2014

Grouping and Sequencing of Machining Operations for High Volume Transfer Lines

Soumitra Subhash Bhale
University of Windsor

Grouping and Sequencing of Machining Operations for High Volume Transfer Lines

by

Soumitra Subhash Bhale

A Thesis

Submitted to the Faculty of Graduate Studies

through the department of Industrial & Manufacturing Systems Engineering

in Partial Fulfillment of the Requirements for

the Degree of Master of Applied Science

at the University of Windsor

Windsor, Ontario, Canada

2014

Grouping and Sequencing of Machining Operations for High Volume Transfer Lines

by

Soumitra Bhale

APPROVED BY:

B. Chaouch

Odette School of Business

W. ElMaraghy

Industrial and Manufacturing Systems Engineering

F. Baki, Co-Advisor

Odette School of Business

A. Azab, Co-Advisor

Industrial and Manufacturing Systems Engineering

24 April 2014

DECLARATION OF PREVIOUS PUBLICATION

This thesis includes one original paper that has been previously submitted for publication in a peer reviewed conference, as follows:

Publication title and full citation	Publication Status
Bhale S, Baki MF, Azab A. Grouping and Sequencing of Machining Operations for High Volume Transfer Lines. Proceedings of the 47th CIRP Conference on Manufacturing Systems; 2014 Apr 28-30; Windsor, Canada. Windsor: Canada; 2014. P. 413-18.	in press

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

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

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

ABSTRACT

Transfer lines are employed for mass production of a fixed product or a very narrow range of product variants. This thesis considers a simple transfer line balancing problem with a focus on process planning and line configuration. Design features of the product are grouped and machining operations are sequenced in an optimal manner. The objective is to minimize the handling time fraction of the cycle time consisting mainly of orientation change time and tool change time. A new MILP model is developed to solve the problem with the aforementioned objectives while respecting a set of constraints, which include cutting tool allocation, tool magazine limit, tool life limit, takt time limit and precedence, inclusion & exclusion constraints. A problem-specific simulated annealing algorithm to solve large problems is also proposed. Numerical experiments are presented to illustrate the functionality of the MILP model and the meta-heuristic with respect to optimality and computation time.

DEDICATION

॥श्री वक्रतुण्ड महाकाय सूर्य कोटी समप्रभा निर्विघ्नं कुरु मे देव सर्व-कार्येशु सर्वदा॥

Shree Vakratunda Mahakaya Suryakoti Samaprabha
Nirvighnam Kuru Me Deva Sarva-Kaaryeshu Sarvada

The Lord with the curved trunk and a mighty body, who has the magnificence of a Million suns, I pray to you Oh Lord, to remove the obstacles from all the actions I intend to perform.

The above ‘shloka’ is recited whenever a new endeavor is undertaken. It is an evocation of Lord Ganesha to seek his blessings for actions to be performed.

Dedicated to my

Teachers, family and friends

ACKNOWLEDGEMENTS

I wish to acknowledge the support of my supervisors Dr. Baki and Dr. Azab. The prompt guidance from Dr. Baki was a guiding light in the completion of this work. The long meetings with Dr. Azab were not only helpful to the research but also intellectually stimulating.

I would also like to thank committee members Dr. W. ElMaraghy and Dr. B. Chaouch for their guidance that helped me explore other relevant areas of my research topic and present them in this final version.

The support of my guru Mr. Mayure cannot be forgiven. Without his blessings, it would have not been possible. My parents have been a source of blessings, love, care and support at every step. I want to thank them along with other members of my family including my sister.

Thanks are also due to the Department of Industrial and Manufacturing Systems Engineering for the opportunity to work that helped me ‘earn and learn’. I would also acknowledge the research and recreational facilities at the University of Windsor for helping students balance work and play.

Last but not the least: I thank my friends in Windsor (Navneet, Shivakumar and everyone from Marathi katta) and elsewhere for wonderful support and camaraderie that kept me pushing through.

TABLE OF CONTENTS

DECLARATION OF PREVIOUS PUBLICATION iii
ABSTRACT iv
DEDICATION v
ACKNOWLEDGEMENTS vi
LIST OF TABLES ix
LIST OF FIGURES xi
LIST OF ABBREVIATIONS xii
CHAPTER 1: INTRODUCTION 1
1.1 Transfer lines 1
1.2 Line balancing 3
1.3 Context of problem 4
1.4 Motivation of thesis 7
1.5 Organization of thesis 7
CHAPTER 2: LITERATURE REVIEW 8
2.1 Problem definition 8
2.2 Relevant papers in literature 11
2.3 Gap analysis and novelty of solution 16
CHAPTER 3: MODEL DEVELOPMENT 19
3.1 Assumptions for MILP model 19
3.2 Indices for MILP model 19
3.3 Decomposition algorithm for MILP model 20
3.4 Parameters for grouping sub-model 23
3.5 Decision variables for grouping sub-model 23
3.6 Grouping sub-model objective function and Linearization 24
3.7 Grouping sub-model constraints 25
3.8 Parameters for sequencing sub-model 26
3.9 Decision variables for sequencing sub-model 28
3.10 Sequencing sub-model objective function and Linearization 29

3.11 Sequencing sub-model constraints 30
CHAPTER 4: SIMULATED ANNEALING 32
4.1 Introduction 32
4.2 Generation of initial solution for grouping problem 36
4.3 Neighbor generation scheme for grouping problem 36
4.4 Simulated annealing algorithm for grouping problem 40
4.5 Generation of initial solution for sequencing problem 43
4.6 Neighbor generation scheme for sequencing problem 43
4.7 Simulated annealing algorithm for sequencing problem 45
CHAPTER 5: NUMERICAL EXPERIMENTS 48
5.1 Case study 1 48
5.2 Case study 2 56
5.3 Case study 3 62
CHAPTER 6: CONCLUSIONS AND FUTURE WORK 64
BIBLIOGRAPHY 65
APPENDIX A: AMPL PROGRAM FOR MATHEMATICAL MODEL: CASE STUDY 1 68
APPENDIX B: C++ PROGRAM FOR SA ALGORITHM: CASE STUDY 1 109
APPENDIX C: GAP ANALYSIS FOR LITERATURE 141
APPENDIX D: TLCT MATRIX FOR CASE STUDY 1 143
APPENDIX E: DETAIL RESULTS FOR CASE STUDY 2 148
VITA AUCTORIS 158

LIST OF TABLES

Table 1: Comparison of high speed and standard machining centers[8] 6
Table 2: Orientation change time matrix for the chosen cylinder head benchmark problem (12) 48
Table 3: Processing information for the chosen cylinder head benchmark problem (12) 49
Table 4: Numerical results showing the grouping and sequence obtained (MILP model) 51
Table 5: Size complexity metrics for Das *et al.*'s planning and sequencing model (12) 53
Table 6: Size complexity metrics for proposed grouping and sequencing models 53
Table 7: Numerical results showing the grouping and sequence obtained (SA algorithm) 53
Table 8: Configuration of the tested benchmark problems (14) 56
Table 9: Parameter settings in the tested benchmark problems (14) 57
Table 10: Original computational results from Osman and Baki (14) 57
Table 11: Computational results for the tested benchmark problems 58
Table 12: Numerical results showing effect of cutting speed on cycle time and machine tool requirement (SA algorithm) 62
Table 13: Gap analysis for relevant literature 141
Table 14: Tool change time matrix for Case Study 1 (Operations 1-19) (sec) 143
Table 15: Tool change time matrix for Case Study 1 (Operations 20-38) (sec) 145
Table 16: Computational results for problem 1 148
Table 17: Computational results for problem 2 148
Table 18: Computational results for problem 3 149
Table 19: Computational results for problem 4 150
Table 20: Computational results for problem 5 150
Table 21: Computational results for problem 6 151
Table 22: Computational results for problem 7 152
Table 23: Computational results for problem 8 152
Table 24: Computational results for problem 9 153
Table 25: Computational results for problem 10 154
Table 26: Computational results for problem 11 154
Table 27: Computational results for problem 12 155

Table 28: Computational results for problem 13 156
Table 29: Computational results for problem 14 156
Table 30: Computational results for problem 15 157

LIST OF FIGURES

Figure 1: Flowchart depicting activities of process planning (5) 3

Figure 2: Top and lower level IDEF diagrams 10

Figure 3: Proposed approach for hierarchical grouping and sequencing of machining operations for transfer lines (MILP model) 21

Figure 4: Decomposition algorithm for MILP model 22

Figure 5: Flowchart of a general simulated annealing algorithm 35

Figure 6: Allocation of Design features to workstations before trade 37

Figure 7: Allocation of Design features to workstations after trade 38

Figure 8: Allocation of Design features to workstations before transfer 39

Figure 9: Allocation of Design features to workstations after transfer 39

Figure 10: Proposed approach for hierarchical grouping and sequencing of machining operations for transfer lines (SA algorithm) 40

Figure 11: Flowchart of a simulated annealing algorithm for the grouping problem 42

Figure 12: Flowchart of a simulated annealing algorithm for the sequencing problem 47

Figure 13: Cycle times against takt time for transfer line workstations (MILP model) 52

Figure 14: Vector diagram showing operation sequence at each workstation (MILP model) [DFU, Operation] 52

Figure 15: Cycle times against takt time for transfer line workstations (SA algorithm) 54

Figure 16: Vector diagram showing operation sequence at each workstation (SA algorithm) [DFU, Operation] 55

Figure 17: Comparison of mean solution time required for MILP model and SA algorithm for problems of small size 59

Figure 18: Comparison of mean solution time required for MILP model and SA algorithm for problems of medium size 60

Figure 19: Comparison of mean solution time required for MILP model and SA algorithm for problems of large size 60

LIST OF ABBREVIATIONS

AMPL	A mathematical programming language
BIP	Binary integer program
DFU	Design feature unit
IP	Integer program
MIP	Mixed integer program
MILP	Mixed integer linear program
NPT	Non-productive time
ORCT	Orientation change time
RT	Re-fixturing and reloading time
SA	Simulated annealing
TLCT	Tool change time
TO	Machining time

CHAPTER 1: INTRODUCTION

1.1 Transfer lines

Manufacturing systems have evolved rapidly since their inception and this transformation is expected to continue in a pursuit of optimum utility. This thesis considers a special type of manufacturing system: transfer lines. Transfer lines employ a fixed sequence of machine tools connected by an automated material handling system for mass production of a small family of complex parts (several million parts per year). They are a special case of flow lines. A flow line may be synchronous or asynchronous. In synchronous lines, all parts move thorough the line at the same speed. In asynchronous lines, some parts have to wait before processing at the next station resulting in a buffer. The workstations are also not governed by cycle time (takt time) limit. However, synchronous lines respect takt time limit at each workstation. The workstations are connected by an automated material handling system, which causes the line to function as a single unit. The layout may be either straight or circular.

Transfer lines have several benefits. They require less manpower and space. They ensure low work in progress and lower lead time. As a result, they are widely used in the automotive industry (1). A huge investment is involved in setting up the transfer line for a particular product. This cost in turn affects the cost of the finished product coming out of the line. Therefore, profitability depends on the investment cost and production efficiency. Hence, it is required that line design be done in an optimal manner because cost and efficiency can be optimized at this stage by solving the line balancing problem (2). Line design encompasses analysis of the product, process planning, line configuration, dynamic flow analysis and transport system design and detailed design and line implementation. This thesis will consider line configuration and some activities of process planning. Line configuration and process planning are inter-dependent aspects of line design.

The process planning problem involves preparation of plan for performing machining operations in an optimal manner. Several technological constraints like inclusion and exclusion constraints are respected while solving this problem (1). Operation sequencing and cutting tool allocation problems are part of process planning. The operation sequence generated ought to be feasible with respect to constraints and also serve as the optimal solution with respect to the objective (3). Figure 1 presents a flowchart depicting activities involved in process planning.

Line configuration determines the allocation of machining operations and required equipment to workstations to maximize utility. Takt time limit and precedence constraints are respected in this problem (1). It aims to distribute the operations to the workstations for the total processing time at each workstation to be less than or equal to the takt time. The optimum number of machine tools required by each workstation is determined. Precedence constraints ensure that the operations are performed in the required order (4). A classification of Line balancing problems is presented in 1.2.

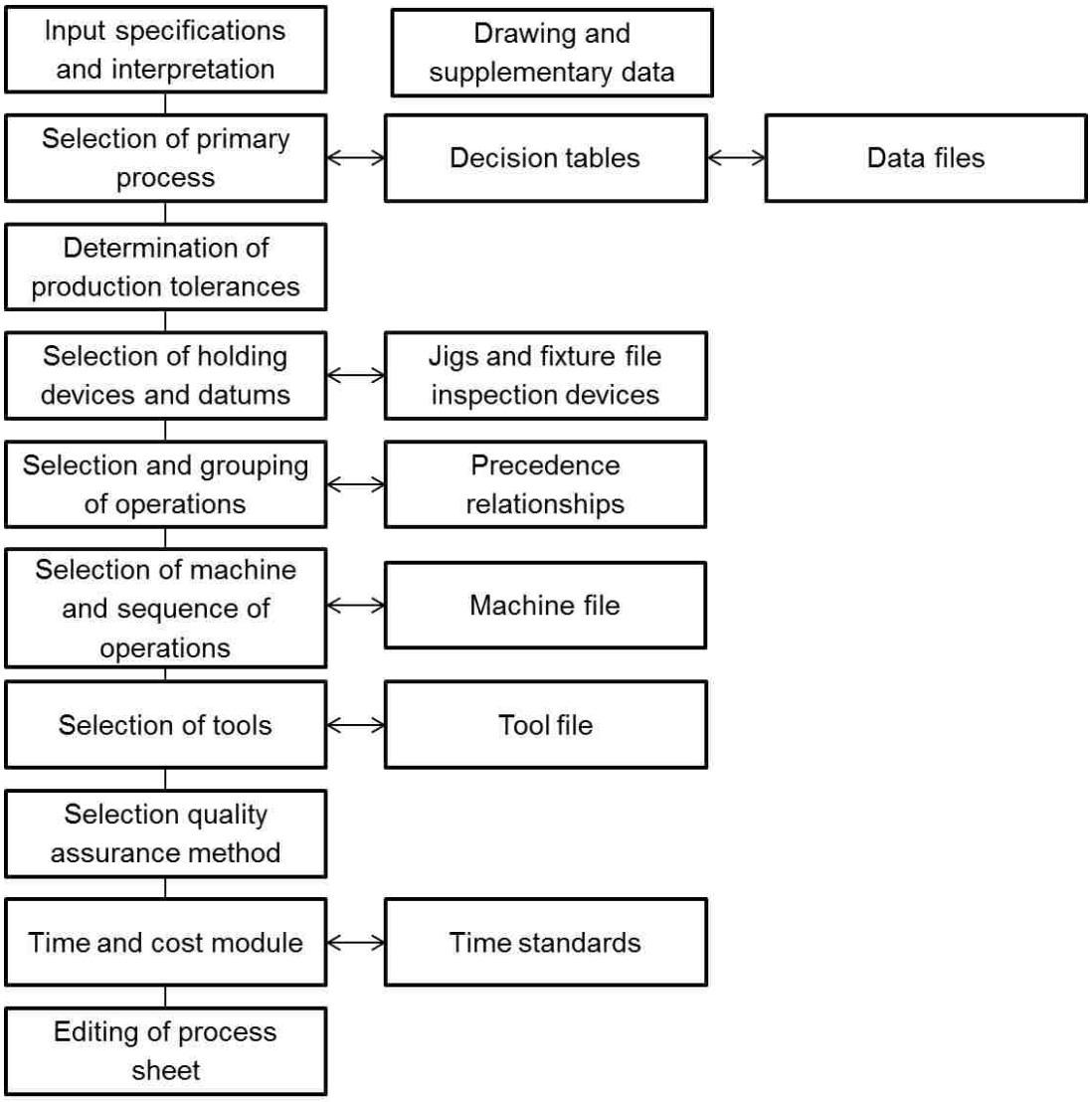


Figure 1: Flowchart depicting activities of process planning (5)

1.2 Line balancing

Both manual assembly line and transfer line classify as flow line production systems. A line balancing problem performs assignment of operations to workstations in order to optimize a criterion while respecting precedence and takt time constraints. When the number of workstations is minimized while respecting a given cycle time, the problem is called time-oriented simple assembly line

balancing problem (SALBP). If skills of workers are differentiated and the total cost of the product is optimized, the problem is called cost-oriented assembly line balancing problem (COALBP) (2). Depending upon the variety of products manufactured, the line may be classified as single-model, mixed-model (different units in arbitrary sequence) and multi-model (sequential batches of different units) (6). When mixed-model production, equipment selection, cost objectives and parallel stations are considered in ALBP, the problem is called generalized assembly line balancing problem (GALBP) (7). The assembly line design problem (ALDP) relates equipment selection to the operations assigned to a station and their execution. When ALBP is extended to apply to transfer lines, the problem is called transfer line balancing problem (TLBP). In TLBP, operations are grouped together to form blocks and these blocks are allocated to workstations. The blocks on different workstations are executed simultaneously and the longest execution time among the blocks determines the pace of the line (8). The problem considered here is analogous to the simple assembly line balancing problem and hence it is called a simple transfer line balancing problem.

