Fuzzy Modeling and Classification: Complexity and Performance. *IEEE Transactions on Fuzzy Systems*, 8(5), 509–522.
- Sthamer, H. and Wegener, J. and Baresel, A. (2002). Using Evolutionary Testing to improve Efficiency and Quality in Software Testing. *Proceedings of the 2nd Asia-Pacific Conference on Software Testing Analysis and Review (AsiaSTAR2002)*, 50–54.
- Suzuki, J. (1995). A Markov chain analysis on simple genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 25(4), 655–659.

- Suzuki, J. (1995). A Further Result on the Markov Chain Model of Genetic Algorithms and Its Application to a Simulated Annealing-Like Strategy. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 28(1), 95–102.
- Tang, K. and Yao, X. and Suganthan, P. N. and MacNish, C. and Chen, Y. P. and Chen, C. M. and Yang, Z. (2007). Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization. *Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China*.
- Tracey, N. and Clark, J. and Mander, K. (1998). Automated Program Flaw Finding Using Simulated Annealing. *Proceedings of the International Symposium on Software Testing and Analysis*, 73–81.
- Vieira, F. E. and Menezes, R. and Braga, M. (2006). Using genetic algorithms to generate test plans for functionality testing. *proceedings of Annual Southeast Regional Conference*, 140–145.
- Vose, M. D. and Liepins, G. E. (1991). Punctuated Equilibria in Genetic Search. *Complex Systems*, 5(1), 31–44.
- Wegener, J. (2005). *Lecture Notes in Computer Science: Evolutionary Testing Techniques*. Springer-Verlag Berlin Heidelberg.
- Xu, R. and Qian, L. and Jing, X. (2003). CMM-based software risk control optimization. *IEEE International Conference on Information Reuse and Integration (IRI)*, 499–503.