1.3 Context of problem

The automotive industry typically requires manufacturing of complex parts in large quantities. Various engine components are produced in large numbers without significant change in the design. The following five parts require critical and high-technology operations: Cylinder block, cylinder head, crank shaft, cam shaft and connecting rod (8). A chronological sequence of the shift in the manufacturing systems adopted for production of these parts is as follows:

i) “General purpose machines such as milling machines, radial drilling machines, etc. with special fixtures with crude material transportation between the machines.

ii) Special purpose machines with multi-spindle, multi-slide, linear or rotary indexing with built-in special fixtures and jigs and roller conveyor in between for part transfer.

iii) Transfer machines with a large number of machining heads built around the different workstations where the component is transferred after completion of operation on one station to the next station through varied types of transfer mechanisms.

iv) Flexible manufacturing systems with conventional CNC machining centers with pallets, large size ATC and transportation system with a centralized control. Fixtures are mostly manual and modular.

v) Flexible transfer machine with CNC machining centres/modules, CNC head changers and automatic transfer of component as in conventional transfer line.

vi) Agile manufacturing with special NC 3-axis machining units - different for heavy duty operations such as milling and for light duty operations such as drilling, tapping, etc. doing parallel processing and has better re-configurability to meet increasing or decreasing volume of production, if so required." (8)

Traditional transfer machines utilize dedicated machining stations for milling, drilling and other operations. The operations are allocated to workstations according to their specialty and hence a sequence of machines is set-up. This type of system permits very small change in the design and time and cost required for a change in set-up are high. They are suitable for high-volume production without significant changes in design.

Current trend in the market is to launch new products or make frequent changes to the existing products. Therefore, a degree of flexibility is required from the manufacturing system to accommodate these changes in design. Flexible transfer lines provide a certain degree of flexibility by following a modular design (7). They employ CNC machining centers and/or multi-spindle head changers for high volume production combined with flexibility. "For a typical 4-

cylinder aluminum cylinder head, numbers of holes vary between 50 & 100, and almost 50-60 processes are used. In a conventional flexible transfer line, the numbers of machines required for a cycle time of 3-4 minutes are: 13-15 NC single spindle machining centers, 5-6 Special Purpose machines and 6-7 assembly machines" (8). However, single spindle machining centers are not as efficient as high speed machining centers. High speed machining centers provide lower operation time due to the use of high speed spindles, high speed traverse and lower tool change time but at a higher cost than single spindle machining centers. A comparison of specifications for standard and high speed machining centers is presented in the Table 1.

Table 1: Comparison of high speed and standard machining centers[8]

No.	Feature	High speed	Standard
1	Spindle speed (rpm)	12000-24000	6000
2	Chip to chip time (sec)	3.0-4.5	10
3	Rapid traverse rate (m/min)	40-60	15-20
4	Acceleration and deceleration	Less than 3	4-6
5	Cycle time (sec)	35-70%	100%

The planning time for transfer line is high. The transfer line balancing problem involves process planning and line configuration. If a long time is spent in planning after commissioning of transfer line, time required to reach full capacity of production increases. This augments the cost of the product coming out of the line. Due to such a limitation, agile manufacturing system is preferred. Agile system provides adaptability and flexibility with CNC cells. As the cells can be built in less time and added as per the requirement, production can be gradually increased. However, the floor space and total cost needed for such a system is high. Only a small batch can be manufactured and the lead time is also high compared to a transfer line. Moreover, a higher level of training is required to be provided to the operators in case of an agile system (8).

1.4 Motivation of thesis

There is a need to develop a solution for simple transfer line balancing problem to reduce the time required for solution. If mathematical techniques are employed to solve the problem, the time spent between commissioning of lines and reaching full capacity of production can be reduced. The automation of process planning and line configuration can enable manufacturers to perform sequencing of operations, line balancing, machine tool and cutting tool allocation and in an integrated manner. Several constraints are respected while optimizing a criterion. It gives better control on the planning of the process and better accuracy. Hence, the motivation is to develop such a technique for efficient balancing of flexible transfer lines utilizing single spindle machining centers.

1.5 Organization of thesis

The thesis is organized in the next few chapters as follows: chapter 2 includes definition of the problem and literature review. The MILP model is described in chapter 3. The problem-specific simulated annealing algorithm is presented in chapter 4. Chapter 5 includes numerical experiments conducted to verify the functioning of the proposed methods. Conclusions and recommendations for future work are specified in chapter 6.

CHAPTER 2: LITERATURE REVIEW

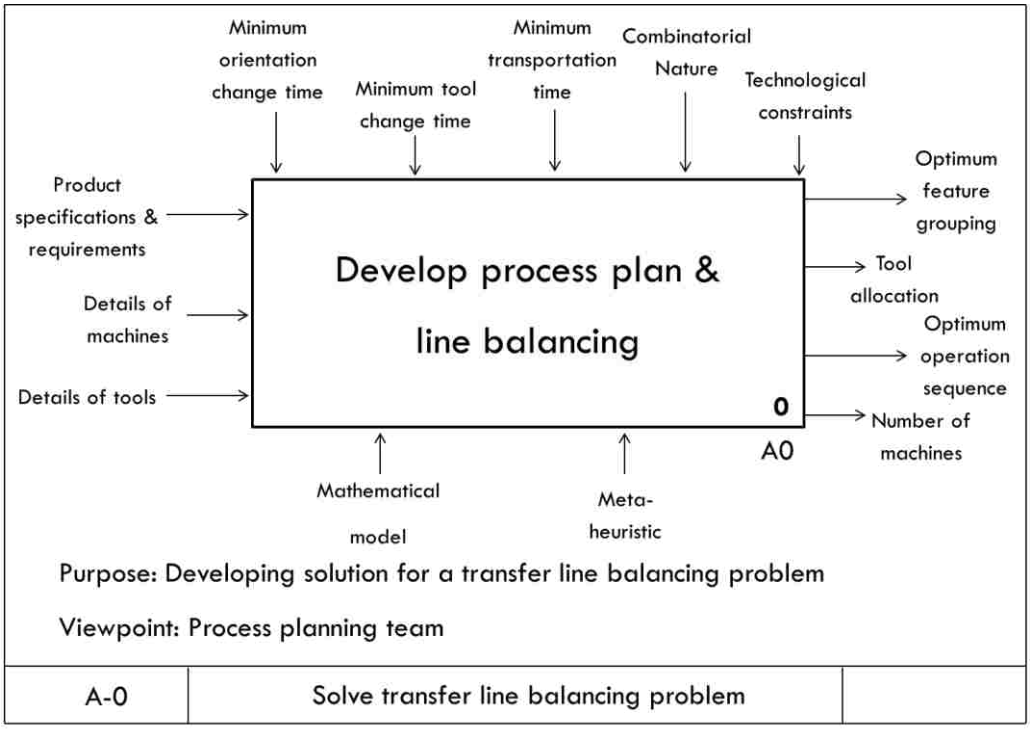
2.1 Problem definition

The simple transfer line balancing problem considered in this thesis involves machine tool and cutting tool allocation, grouping of design features and sequencing of machining operations. Synchronous transfer lines with a straight layout to manufacture large quantities of a complex product are considered. All workstations are identical, consisting of CNC machine tools with a single spindle and tool magazine. The takt time is constant and equal for all machine tools. The objective of the problem is to minimize the handling time portion of the cycle time consisting mainly of orientation change time and tool change time. Various design features are located on different faces of the product. Whenever a feature is processed on one face, the product is rotated if the next feature to be processed lies on a different face. The time spent for this change in orientation is called orientation change time (ORCT). It is time spent for a non-value added activity and hence, it needs to be minimized. If the design features on one face are processed together, the change in orientation can be reduced. Thus, the grouping problem aims to group the design features with an objective of minimizing the ORCT. However, the inclusion and exclusion constraints are to be respected by the grouping problem. The inclusion constraint ensures that two design features are allocated to the same group. The exclusion constraint prevents allocating two features to the same group.

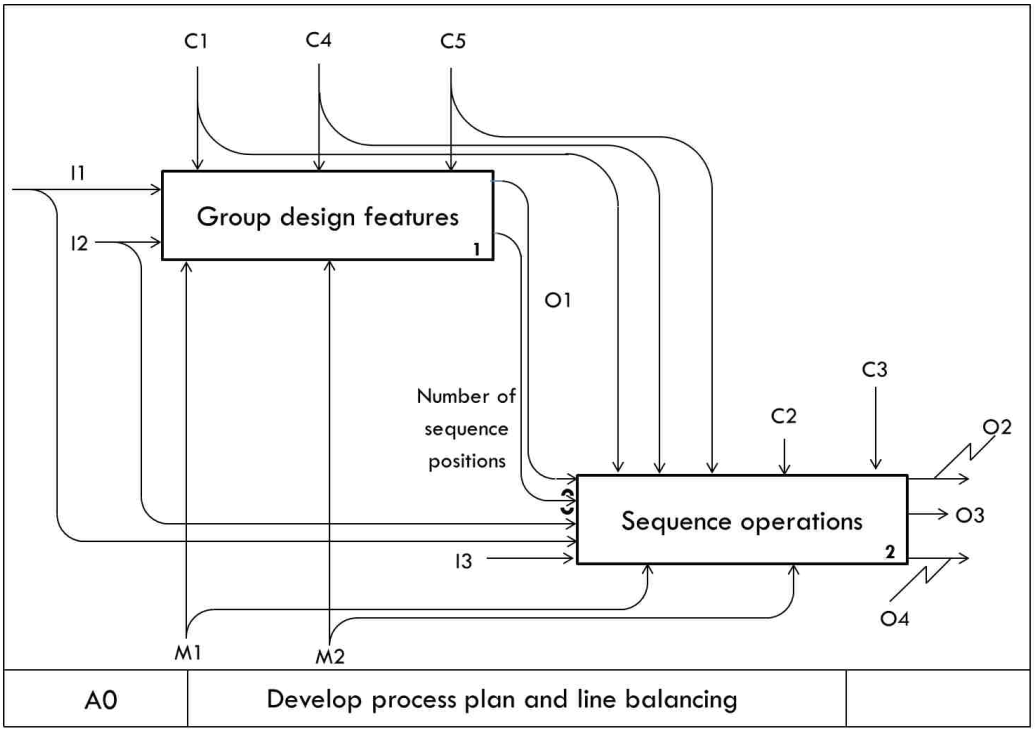
After grouping the features, an optimum sequence for machining operations is to be developed. There are fixed number of operations for each design feature. A specific cutting tool is needed to perform an operation. If the cutting tool required for the next operation is different, the tool spindle needs to go back to the tool magazine to change the cutting tool. The time spent in changing the cutting tool is tool change time (TLCT). It is also a non-value added fraction of the cycle time and it needs to be minimized. Also, ORCT needs to be kept at a minimum while deciding the optimum sequence of

operations. Thus, the objective of the sequencing problem is to minimize a sum of ORCT, TLCT and Transportation time. Transportation time is the time spent for movement of work piece from one machine tool to the other, within a workstation. Precedence constraint ensures that a logical sequence is followed when operations are processed.

Machine tool and cutting tool allocation also require satisfaction of certain constraints. Capacity constraint ensures the number of cutting tools allocated to a machine tool is not more than the tool magazine capacity. Tool life constraint ensures the cutting tool has required life for processing the operations. The cycle also consists of machining time (TO) and refixturing or reloading time (RT). RT is the sum of tool positioning time, tool retracting time and rapid time spent by the cutting tool motion. TO and RT are not affected by the sequence of operations. Takt time limit is to be respected at all workstations. Figure 2 illustrates the nature of the problem. It is an IDEF0 representation including top level and lower level diagrams that follows the Integration Definition for Function Modeling Standard (9).



Top level IDEF Diagram



Lower level IDEF Diagram

Figure 2: Top and lower level IDEF diagrams

2.2 Relevant papers in literature

Due to a high cost involved in set-up of transfer lines, maximization of line utilization is found necessary. A cost-based approach to transfer line balancing is presented in Dolgui and Ihnatsenka (2) and Dolgui *et al*. (10-11). The objective is to minimize the line investment cost which is the sum of equipment and workstation cost. Multi-spindle machining centers are considered and the problem of operation allocation is solved.

Dolgui and Ihnatsenka (2) present a transfer line balancing problem to minimize line investment cost using a branch and bound algorithm. A subset of spindle heads from the available set of blocks to be assigned to workstations is determined while respecting inclusion, exclusion, precedence and cycle time constraints. The branch and bound algorithm develops solution in a reasonable time for the small problems. However, the solution time exponentially increases for medium size problems. Dolgui *et al*. (10) consider a similar problem of equipment block allocation. However, the solution procedure uses an MIP and some heuristic algorithms. First, an exact algorithm based on Mixed Integer Program and calculation of bounds is considered which provides exact solution to small and medium problems in a small time. Second, two heuristic algorithms viz. Random allocation of blocks and Depth-first search technique are provided for solving large size problems for near-optimal solution in a reasonable time. Dolgui *et al*. (11) present an improved mixed integer program to solve large instances of the block allocation problem in a reasonable time.

Transfer line balancing problem with an objective of minimizing number of machines is presented in Essafi *et al*. (1). They consider a flexible transfer line with parallel workstations that utilize CNC machines. A Mixed Integer Program (MIP) with the objective of minimizing the number of machines utilized is presented. Precedence, inclusion and exclusion constraints are specified and sequence dependent set-up times are specified. An algorithm is proposed to reduce the size of the problem by computing range of variables. Long time is

required to solve even a small sized problem when the density of the precedence graph is high. Moreover, the scope is limited to line configuration. No process planning in terms of feature grouping or tool allocation is considered.

A time-dependent objective for line balancing is presented in Das *et al*. (12) and Osman and Baki (13-14). Single spindle machining centers are considered.

Das *et al*. (12) present a grouping and scheduling problem with an objective of minimizing the non-cutting time. The problem is solved in a hierarchical manner using a mathematical model. Machine and tool allocation problems are solved at the higher level assuming a particular sequence. At the lower level, the sequencing model is solved and if a better solution is obtained, the new sequence is provided to the planning model to repeat iteration. This procedure is followed until an optimal solution is obtained. This method is time-consuming and solution time is long for the small problem considered. Transportation time, inclusion and exclusion constraints are not considered.

Osman and Baki (13) provide a decomposition based approach for transfer line balancing with an objective of minimizing non-value added time. Bender's Decomposition is applied to solve small problem instances. Computation time is prohibitive to solve large problems. Transportation time, tool magazine capacity and tool life constraints are not considered.

Osman and Baki (14) develop ant colony meta-heuristic and a hybrid method to solve the problem for medium and large instances. However, computation time is long for large problems. Transportation time between any two machine tools is considered. However, transportation time between two workstations needs to be neglected. Tool magazine capacity constraint considers capacity of a workstation without considering number of machine tools allocated to the workstation.

An investigation on line balancing of an automated cylinder block production transfer line is carried out by Masood (15). A case study is considered to improve cycle time performance and machine utilization. Re-sequencing of operations is

carried out to improve the throughput. The results are validated by simulation. A systematic methodology is not provided for re-sequencing.

Tolio and Urgo (7) present a mixed integer linear program to consider design of flexible transfer lines. The equipment cost for a multi-model rotary transfer line is minimized while respecting design constraints. An industrial case study is considered to analyze the cost for configuration and reconfiguration of the line for three different parts. There is a significant effect of reconfiguration on the cost. Zhang *et al.* (16) provide a hierarchical process planning approach for flexible transfer line schematic design. The activities include the selection of manufacturing feature, machining operation, part set-up planning, feature sequencing, operation sequencing and process plan generation. The evaluation is performed on the basis of quality, flexibility, reliability, machine load and cost.

The machine loading and resource allocation problems for other manufacturing systems are considered in Reddy *et al.* (3), Kim and Suh (4), Persi *et al.* (17), Sarin and Chen (18), Sinreich *et al*. (19), Ecker and Gupta (20) and Lin and Wang (21).

Persi *et al.* (17) present a hierarchical approach to production planning and scheduling for a flexible manufacturing system with the objective of minimizing lead time. At the higher level, part grouping and tool allocation problems are solved using a Mixed Integer Program (MIP). Time and tool magazine capacity constraints are respected. At the lower level, batches are sequenced and parts are scheduled within the batches using dispatching rules. A numerical example demonstrates the efficiency of the model.

Sarin and Chen (18) discuss a machine loading and a tool allocation problem for a flexible manufacturing system. The objective of the problem is to minimize the total cost of machine utilization and cutting tools. A Mixed Integer Program (MIP) is presented with cycle time limit, tool life and tool magazine capacity constraints. A computational example is included in the discussion.

Sinreich *et al*. (19) consider minimization of non-productive machining time in a Single-Stage Multifunctional Machining System (SSMS). The effect of bottleneck resources like setup tasks, machine idle time, machine breakdown and defective jobs on job scheduling is considered. A Binary Integer Program (BIP) is presented and linearized using additional constraints. A heuristic sequencing algorithm is developed and a numerical example is solved using GAMS. The non-productive time of the internal set-up is reduced but at the expense of an increase in the external set-up.

Kim and Suh (4) present an optimal grouping and sequencing technique for a Multi-stage Machining System. A combination of an expert system and an Integer Program (IP) is used to develop a process plan with global and local scope. A heuristic algorithm further reduces the size of the problem. The objective is to minimize the non-cutting time while respecting precedence and tolerance constraints. Line balancing is also taken into consideration. A numerical example is included to verify the functionality of the approach.

Ecker and Gupta (20) present an algorithm to sequence tasks on a machine by reducing the tool change time for a flexible manufacturing system. The precedence constraints of tasks and difference in tool changeover times are taken into consideration. A numerical example is included which illustrates the solution. Simulation experiments are also performed to verify the effectiveness of the heuristic algorithms.

Lin and Wang (21) present an operation planning and sequencing technique for tool changeovers to minimize cost. A two stage integer program is provided. At the first stage, tool allocation is performed while respecting precedence constraints for each part to be produced. The objective is to minimize the total machining and tooling cost. At stage two, an optimal tool changeover sequence is prepared. A numerical example is included to verify performance at both stages. Bhaskara Reddy *et al.* (3) present an operation sequencing solution to minimize tool changeover and set-up and maximize utilization. Accessibility, precedence and geometric constraints are considered. They acknowledge that

operation sequencing problem is difficult to solve using integer programming or branch and bound approaches. A genetic algorithm is provided to solve the problem in a reasonable time.

Simulated annealing is an effective technique to solve large combinatorial optimization problems. It is applied in McMullen and Frazier (22), Pandey *et al*. (23), Azab and Elmaraghy (24) and Suresh and Sahu (26).

McMullen and Frazier (22) present a simulated annealing approach to solve multi-objective assembly line balancing problem. The objectives of the heuristic are to minimize total cost (labor and equipment), to minimize the smoothness index and to minimize the probability of lateness. A Design of experiment is presented with seven problems with different number of tasks. Some of the problems have more than one product and hence a mixed-model sequencing approach is discussed. The results obtained from different selection rules and their comparison is included. Simulated annealing is found to be an effective tool to improve cycle time performance.

Pandey *et al*. (23) solve a multi-objective operation sequencing problem using simulated annealing. The objective functions considered are minimization of setup changeover and tool changeover, maximization of tool motion continuity and loose precedence. Each of these functions is assigned an index and a weighted sum of indices is called Operation Sequencing Rating Index (OSRI). Neighboring solutions are developed each time by Modified Shifting Scheme (MSS). A feature precedence graph is prepared beforehand to reduce computational time.

Azab and Elmaraghy (24) present a macro-level approach to process planning for a reconfigurable manufacturing system. A binary integer program and a simulated annealing algorithm are applied to the problem with the objective of minimizing non-cutting time. Operation sequencing is performed while respecting precedence constraints. An industrial case study is presented to verify the performance of the two methods.

Suresh and Sahu (26) adopt simulated annealing technique to solve the assembly line balancing problem with stochastic task times. A problem-specific simulated annealing algorithm is presented with two different objectives considered independently. The first is to minimize the smoothness index and the other is to minimize the probability of line stopping. Several experiments are performed to solve the problem for different number of tasks. It is observed that better solution is obtained at a lower rate of cooling. However, slower cooling results in an increase in the computational time.

2.3 Gap analysis and novelty of solution

It is observed that transfer line balancing problem can be solved either by a time-based or a cost-based approach. If multi-spindle machining centers are considered, the objective is generally to minimize equipment cost. When considering single spindle machining centers, the objective is generally to minimize non-cutting time. There is a need to develop an efficient solution to solve the latter problem for large instances. Therefore, such a problem is considered here.

Moreover, most of the papers consider only one of the two aspects of transfer line balancing problem: process planning and line configuration. There is a need to consider both process planning and line configuration while solving the transfer line balancing problem. A comparative gap analysis is presented in the Appendix C.

Transfer line balancing problems consider only manufacturing operations but not the design features. Operations belonging to a design feature have certain similar characteristics. For instance, all operations of a feature lie on one face of the product. If this property of operations is taken into consideration, the size of the problem can be considerably reduced. Moreover, if precedence constraints are defined using design features instead of operations, the number of constraint

equations reduces considerably. The reduction in the density of the precedence graph shortens the computation time. Changes in design of the product can be easily implemented. Any change in the design of the product translates into addition or removal of a design feature. The transfer line balancing problem is solved again for a new solution which is implemented by utilizing the flexibility of the transfer line. Hence, the grouping of design features is considered instead of grouping individual operations.

Moreover, the current trend is to utilize flexible transfer lines instead of traditional transfer lines with dedicated machines and a fixed set of tools. The tool allocation problem in flexible transfer lines must consider tool magazine capacity constraint and tool change time. The handling time also consists of transportation time and orientation change time which need to be reduced. The variable number of machines at a workstation is another distinguishing characteristic. A flexible transfer line may allow duplication of machine at a workstation in order to respect the takt time. Therefore, different workstations of the line may have different number of machines allotted to them as per the assigned workload.

Some of these considerations for flexible transfer lines are found in Das *et al.* (12) and Osman and Baki (13-14).

Das *et al.* (12) present a mixed integer linear program for grouping and sequencing of operations. However, the computation time required to solve the small problem considered is long. Moreover, transportation time between workstations is not considered. Inclusion, exclusion and tool life constraints are also not considered.

Osman and Baki (13) present a Bender's Decomposition approach to solve the transfer line balancing problem. However, long computation time restricts the problem size that can be solved.

Osman and Baki (14) develop ant colony and hybrid meta-heuristic methods to solve the transfer line balancing problem. Small, medium and large problem sizes are solved using the proposed methods. Solution time is long for large problems.

Transportation time between any two machine tools is considered. However, transportation time between two workstations needs to be neglected. Tool magazine capacity constraint considers capacity of a workstation without considering number of machine tools allocated to the workstation.

This thesis presents a new Mixed Integer Linear Programming (MILP) model that is developed to solve the problem in an efficient manner. Design features are grouped at the higher level and machining operations are sequenced at the lower level. The required machine tools and cutting tools are allocated to the workstations.

The tool magazine constraint is revised for accurate representation. The number of machine tools allocated to a workstation is considered while calculating the tool magazine capacity of a workstation.

Transportation time between two machine tools of a workstation is considered. The number of machine tool is a variable and the workstations are configured with the necessary machine tools to satisfy the prescribed takt time, minimizing the transportation between the machine tools within the workstation. However, transportation time between workstations is neglected. As the number of workstations is fixed, transportation time between workstations cannot be minimized.

It is observed that simulated annealing is an effective tool for solving line balancing and operation sequencing problems of large size and combinatorial nature. It helps to obtain near-optimal solutions to problems of large size in a reasonable time without getting trapped in a local minimum. Simulated annealing can be effectively applied to transfer line balancing problem. Hence, a problem-specific simulated annealing algorithm is proposed to solve transfer line balancing problem in a reasonable time.

Numerical experiments are performed to assess the performance of the mathematical model and the meta-heuristic. The results from the SA algorithm are compared with the optimal results from the MILP model.

CHAPTER 3: MODEL DEVELOPMENT

The following sub-sections present the assumptions, indices, algorithm and sub-models for the proposed Mixed Integer Linear Programming model.

3.1 Assumptions for MILP model

1) Details of design features and machining operations are known. Machining time, orientation change time, tool change time and refixturing / reloading time and transportation time are known.
2) Every operation requires a specific cutting tool for machining. Cutting tools are identified by numbers.
3) Workstations are identical and any cutting tool can be allotted to any machine group. The number of machine groups is known.
4) Machine groups are allotted the required number of CNC machine tools. The machine tools have one spindle and a tool magazine. Tool magazine capacity is known.
5) The fixtures on each workstation can be rotated to any degree as per processing requirement.
6) All parts visit all workstations while they move at equal speeds. A machine tool processes only one workpiece at a time. There are no buffers between workstations.
7) Transportation time between workstations is not considered. However, transportation time between machine tools within a workstation is considered.

3.2 Indices for MILP model

$g \in (1,2,3,..., G)$ Index set of machine groups

$r \in (1,2,3,..., R)$	Index set of design features
$o \in (1,2,3,..., O^r \forall r)$	Index set of machining operations
$s \in (1,2,3,..., S_g \forall g)$	Index set of sequence positions
$l \in (1,2,3,..., L)$	Index set of cutting tools

3.3 Decomposition algorithm for MILP model

The model is solved in a hierarchical manner. Grouping of design features is performed at higher level. Sequencing problem is solved at the lower level. The model is divided into two sub-models: grouping and sequencing. The grouping sub-model allocates design features to machine groups. The result is provided to the sequencing sub-model to sequence the operations. This result may not always satisfy takt time limit as the constraint appears in the sequencing sub-model.

The sequencing sub-model is solved for each group separately. Machine tool and cutting tool allocation is performed by the sequencing sub-model. If a feasible solution is not obtained for all groups, iteration is repeated. The grouping sub-model is solved again by specifying different inclusion and exclusion relationships between features while respecting design intent. This procedure is repeated until a feasible solution is obtained. Figure 3 depicts the proposed approach for hierarchical grouping and sequencing of machining operations for transfer lines. Figure 4 depicts the algorithm followed to solve the problem.

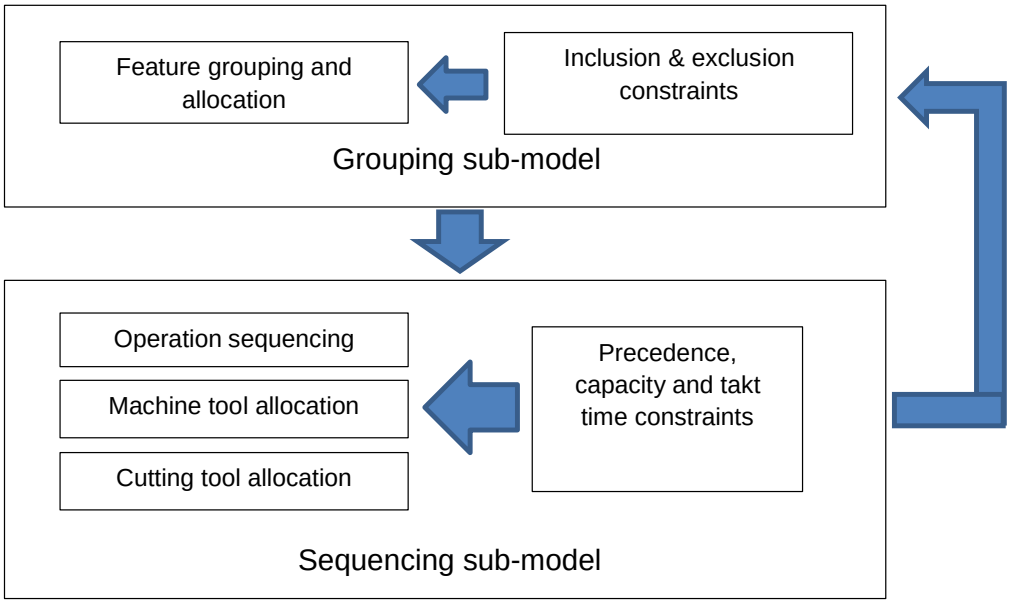


Figure 3: Proposed approach for hierarchical grouping and sequencing of machining operations for transfer lines (MILP model)

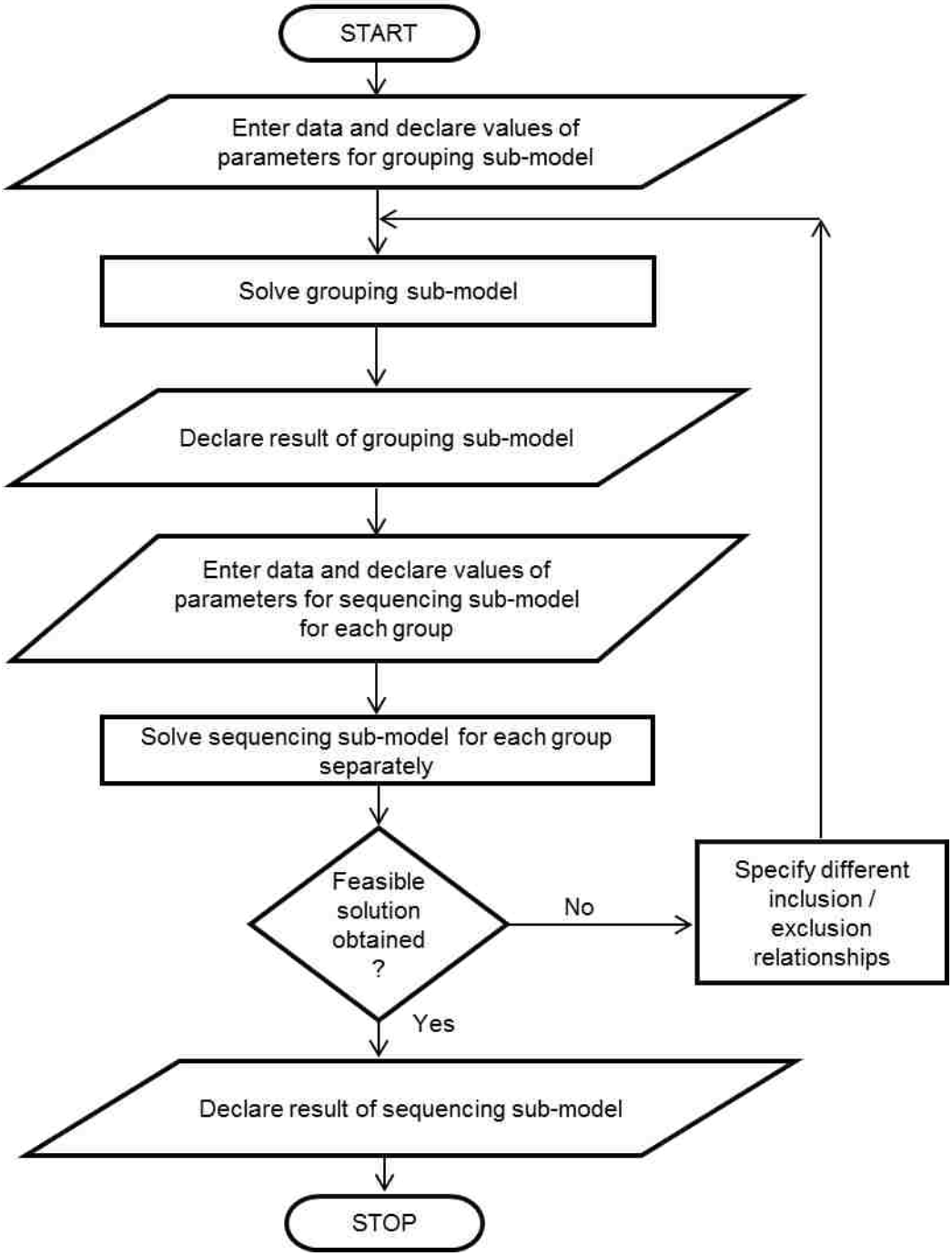


Figure 4: Decomposition algorithm for MILP model

3.4 Parameters for grouping sub-model

O^r Number of machining operations of design feature r

NO Total number of operations in a cycle

$i_{rr'}$ 1 if feature r and feature r' are required to be allocated to one group, 0 otherwise

$e_{rr'}$ 1 if feature r and feature r' are not be allocated to one group, 0 otherwise

f_r Face on which design feature r is located

$ORCT_{rr'}$ Orientation change time after processing design feature r on face f before performing design feature r' on face f'

3.5 Decision variables for grouping sub-model

Q_g 1 if group g is formed, 0 otherwise

Variable Q_g decides formation of group g.

X_{rog} 1 if operation o of design feature r is processed in group g, 0 otherwise

Variable X_{rog} decides allocation of operation o of design feature r to machine group g.

Z_{rg} 1 if Design Feature r is processed in group g , 0 otherwise

Variable Z_{rg} decides allocation of design feature r to group g

S_g Number of sequence positions in group g

Variable S_g decides the number of sequence positions in group g .

Equations (1) and (2) specify other variables for the problem.

Equation (1) determines the total number of operations.

$$NO = \sum_{r=1}^{R} O^r \quad (1)$$

Variable $W1$ calculates the total ORCT for group g .

$$W1(g) = \sum_{r=1}^{R} \sum_{r'=1}^{R} ORCT_{rr'} Z_{rg} Z_{r'g} \quad (2)$$

3.6 Grouping sub-model objective function and Linearization

The objective of the grouping sub-model is to minimize the total ORCT for all groups.

Minimize $\sum_{g=1}^{G} W1(g)$ (3)

Variable $W1(g)$ is defined in equation (2).

As the equation is non-linear in nature, a linearization scheme from Osman and Baki (13) is followed to introduce a new binary variable $B_{rgr'g'}$. It has a value of 1 if design feature r is allotted to group g and design feature r' is allotted to group g' and 0, otherwise. This variable replaces the multiplication of two variables in the objective function of grouping sub-model. Equations (12) and

(13) are added which are auxiliary constraints for the binary variable. Variable $W1(g)$ changes to:

$$W1(g) = \sum_{r=1}^{R} \sum_{r'=1}^{R} ORCT_{rr'} B_{rgr'g} \quad (4)$$

3.7 Grouping sub-model constraints

Equation (5) ensures that each design feature r is allocated to only one group g.

$$\sum_{g=1}^{G} Z_{rg} = 1 \qquad \forall r \quad (5)$$

Equation (6) allows variable values for Z_{rg}. For every group, the value of LHS is zero if no feature is allocated to it and positive if at least one feature is allocated.

$$\sum_{r=1}^{R} Z_{rg} >= 0 \qquad \forall g \quad (6)$$

Equation (7) ensures all operations of a group are processed in the same group in which the feature is allocated.

$$X_{rog} = Z_{rg} \qquad \forall o, r, g \quad (7)$$

Equation (8) represents inclusion constraint to assign the two design features to the same group.

$$(Z_{rg} - Z_{r'g}) = 0 \qquad \forall g, r, r' : r' > r, i_{rr'} = 1 \quad (8)$$

Equation (9) provides exclusion constraint to assign the two design features to the different groups.

$$(Z_{rg} + Z_{r'g}) \leq 1 \qquad \forall g, r, r' : r' > r, e_{rr'} = 1 \quad (9)$$

Equation (10) indicates that a design feature is allocated to a group only when it is formed. M is a big number which facilitates allocation of feature only when Q_g is 1.

$$\sum_{r=1}^{R} Z_{rg} \leq MQ_g \qquad \forall g \qquad (10)$$

Equation (11) calculates the total number of sequence positions allotted to group g viz. S_g.

$$\sum_{r=1}^{R} O^r Z_{rg} = S_g \qquad \forall g \qquad (11)$$

Equations (12) and (13) are auxiliary constraints for linearization.

$$Z_{rg} = \sum_{g'=1}^{G} B_{rgr'g'} \qquad \forall g, r, r' \qquad (12)$$

$$B_{rgr'g'} = B_{r'g'rg} \qquad \forall r, g, r', g' : r' > r \qquad (13)$$

Equations (1)-(13) are adopted from Bhale *et al*. (25).

3.8 Parameters for sequencing sub-model

O^r Number of machining operations of design feature r

D_r Total number of times design feature r is processed for each product

NM_g Maximum number of machine tools allowed for a group

H_l Number of tool slots needed by cutting tool l

T_l Life of cutting tool l

A Size of the tool magazine of each machine tool

E Cycle time

TO_{ro} Time for processing operation o of design feature r

f_r Face on which design feature r is located

$ORCT_{rr'}$ Orientation change time after performing design feature r on face f before performing design feature r' on face f'

$TLCT_{ror(o+1)}$ Tool change time for changing cutting tool l to l' after performing operation o of design feature r before performing operation $o+1$ of the same design feature r

$TLCT_{ror'o'}$ Tool change time for changing cutting tool l to l' after performing operation o of design feature r before performing operation o' of the next design feature r'

RT_{ro} Refixturing or reloading time for operation o of feature r

S_g Number of sequence positions in group g

P_{rol} 1 if operation o of design feature r requires cutting tool l, 0 otherwise

Z_{rg} 1 if Design Feature r is processed in group g, 0 otherwise

$i_{rr'}$ 1 if feature r and feature r' are required to be allocated to one group, 0 otherwise

t Transportation time between two machine tools of a workstation

3.9 Decision variables for sequencing sub-model

N_g Number of machine tools in group g (Integer)

Variable N_g decides the number of machine tools allocated to group g.

X_{rogs} 1 if operation o of design feature r is processed on sequence position s in group g, 0 otherwise

Variable X_{rogs} decides allocation of sequence position to an operation o of design feature r belonging to group g.

$Y_{\lg}$ 1 if cutting tool l is assigned to group g, 0 otherwise

Variable $Y_{\lg}$ decides allocation of cutting tool l to group g.

Equations (14)-(18) specify other variables for the problem.

Variable $W1$ calculates the total ORCT for group g.

$$W1(g) = \sum_{r=1}^{R} \sum_{r'=1}^{R} \sum_{o=1}^{Or} \sum_{o'=1}^{Or'} \sum_{s=1}^{Sg-1} ORCT_{rr'} X_{rogs} X_{r'o'g(s+1)} Z_{rg} Z_{r'g} \qquad (14)$$

Variable $W2$ calculates the total TLCT for group g.

$$W2(g) = \sum_{r=1}^{R} \sum_{o=1}^{Or-1} \sum_{s=1}^{Sg-1} TLCT_{ror(o+1)} X_{rogs} X_{r(o+1)g(s+1)} Z_{rg} + \sum_{r=1}^{R} \sum_{r'=1:r'\neq r}^{R} \sum_{o=1}^{Or} \sum_{o'=1}^{Or-1} \sum_{s=1}^{Sg-1} TLCT_{ror'o'} X_{rogs} X_{r'o'g(s+1)} Z_{rg} Z_{r'g} \qquad (15)$$

Variable $W3$ calculates the total RT for an operation o of feature r for group g.

$$W3(g) = \sum_{r=1}^{R} \sum_{o=1}^{Or} \sum_{s=1}^{Sg} D_r RT_{ro} X_{rogs} \quad (16)$$

Variable $W4$ calculates the total TO for group g.

$$W4(g) = \sum_{r=1}^{R} \sum_{o=1}^{Or} \sum_{s=1}^{Sg} D_r TO_{rou} X_{rogs} \quad (17)$$

Variable $W5$ calculates the total transportation time for group g.

$$W5(g) = t(N_g - 1) \quad (18)$$

3.10 Sequencing sub-model objective function and Linearization

Z_{rg} and S_g are input parameters to the sequencing model as their value is obtained from grouping sub-model. The sequencing model is solved independently for each group.

The sequencing sub-model considers minimization of a sum of ORCT, TLCT and transportation time for a group. The objective function is as follows:

Minimize $W1(g) + W2(g) + W5(g)$ (19)

Variables $W1(g)$, $W2(g)$ and $W5(g)$ are defined in equations (14), (15) and (18) respectively.

As the objective function is non-linear, a linearization scheme is adopted from Osman and Baki (13) to introduce a new binary variable $C_{grosr'o's'}$. It has a value of 1 if operation o of design feature r is allotted to sequence position s and operation o' of design feature r' is allotted to sequence position s' in group g

and 0, otherwise. Equations (30) and (31) are added which are auxiliary constraints for the binary variable.

Variables $W1(g)$ and $W2(g)$ change to:

$$W1(g) = \sum_{r=1}^{R} \sum_{r'=1}^{R} \sum_{o=1}^{Or} \sum_{o'=1}^{Or'} \sum_{s=1}^{Sg-1} ORCT_{rr'} C_{grosr'o'(s+1)} Z_{rg} Z_{r'g} \quad (20)$$

$$W2(g) = \sum_{r=1}^{R} \sum_{o=1}^{Or-1} \sum_{s=1}^{Sg-1} TLCT_{ror(o+1)} C_{grosr(o+1)(s+1)} Z_{rg} + \sum_{r=1}^{R} \sum_{r'=1:r'\neq r}^{R} \sum_{o=1}^{Or} \sum_{o'=1}^{Or'} \sum_{s=1}^{Sg-1} TLCT_{ror'o'} C_{grosr'o'(s+1)} Z_{rg} Z_{r'g} \quad (21)$$

3.11 Sequencing sub-model constraints

Equation (22) ensures that the number of machine tools allocated to a group is within the upper limit for the group.

$$N_g \leq NM_g \qquad \forall g \quad (22)$$

Equation (23) specifies the precedence constraint to ensure that the operation o of feature r precedes operation $o+1$ of the same feature r.

$$\sum_{s'=1}^{s} X_{rogs'} \geq \sum_{s'=1}^{s} X_{r(o+1)gs'} \qquad \forall r, o, g, s \quad (23)$$

Equation (24) specifies the precedence constraint to ensure that the last operation O^r of feature r precedes first operation of the feature r'.

$$\sum_{s'=1}^{s} X_{rO^r gs'} \geq \sum_{s'=1}^{s} X_{r'1gs'} \qquad \forall r, r', g, s : i_{rr'} = 1 \quad (24)$$

Equation (25) ensures that each operation o is assigned only to one sequence position s.

$$\sum_{s=1}^{S_g} X_{rogs} = Z_{rg} \qquad \forall r, o, g \quad (25)$$

Equation (26) ensures that each sequence position s is assigned to only one operation o.

$$\sum_{r=1}^{R}\sum_{o=1}^{O^r} X_{rogs} = 1 \qquad \forall g, s \tag{26}$$

Whenever an operation is assigned to a group, the required cutting tool must be assigned to the group and the tool life limit is satisfied. Equation (27) represents this limit.

$$\sum_{r=1}^{R}\sum_{o=1}^{O^r} P_{rol} D_r TO_{ro} X_{rogs} \le T_l Y_{\lg} \qquad \forall g, s, l \tag{27}$$

Equation (28) indicates tool magazine capacity limit for each machine group. The number of machine tools allocated to a workstation is considered while calculating the tool magazine capacity of the group. The number of machine tools allotted to a group is not considered while calculating the capacity of the group in Das *et al.* (12) and Osman and Baki (13-14).

$$\sum_{l=1}^{L} H_l Y_{\lg} \le AN_g \qquad \forall g \tag{28}$$

Equation (29) specifies the cycle time limit of each machine tool to prevent overloading of machine tools.

$$W1(g) + W2(g) + W3(g) + W4(g) \le EN_g \qquad \forall g \tag{29}$$

Equations (30) and (31) are auxiliary constraints for binary variable $C_{grosr'o's'}$.

$$X_{r'o'gs'} = \sum_{r=1}^{R}\sum_{o=1}^{O^r} C_{grosr'o's'} \qquad \forall g, s, r', o', s' : s \ne s' \tag{30}$$

$$C_{grosr'o's'} = C_{gr'o's'ros} \qquad \forall g, r, o, s, r', o', s' : s \ne s' \tag{31}$$

Equations (22) – (31) are adopted from Bhale *et al*. (25).

CHAPTER 4: SIMULATED ANNEALING

The problem considered is of combinatorial nature. When optimization problems are solved using calculus-based methods, the solution is trapped in the local minimum. Solving large size problems for global optimum solutions using such methods requires long computation time. Hence, random-based search techniques like simulated annealing or genetic algorithm are usually preferred. They provide near-optimal solutions to large problems of combinatorial nature with a better performance than greedy algorithms (26). Simulated annealing is an efficient technique for solving assembly line balancing problem (22, 26) and operation sequencing problem (23, 24). Therefore, a problem-specific simulated annealing algorithm is developed which is presented in the following sections along with introduction to the method in 4.1.

4.1 Introduction

Simulated annealing derives its name from physical annealing of solids (26). Annealing of solids involves heating of the solid to a high temperature (melting point) and slow cooling to a low temperature at a controlled rate. The slow cooling ensures more time is spent near the freezing point of the solid. This facilitates the re-arrangement of the molecules and the solid eventually attains the desired properties. Simulated annealing is a high quality approximation algorithm to solve combinatorial problems for minimizing the objective function using a similar procedure. The algorithm begins when an initial feasible solution is specified along with the value of the control parameter. The control parameter is analogous to the temperature in annealing as the value of control parameter is reduced according to the cooling rate. Similar to energy in annealing, the value of the objective function is calculated for each configuration. Neighboring configurations are generated according to the scheme being followed. The neighboring solution is accepted as a current solution if its energy value is lower.

It may also be accepted if it satisfies the Metropolis criterion. In this criterion, a random number from a uniform distribution between (0, 1) is drawn. It is compared with the acceptance probability which is a ratio of change in energy to the control parameter.

Due to the acceptance of new configuration according to the metropolis criterion, the algorithm also accepts inferior solutions at a probability. This prevents being stuck in a local minimum. A certain number of iterations are performed for each value of the control parameter and then the value is lowered. At high values of the control parameter, more uphill changes are accepted. On the contrary, the number of uphill values being accepted is lower for lower values of the control parameter. The algorithm stops when the control parameter reaches the pre-specified minimum value. For slower cooling, better solution is obtained. However, there is an exponential increase in computational time with a lower rate of cooling (26).

The following parameters are required for implementation of simulated annealing technique:

i) “Initial configuration and solution space
ii) Initial value of control parameter
iii) Neighborhood generation procedure
iv) Cooling rate
v) Stopping criterion” (26).

The general algorithm consists of the following steps:

1) Declare all parameters. Enter initial solution, initial temperature (T_{in}), minimum temperature (T_{min}), number of iterations (N) and cooling rate (CR).
2) Calculate the energy for the initial configuration (E_c).
3) Set the value of n=0.

DO

4) Develop a neighboring solution and calculate new energy (E_n).
5) IF new energy is less than current energy, proceed to 6.
   ELSE IF metropolis criterion is satisfied, proceed to 6.
   ELSE Increment the value of n (n=n+1). Proceed to step 7.
6) New state = Current state ($E_{n=}E_c$). Increment the value of n (n=n+1).
7) IF n<=N, go to step 4.
   ELSE T=CR*T. Go to step 4.

UNTIL Stopping criterion is reached. IF T<=T_{min}, declare final solution. ELSE Go to step 4.

Figure 5 shows flowchart of a general simulated annealing algorithm.

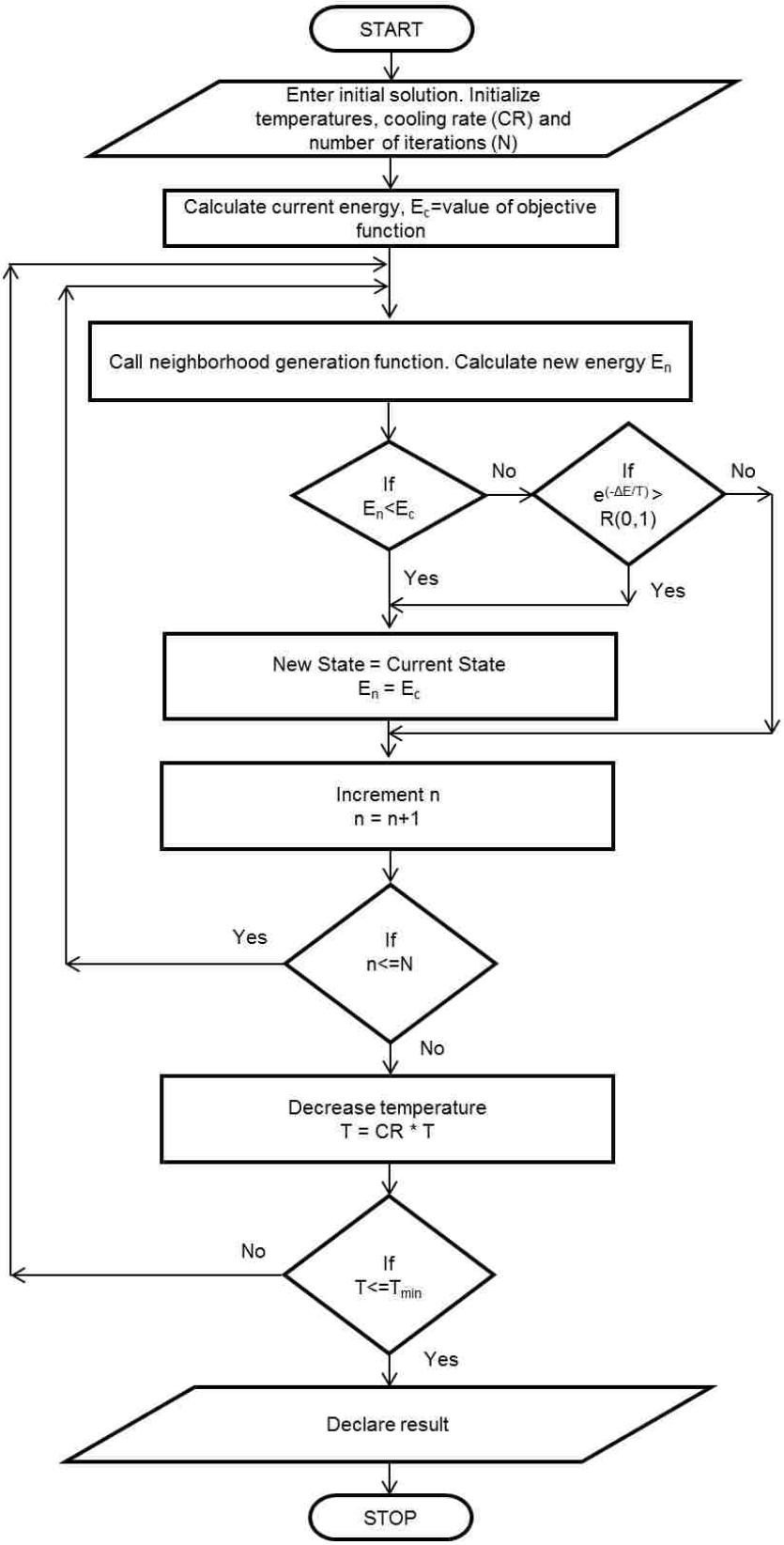


Figure 5: Flowchart of a general simulated annealing algorithm

4.2 Generation of initial solution for grouping problem

The scheme for generation of initial feasible solution for grouping problem respects inclusion and exclusion constraints.

The steps are listed below:

1. Calculate upper limit on the number of design features to be allotted to each group for uniform distribution.

DO

2. Check if all features for which inclusion constraints are specified have been assigned a group.
   IF NO, go to step 3.
   ELSE go to step 4.
3. Select one design feature from those for which inclusion feature is specified and no group is assigned. Assign it to current group along with all features that have inclusion constraints specified for this feature.
4. Assign features for which no inclusion constraint is specified to current group while respecting exclusion constraints and upper limit on feature allocation. Go to next group.

UNTIL all machine groups reach upper limit.

5. If any design feature is not assigned a group, assign it to a random group while respecting exclusion constraints and neglecting upper limit.

4.3 Neighbor generation scheme for grouping problem

The scheme for generation of neighborhood solutions for grouping problem is adopted from Mcmullen and Frazier (22). It involves performing either a trade or transfer at each iteration. A random number is drawn to determine which activity is to be performed for the iteration.

Trade:

Trade involves swapping of two design features between two machine groups. A machine group is selected at random. The last design feature of the group is assigned to the next machine group. At the same time, the first design feature of the next machine group is assigned to the previous machine group.

Consider the example shown in Figure 6. The workstation 2 is selected at random. The last design feature 9 is to be allocated to the next workstation. After trade, the design feature 10 is allocated to workstation 2 and DFU 9 is allocated to workstation 3 as seen in Figure 7.

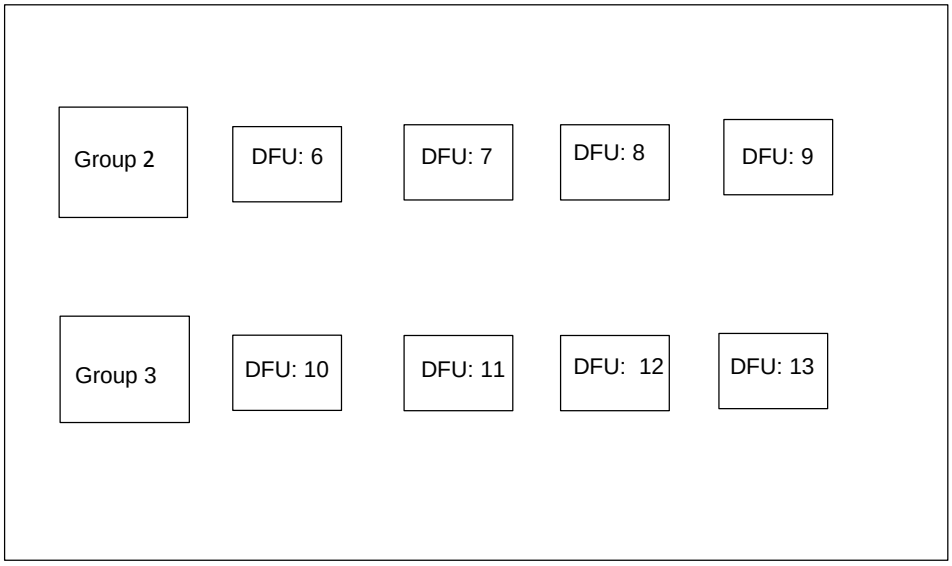


Figure 6: Allocation of Design features to workstations before trade

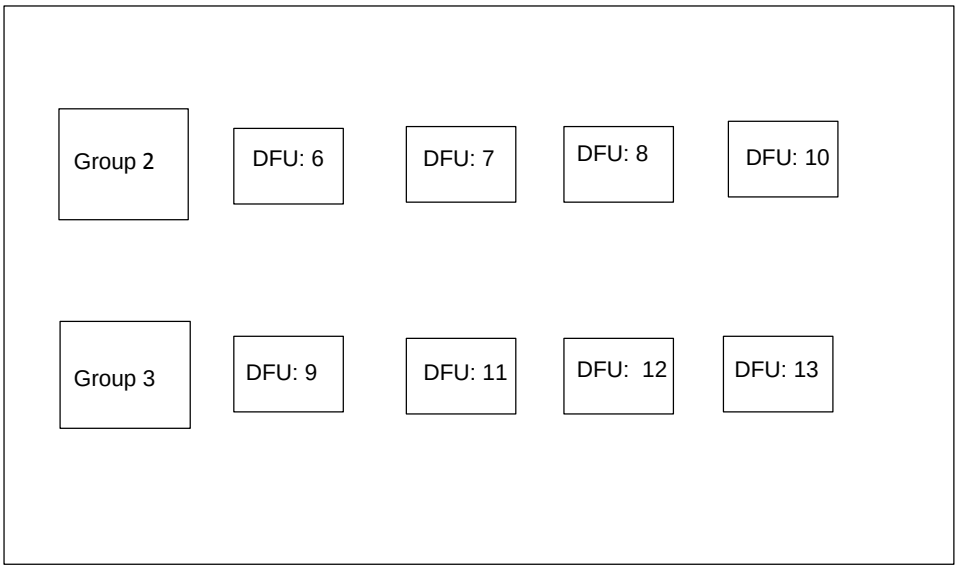


Figure 7: Allocation of Design features to workstations after trade

Transfer:

Transfer involves assigning a design feature of a machine group to the next group without allocating any design feature back to the first group. A machine group is selected at random and the last design feature of the group is assigned to the next group.

An example is shown in Figure 8. Workstation 4 is selected for transfer at random. The last design feature 24 is to be allocated to the next workstation. In Figure 9, it is seen that DFU 24 is allocated to workstation 5 after the transfer is performed.

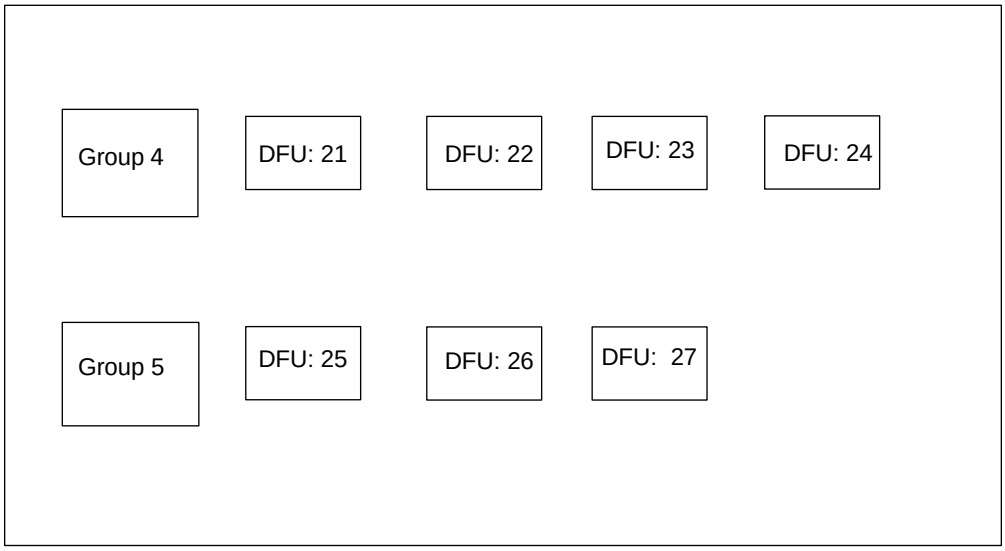


Figure 8: Allocation of Design features to workstations before transfer

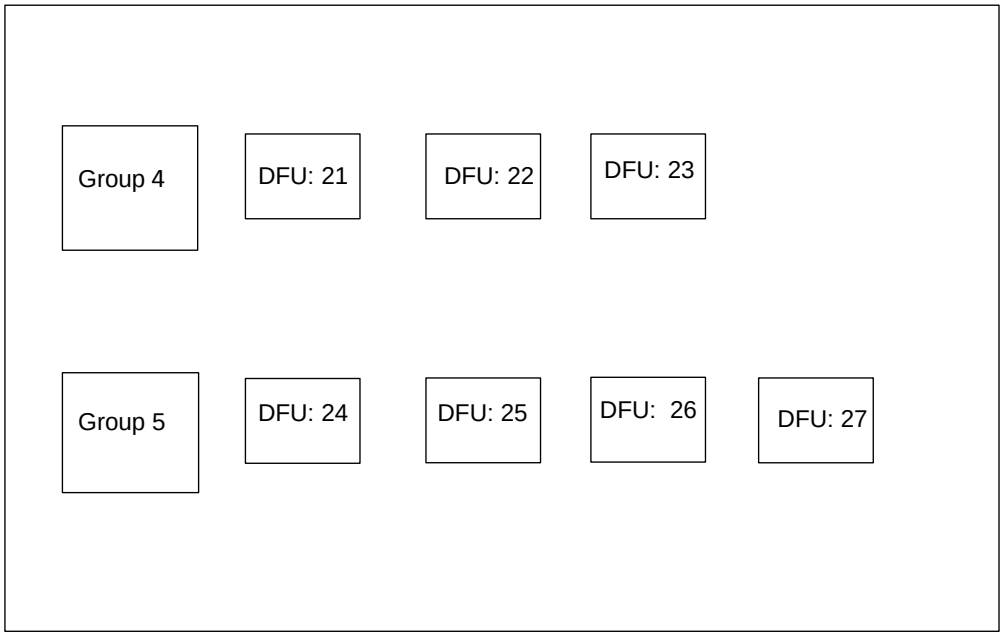


Figure 9: Allocation of Design features to workstations after transfer

4.4 Simulated annealing algorithm for grouping problem

The algorithm for grouping problem allocates design features to machine groups. The objective is to minimize the sum of orientation change time and transportation time within each workstation. It also allocates the required machine tools and cutting tools to the machine groups. Tool life, tool magazine capacity, inclusion and exclusion, upper limit on machine tool allocation and takt time limit are the constraints respected by this algorithm. Figure 10 shows the proposed approach for hierarchical grouping and sequencing using SA algorithm. The steps involved in the proposed simulated annealing algorithm for grouping problem are listed below and depicted in Figure 11.

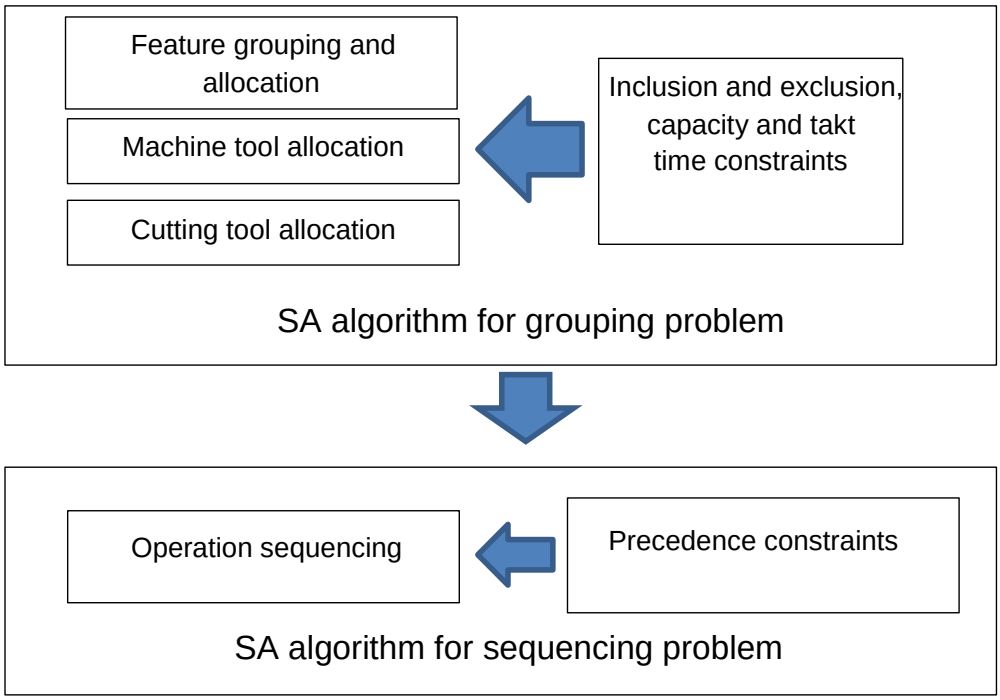


Figure 10: Proposed approach for hierarchical grouping and sequencing of machining operations for transfer lines (SA algorithm)

The algorithm consists of the following steps:

1) Declare all parameters. Generate initial feasible solution while respecting inclusion and exclusion constraints. Enter initial temperature (T_{in}=1000000), minimum temperature (T_{min}=1), number of iterations (N=50) and cooling rate (CR=0.99).
2) Calculate the energy for the initial configuration (E_c).
3) Set the value of n=0.

DO

4) Develop a neighboring solution using either a trade or transfer and calculate new energy (E_n) by calling Grouping energy function.
5) Grouping energy function: Measure ORCT, TLCT, OPT and transportation time and verify takt time limit. Allocate the required cutting tools and machine tools to each machine group. Verify tool life limit, tool magazine limit, and upper limit on number of machines, inclusion and exclusion constraints.
   IF the configuration is feasible, proceed to step 7.
   ELSE Go to step 6.
6) Repair configuration to old state by reversing trade or transfer performed for this iteration. Increment the value of n (n=n+1). Proceed to 9.
7) IF new energy is less than current energy, proceed to 8.
   ELSE IF metropolis criterion is satisfied, proceed to 8.
   ELSE Increment the value of n (n=n+1). Go to step 9.
8) New state = Current state ($E_{n=}E_c$). Increment the value of n (n=n+1).
   IF new solution is better than best solution, accept current solution as best solution. Proceed to 9.
   ELSE Go to 9.
9) IF n<=N, go to step 4.
   ELSE T=CR*T. Go to step 4.

UNTIL Stopping criterion is reached.

IF T<=T_{min}, declare final solution and best solution. ELSE Go to step 4.

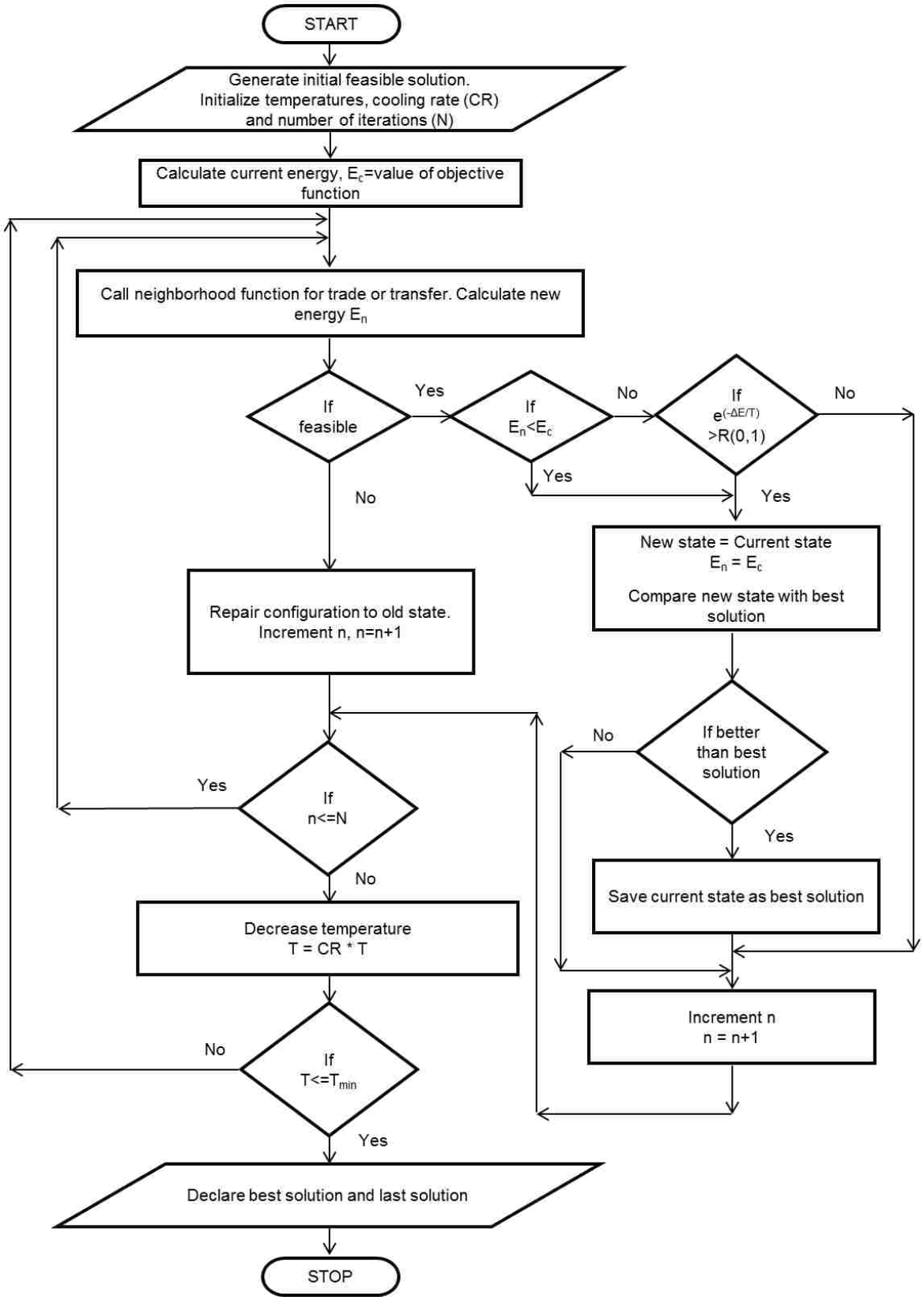


Figure 11: Flowchart of a simulated annealing algorithm for the grouping problem

4.5 Generation of initial solution for sequencing problem

The initial feasible solution for sequencing problems is generated from the best solution obtained from the grouping problem. For each machine group, the initial sequence is obtained by serially placing operations from each design feature allotted to the group in a numerical order. The precedence constraints are satisfied by default. The complete operation sequence from all machine groups is built up and placed into Q which is the initial feasible operation sequence for the sequencing problem.

4.6 Neighbor generation scheme for sequencing problem

The scheme for generation of neighborhood solutions for sequencing problem is called modified shifting scheme (MSS). It is a novel perturbation scheme developed by Pandey *et al.* (23). Whenever the operation sequence is modified, it may lead to an infeasible solution. MSS involves making changes to the operation sequence such that the resulting sequencing is always feasible. This reduces the number of iterations required to reach the optimum solution and also minimizes the search space. The following steps are followed to generate a neighboring solution.

1. Input the current sequence of operations Q and precedence constraint for features.
2. Generate a random number ‘x’ between 1 and total number of operations. This is the operation number which is to be moved.
3. Copy the operation sequence for the machine group to which x belongs from Q into R.

DO

4. Replace x with an operation of R serially starting from 1.

5. IF precedence within a feature or precedence between features constraint is violated, replace x to its original position and go to step 4.
   ELSE go to step 6.
6. Copy R into the machine group portion of Q to which x belongs.

UNTIL R is copied to Q.

Consider an example.

1. Q = {32, 33, 34, 35, 1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 26, 27, 7, 8, 9, 24, 25, 28, 29, 30, 31} and precedence constraint implies for the following pair of design features: [1,4], [2,6], [3,9] and [4,5]. The operations belonging to each design feature are as follows: [1: 1, 2, 3], [2: 4, 5, 6], [3: 7, 8, 9], [4: 10, 11, 12], [5: 13, 14], [6: 16, 17], [7: 18, 19, 20], [8: 21, 22, 23], [9: 24, 25], [10: 26, 27], [11: 28, 29, 30, 31] and [12: 32, 33, 34, 35].
2. A random number is generated x=6.
3. As operation number 6 belongs to group 2, all operations belonging to group 2 are copied to R from Q. R= {1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23}.
4. Replace 1 with 6.
5. Precedence within feature constraint is violated.

4. Replace 2 with 6.

5. Precedence within feature constraint is violated.

4. Replace 3 with 6.

5. Precedence within feature constraint is violated.

4. Replace 4 with 6.

5. Precedence within feature constraint is violated.

4. Replace 5 with 6.

5. Precedence within feature constraint is violated.

4. Replace 10 with 6.

5. Precedence constraints are not violated.

6. Copy updated R = {1, 2, 3, 4, 5, 10, 6, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23} into Q.

New Q = {32, 33, 34, 35, 1, 2, 3, 4, 5, 10, 6, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 26, 27, 7, 8, 9, 24, 25, 28, 29, 30, 31}. This is a feasible neighborhood sequence.

4.7 Simulated annealing algorithm for sequencing problem

The algorithm for sequencing problem develops optimum operation sequence for each machine group. The objective is to minimize the sum of orientation change time and tool change time within each workstation. Precedence constraints are respected by this algorithm. The steps involved in the proposed simulated annealing algorithm for sequencing problem are listed below and depicted in Figure 12.

The algorithm consists of the following steps:

1) Declare all parameters. Generate initial feasible solution while respecting precedence constraints. Enter initial temperature (T_{in}=1000000), minimum temperature (T_{min}=1), number of iterations (N=50) and cooling rate (CR=0.99).
2) Calculate the energy for the initial configuration (E_c).
3) Set the value of n=0.

DO

4) Develop a neighboring feasible solution using modified shifting scheme and calculate new energy (E_n) by calling Sequencing energy function.

5) Sequencing energy function: Measure ORCT and TLCT, the sum of which is the value of energy.
6) IF new energy is less than current energy, proceed to 7.
   ELSE IF metropolis criterion is satisfied, proceed to 7.
   ELSE Increment the value of n (n=n+1). Proceed to 8.
7) New state = Current state ($E_{n=}E_c$). Increment the value of n (n=n+1).
   IF new solution is better than best solution, accept current solution as best solution. Proceed to 8.
   ELSE Proceed to 8.
8) IF n=N, go to step 4.
   ELSE T=CR*T. Go to step 4.

UNTIL Stopping criterion is reached. IF $T=T_{min}$, declare final solution and best solution. ELSE Go to step 4.

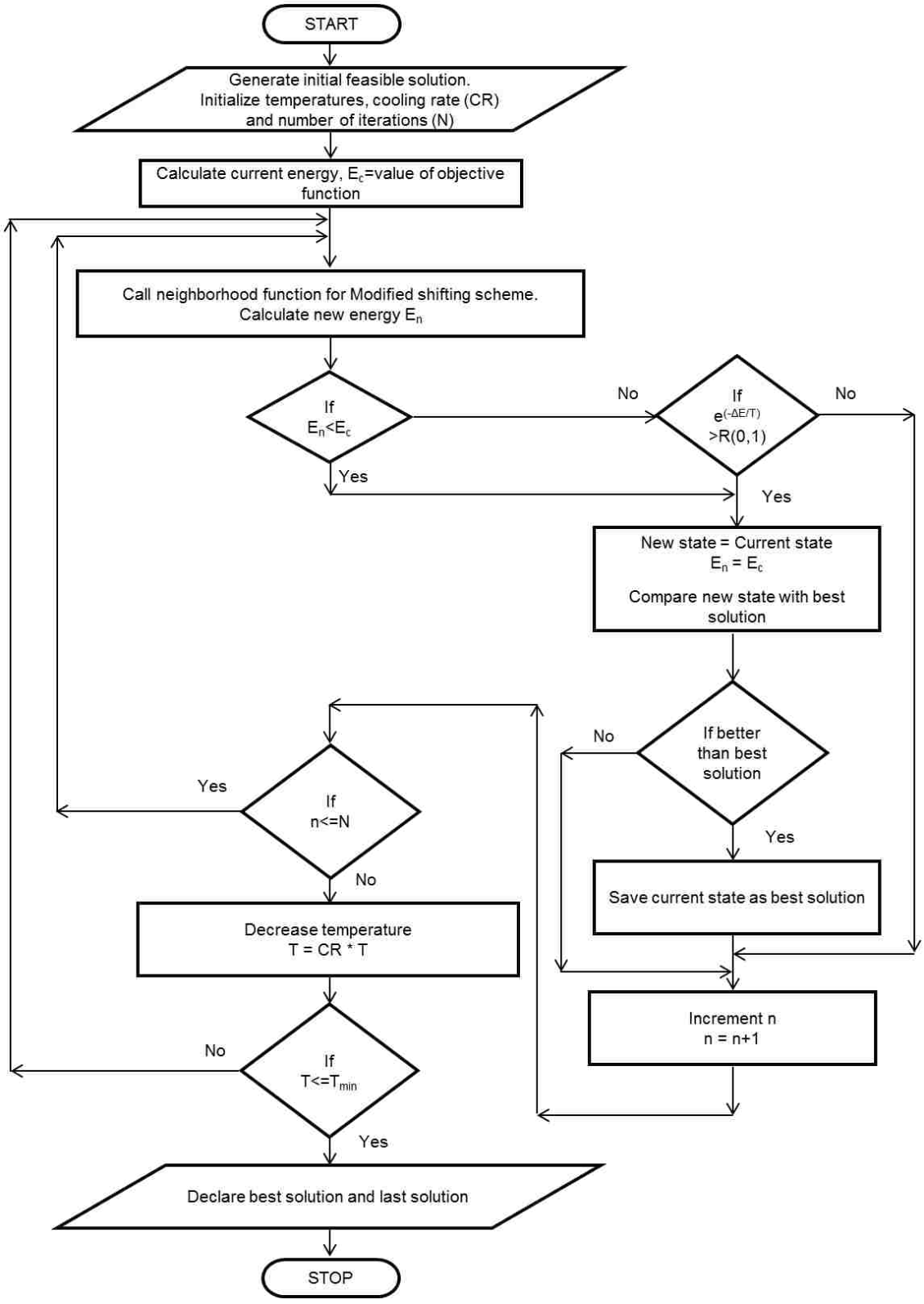


Figure 12: Flowchart of a simulated annealing algorithm for the sequencing problem

CHAPTER 5: NUMERICAL EXPERIMENTS

A number of experiments are conducted to verify the functionality of the MILP model and simulated annealing algorithm. Three case studies are presented in the following sections. The first case study is the benchmark problem from Das *et al.* (12). It is a cylinder head manufacturing problem consisting of 38 operations. The second case study is considered from Osman and Baki (14). Fifteen problems with different sizes are solved. The last case study considers the effect of variation in machining time on machine tool requirement.

5.1 Case study 1

A case study from Das *et al.* (12) is considered. An automotive cylinder head is to be manufactured. It has 12 Design features located on five different faces consisting of 38 machining operations in total. They are to be distributed amongst four machine groups. Twelve different types of cutting tools are required. The orientation change time matrix (in seconds) is presented in Table 2. The details of the operations are specified in Table 3. The tool change time is a random number in the range (1, 10). The value is zero for no change in cutting tool. As the complete data for tool change time is not available, new values are generated. The matrix for tool change time is included in Appendix D. The MILP model is coded using AMPL[27] and solved using CPLEX solver. The simulated annealing algorithm is coded and solved using C++[28] in Microsoft Visual Studio Express. The test is conducted using an Intel i-3 processor @1.4 GHz with 4GB RAM.

Table 2: Orientation change time matrix for the chosen cylinder head benchmark problem (12)

Face	Face				
	1	2	3	4	5
1	--	1.95	2.5	4	5
2	1.95	--	1.95	2.5	4
3	2.5	1.95	--	1.95	2.5

4	4	2.5	1.95	--	1.95
5	5	4	2.5	1.95	--

Table 3: Processing information for the chosen cylinder head benchmark problem (12)

DFU	No. of DFUs	Location on face	Data Type				
1	16	1	UMF	1	2	3	4
			Tool #	1	2	3	4
			No. of slots	4	2	2	2
			TO(s)	2.88	1.25	1.71	2.93
			RT(s)	1.73	0.75	1.03	1.8
2	4	1	UMF	5	6	7	8
			Tool #	1	2	4	5
			No. of slots	4	2	2	2
			TO(s)	1.54	1.57	2.62	1.7
			RT(s)	0.92	0.94	1.57	1
3	8	1	UMF	9	10	11	
			Tool #	1	4	5	
			No. of slots	4	2	2	
			TO(s)	2.75	1.79	2.89	
			RT(s)	1.65	1.07	1.73	
4	8	2	UMF	12	13	14	
			Tool #	6	7	8	
			No. of slots	1	1	1	
			TO(s)	1.95	1.2	1.65	
			RT(s)	1.17	0.75	0.99	
5	4	2	UMF	15		16	
			Tool #	9		10	
			No. of slots	1		1	
			TO(s)	2.68		2.3	
			RT(s)	1.61		1.38	
6	4	3	UMF	17	18	19	20
			Tool #	9	10	11	12
			No. of slots	1	1	3	3
			TO(s)	1.97	3.37	1.36	2.74
			RT(s)	1.18	2.02	0.82	1.4

7	4	3	UMF	21	22	23	
			Tool #	10	11	12	
			No. of slots	1	3	3	
			TO(s)	1.35	1.22	3.09	
			RT(s)	0.81	0.73	1.85	
8	20	4	UMF	24	25	26	
			Tool #	3	4	5	
			No. of slots	2	2	2	
			TO(s)	1.2	1.62	2.06	
			RT(s)	0.72	0.97	1.24	
9	20	3	UMF	27		28	
			Tool #	3		4	
			No. of slots	2		2	
			TO(s)	1.9		3.3	
			RT(s)	1.14		1.98	
10	8	5	UMF	29		30	
			Tool #	1		2	
			No. of slots	4		2	
			TO(s)	2.78		1.31	
			RT(s)	1.67		0.79	
11	8	5	UMF	31	32	33	34
			Tool #	11	12	5	6
			No. of slots	3	3	2	1
			TO(s)	3.43	3.42	3.39	2.5
			RT(s)	2.05	2.05	2.03	1.5
12	4	5	UMF	35	36	37	38
			Tool #	8	9	10	11
			No. of slots	1	1	1	3
			TO(s)	2.64	2.26	2.45	2.32
			RT(s)	1.58	1.36	1.47	1.4

The computational results from MILP model are illustrated in Table 4. The solution to the grouping sub-model is obtained in 1 sec. The total solution time for the sequencing sub-models is 3.37 sec. A total makespan of 1248.64 sec is obtained. The makespan obtained by Das *et al*. (12) are 1256.74 seconds, 1213.64 seconds and 1223.14 seconds with a solution time of 2329 sec, 1694 sec and 2376 sec respectively. As the complete data for tool change time is not available and transportation time is not considered in (12), the makespan is not

compared with original results. However, a total computational time of 4.37 sec is significantly lower than the original run time exhausted by Das *et al.* (12). The original metrics that could express the computational complexity of Das *et al.*'s (12) models are presented in Table 5. Table 6 shows those for the proposed models. Hence, the discrepancies with the run time could be attributed to the development and improvement of the commercial solvers that have taken place since Das *et al.* (12) ran their models.

A histogram in Figure 13 shows the cycle time against takt time at each workstation. Figure 14 shows the operation sequence at each workstation.

The computational results from simulated annealing algorithm are presented in Table 7. The makespan obtained is 1342.64 seconds with a solution time of 2 sec. A shorter solution time can be attributed to simple structure of SA algorithm. The deviation with respect to optimal solution from MILP model is 7.5%. A histogram in Figure 15 shows the cycle time against takt time at each workstation. Figure 16 shows the operation sequence at each workstation.

Table 4: Numerical results showing the grouping and sequence obtained (MILP model)

Machine Group	1	2	3	4	Total
DFUs	10, 11, 12	4, 5, 8	1, 2, 3	6, 7, 9	12
Solution time for sequencing sub-model (sec)	1.5	0.19	0.96	0.72	3.37
No. of machine tools allotted	1	1	1	1	4
Cutting tools allocated	1, 2, 5, 6, 8, 9, 10, 11, 12	3, 4, 5, 6, 7, 8, 9, 10	1, 2, 3, 4, 5	3, 4, 9, 10, 11, 12	-
No. of sequence positions	10	8	11	9	38
Optimum sequence [DFU, Sequence]	[11,31], [11,32], [11,33], [12,35], [10,29], [12,36], [12,37], [11,34], [10,30], [12,38]	[8,24], [4,12], [8,25], [4,13], [4,14], [5,15], [5,16], [8,26]	[1,1], [2,5], [1,2], [1,3], [2,6], [1,4], [2,7], [2,8], [3,9], [3,10], [3,11]	[6,17], [6,18], [7,21], [9,27], [9,28], [6,19], [7,22], [6,20], [7,23]	-

ORCT (sec)	0	10	0	0	10
TLCT (sec)	23	21	23	15	82
Transportation time (sec)	0	0	0	0	0
TO + RT (sec)	277.28	249.76	367.76	262.04	1156.84
Makespan (sec)	300.28	280.76	390.76	277.04	1248.84

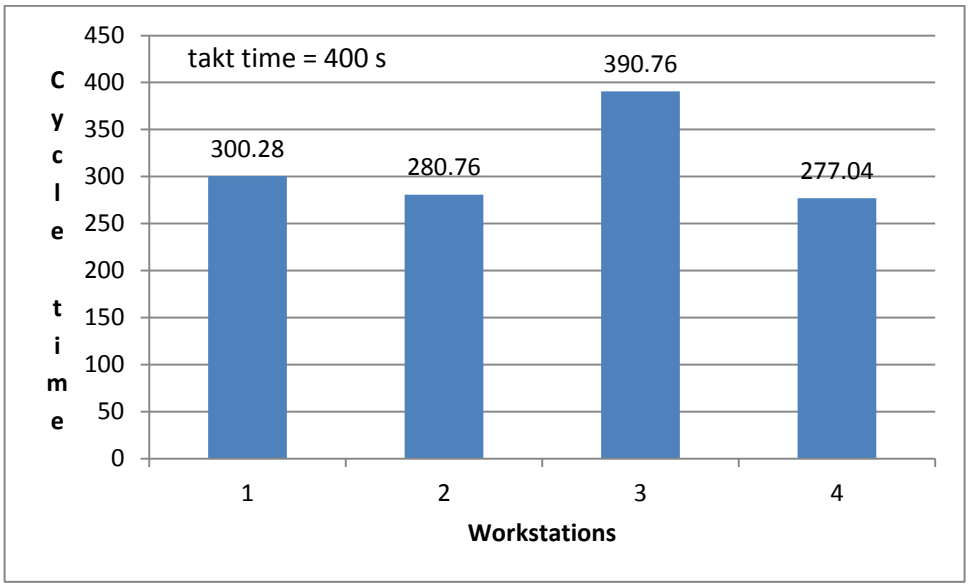


Figure 13: Cycle times against takt time for transfer line workstations (MILP model)

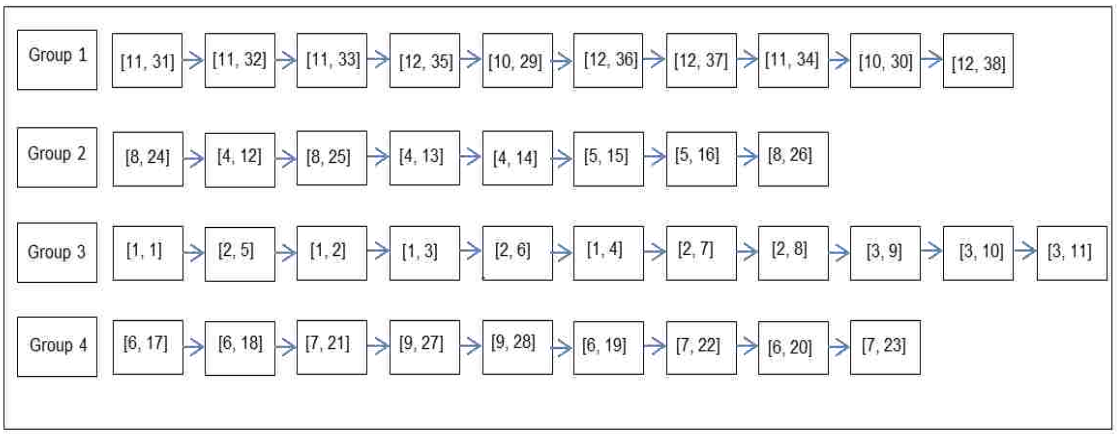


Figure 14: Vector diagram showing operation sequence at each workstation (MILP model) [DFU, Operation]

Table 5: Size complexity metrics for Das *et al.*'s planning and sequencing model (12)

Item	Planning model	Sequencing model			
		Group 1	Group 2	Group 3	Group 4
Total number of variables	1418	17,683	14, 737	11, 867	11, 811
Number of integer variables	828	17,616	14, 670	11, 800	11, 744
Number of constraints	768	3292	1894	956	964
CPU time	18 min 14 sec	18 min 39 sec	1 min	19 sec	2 sec

Table 6: Size complexity metrics for proposed grouping and sequencing models

Item	Grouping sub-model	Sequencing sub-model			
		Group 1	Group 2	Group 3	Group 4
Total number of variables	2508	9113	3661	13444	5926
Number of integer variables	2508	1	1	1	1
Number of constraints	1816	10082	4162	14698	6608
CPU time	1 sec	1.5 sec	0.19 sec	0.96 sec	0.72 sec

Table 7: Numerical results showing the grouping and sequence obtained (SA algorithm)

Machine Group	1	2	3	4	Total
DFUs	1, 4, 5	2, 6, 8	3, 7, 10, 11	9, 12	12
Solution time for sequencing sub-model (sec)	-	-	-	-	2
No. of machine tools allotted	1	1	1	1	4
Cutting tools allocated	1, 2, 3, 4, 6, 7, 8, 9, 10	1, 2, 3, 4, 5, 9, 10, 11, 12	1, 2, 4, 5, 6, 10, 11, 12	3, 4, 8, 9, 10, 11	-
No. of sequence positions	9	11	12	6	38
Optimum sequence [DFU, Sequence]	[1,1], [1,2], [1,3], [4,12],	[2,5], [2,6], [2,7], [2,8],	[3,9], [3,10], [3,11], [7,21],	[9,27], [9,28],	-

	[1,4], [4,13], [4,14], [5,15], [5,16]	[6,17], [6,18], [6,19], [6,20], [8,24], [8,25], [8,26]	[7,22], [10,29], [7,23], [10,30], [11,31], [11,32], [11,33], [11,34]	[12,35], [12,36], [12,37], [12,38]	
ORCT (sec)	5.85	4.45	10	2.5	22.8
TLCT (sec)	35	69	34	25	163
Transportation time (sec)	0	0	0	0	0
TO + RT (sec)	318.84	263.08	346.6	228.32	1156.84
Makespan (sec)	359.69	336.53	390.6	255.82	1342.64

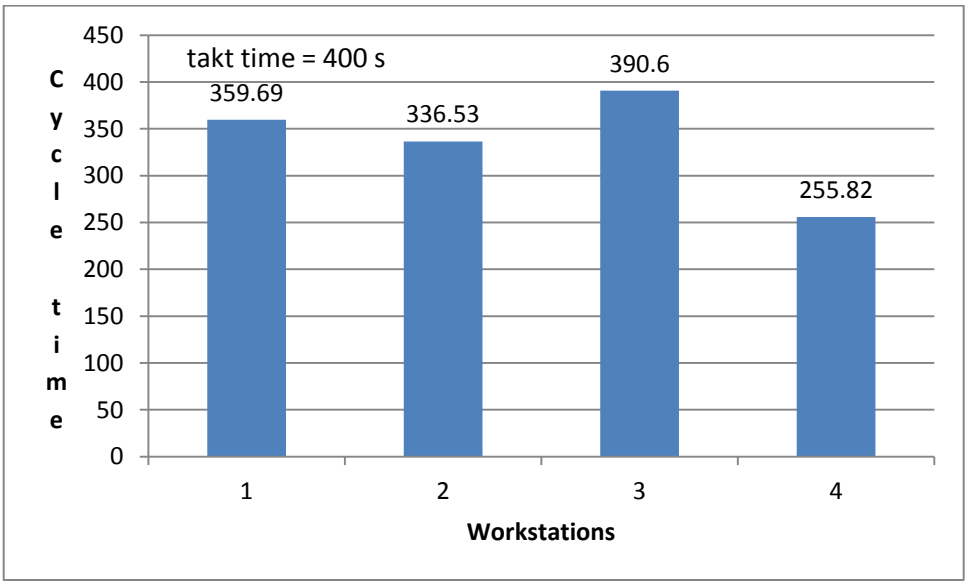


Figure 15: Cycle times against takt time for transfer line workstations (SA algorithm)

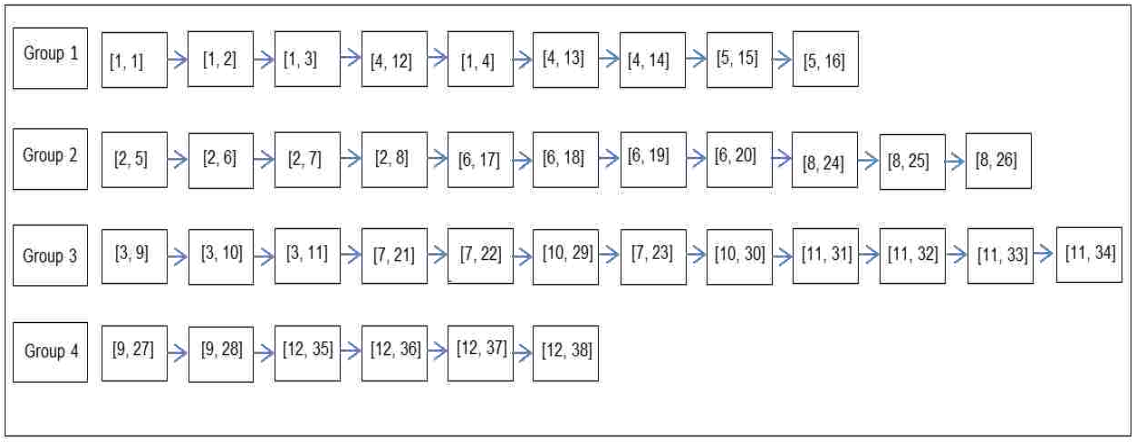


Figure 16: Vector diagram showing operation sequence at each workstation (SA algorithm) [DFU, Operation]

5.2 Case study 2

A case study from Osman and Baki (14) is considered. It consists of 15 problems with different configurations. The problem size is small for the first seven problems, medium for 8-12 and large for 13-15. The configuration of the problems is listed in Table 8. The number of design features, operations and groups are listed in the second column and the inclusion, exclusion relationships are listed in the adjacent columns. The number of operations for a feature is a random number between 2 and 4. Other parameter settings are specified in Table 9. The original results from Osman and Baki (14) are presented in Table 10.

In order to take into account the nature of parameter settings, a total of 10 runs are performed for each of the 15 problems. The mean values of solution time, handling time and makespan are calculated for results from both MILP model and SA algorithm. The comparison is presented in Table 11 and Figures 17-19. The detail results for individual runs for each problem are included in Appendix E.

Table 8: Configuration of the tested benchmark problems (14)

Problem Number	R / O / G	Inclusion relationships	Exclusion relationships
1	3 / 14 / 2	[1,2]	[2,3]
2	4 / 10 / 2	[1,2]	[1,3]
3	5 / 16 / 3	[1,2], [1,4]	[3,4]
4	6 / 17 / 2	[1,4], [3,5]	[3,6], [4,6]
5	6 / 18 / 2	[1,4], [3,5]	[3,6], [4,6]
6	7 / 19 / 3	[2,3], [2,5]	[5,7], [3,6]
7	8 / 22 / 2	[3,4], [5,7]	[4,7], [6,8], [7,8]
8	10 / 30 / 3	[1,4], [2,7]	[6,9], [7,10]
9	12 / 35 / 4	[1,4], [2,6], [4,5], [3,9]	[6,9], [6,10], [9,10]
10	18 / 46 / 4	[1,4], [2,3], [8,11], [11,15], [5,9], [5,7], [5,12]	[1,5], [1,11], [1,16], [2,10], [5,16], [10,12], [11,12]

11	15 / 41 / 5	[5,7], [5,10], [11,15], [12,13]	[1,13], [6,9], [8,10], [8,11], [12,15]
12	15 / 52 / 5	[5,7], [5,10], [11,15], [12,13]	[1,13], [6,9], [8,10], [8,11], [12,15]
13	20 / 64 / 5	[1,4], [1,6], [1,8], [5,9], [5,7], [5,10]	[1,16], [1,11], [10,12], [11,12]
14	25 / 80 / 5	[1,14], [1,19], [21,23], [19,22], [23,24], [5,11], [10,16]	[1,11], [1,13], [1,16], [1,21], [5,14], [13,17], [10,11]
15	30 / 96 / 6	[1,14], [1,18], [5,19], [20,30], [21,24], [22,28], [19,22]	[1,11], [1,13], [13,17], [1,16], [5,14], [18,30], [10,11], [10,12], [11,12]

Table 9: Parameter settings in the tested benchmark problems (14)

Parameter	Data range	Parameter	Data range
$ORCT_{rr'}$	Uniform (1, 5) s	TO_{ro}	Uniform (1, 5) s
$TLCT_{ror(o+1)}$ / $TLCT_{ror'o'}$	Uniform (2, 7) s	RT_{ro}	Uniform (1, 2) s
H_l	Uniform (1, 4)	NM_g	Uniform (2, 5) machine
T_l	Uniform (2000, 3000) s	E	Uniform (330, 550) s
A	Uniform (60, 80) tool	t	Uniform (5, 10) s
D_r	Uniform (5, 12) unit	f_r	Uniform (1, 5)

Table 10: Original computational results from Osman and Baki (14)

Problem No.	Benders Decomposition		Hybrid Benders-ACO			Nested ACO		
	Solution time	Handling time (s)	Solution time	Non-productive time (s)	% from optimal	Solution time	Handling time (s)	% from optimal
1	31.34 s	361.01	3.38 s	361.01	0	2.55 s	361.01	0
2	2 s	56.03	9 s	56.03	0	7 s	56.03	0
3	13.4 min	82.94	30 s	84.07	1.36	13 s	84.27	1.6
4	7.2 min	86.7	8 s	89.6	3.34	3.9 s	87.9	1.38
5	1.9 h	91	16 s	93.2	2.41	2.7 s	93.6	2.85
6	22.5 min	94.41	15.96 s	95.72	1.38	8.55 s	98.05	3.8
7	1.7 h	110.49	41.57 s	114.96	4.04	10.05 s	114.37	3.51
8	>72 h	-	7.75	142.75	-	14.7	139.2	-

			min			min		
9	>72 h	-	4.65 min	175.1	-	14 min	185	-
10	>72 h	-	19.75 min	217.75	-	2.14 h	214.29	-
11	>72 h	-	1.45 h	227.96	-	3.35 h	224.8	-
12	>72 h	-	53 min	245.01	-	2.6 h	249.9	-
13	>72 h	-	16.39 h	296.76	-	3.79 h	310.07	-
14	>72 h	-	>24 h	-	-	19.1 h	366.41	-
15	>72 h	-	>24 h	-	-	20.4 h	464.15	-

Table 11: Computational results for the tested benchmark problems

Problem no.	MILP model			Simulated annealing algorithm				
	Mean Solution time	Mean Handling time (s)	Mean Makespan (s)	Mean Solution time	Mean Handling time (s)	Mean Makespan (s)	% deviation of sub-optimal handling time from optimal solution	% deviation of sub-optimal makespan from optimal solution
1	0.19 s	59.28	615.25	<0.1 s	59.28	615.25	0	0
2	0.15 s	39.55	415.76	<0.1 s	41.26	427.2	4.32	2.75
3	0.25 s	65.03	705.6	< 0.1 s	66.8	707.39	2.72	0.25
4	1.88 s	77.71	738.42	1 s	83	742.62	6.8	0.56
5	1.96 s	81.22	783.22	1 s	89.82	791.83	10.58	1.1
6	0.36	78.66	816.83	1 s	84.48	822.58	7.39	0.7
7	5.23 s	103.07	925.51	1.1 s	116.83	939.27	13.35	1.49
8	7.2 s	136.04	1318.36	2.2 s	147.12	1329.44	8.14	0.84
9	2.31 s	155.06	1497.08	2.2 s	164.37	1512.52	6	1.03
10	24.46 s	218.91	1954.64	4.4 s	231.79	1967.5	5.88	0.66
11	6.17 s	178.48	1740.16	2.8 s	195.53	1757.22	9.55	0.98
12	14.67 s	236.49	2241.23	4 s	254.3	2259.04	7.53	0.79

13	5.14 min	301.59	2819.96	6.6 s	323.03	2843.7	7.1	0.84
14	4.37 h	380.39	3447.17	10.8 s	431.37	3498.11	13.4	1.47
15	>24 h	-	-	11.4 s	442.31	4240.03	-	-

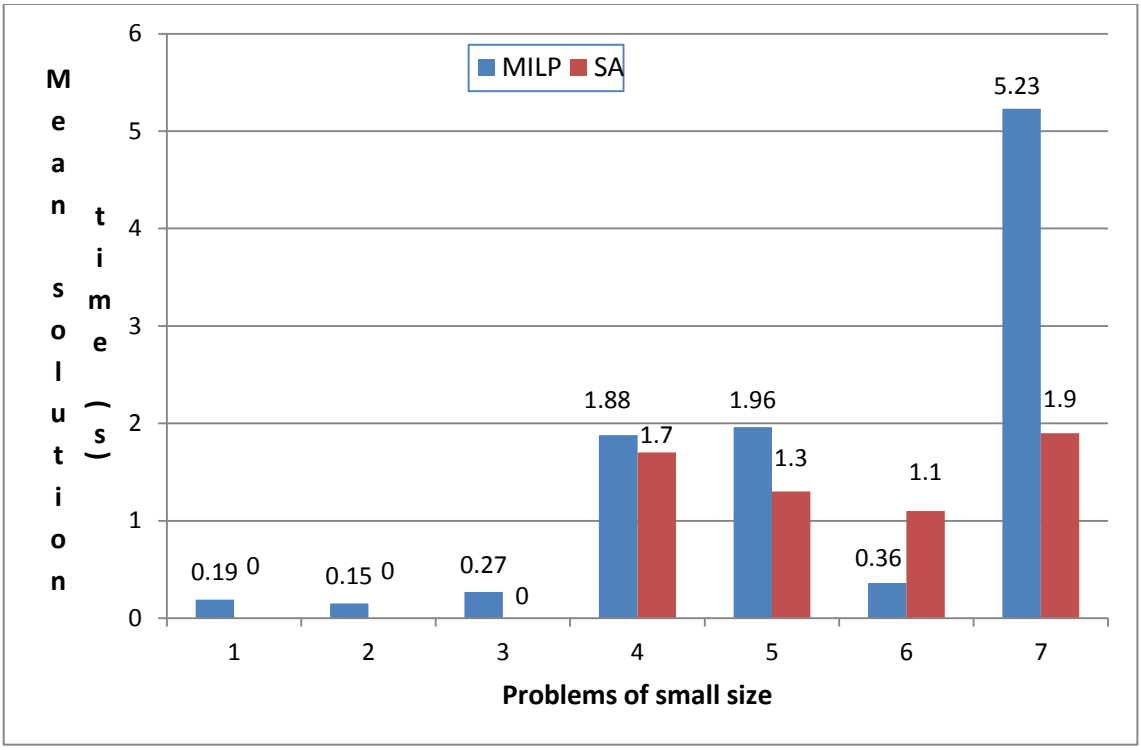


Figure 17: Comparison of mean solution time required for MILP model and SA algorithm for problems of small size

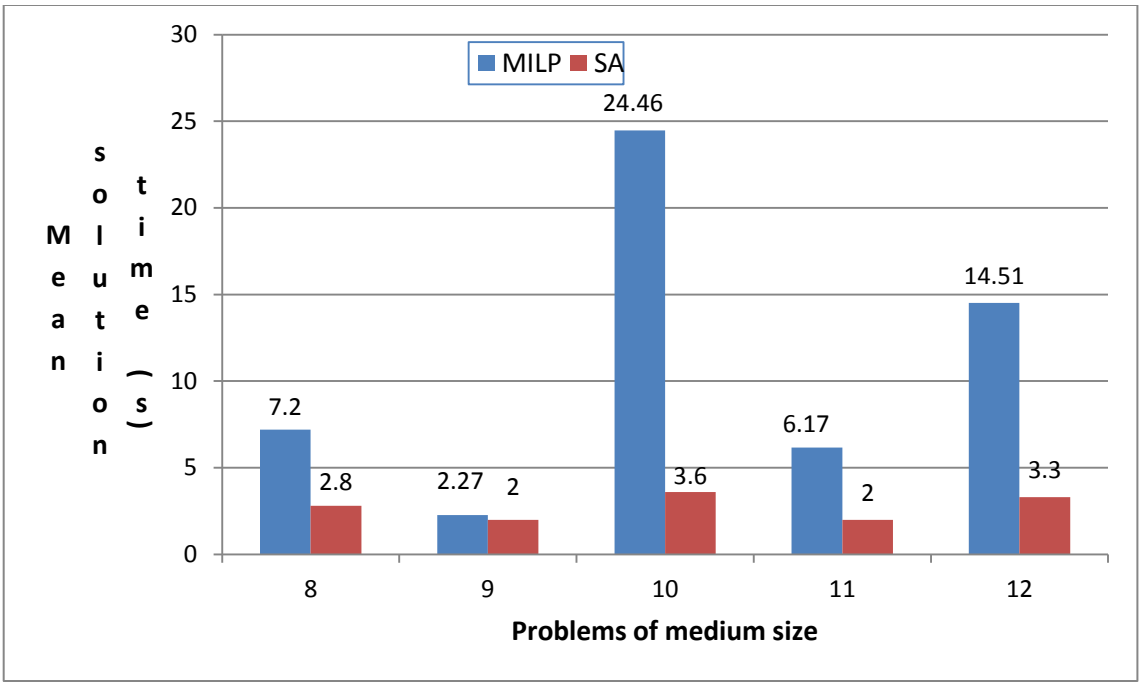


Figure 18: Comparison of mean solution time required for MILP model and SA algorithm for problems of medium size

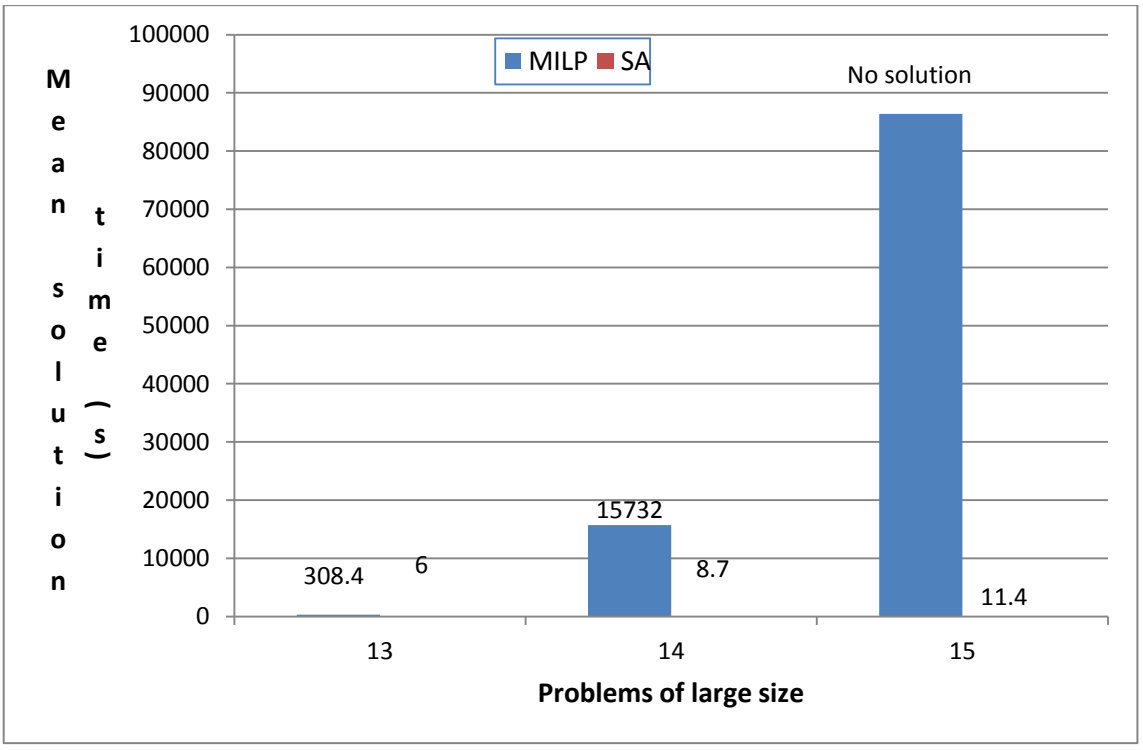


Figure 19: Comparison of mean solution time required for MILP model and SA algorithm for problems of large size

The settings and ranges of the parameters for the case study are given in Table 9. However, the exact values of the parameters being used Osman and Baki (14) are not available. Hence, only a comparison of the elapsed computational time is conducted and not that of the objective function value.

The mean solution time with MILP model is less than the original solution time. As the sequencing problem is solved individually for each workstation, the sequencing problem is solved for (g) number of times. The original Bender's Decomposition method (14) involves solving the sequencing problem (r x g) number of times. Hence, the total time required for solution is lower with the proposed method.

However, the solution time increases exponentially for large problems. The model is able to solve 14 of the total 15 problems within reasonable time.

The solution time for simulated annealing algorithm is further less than that of the MILP model. Therefore, large problems can be solved using this method within a short time. The result for problem 15, which is not solved by MILP model in a reasonable time, is obtained within a short time. The quality of solution is compared with the optimal solution from MILP model by calculating percentage deviation in Table 11. An average deviation of 7.34% is obtained for handling time and 0.96% for makespan.

5.3 Case study 3

This case study considers the effect of variation in processing time of operation on machine tool requirement. With higher cutting speeds, the machining time can be reduced along with a reduction in number of machine tools utilized.

Problem 15 from case study 2 is considered. The Machining time is varied between 4 intervals of the original range (1, 5). All other parameters remain unchanged. The results are included in Table 12. The machine tool requirement decreases for lower machining time. Also, the makespan is lower for lower machining time.

Table 12: Numerical results showing effect of cutting speed on cycle time and machine tool requirement (SA algorithm)

No.	Machining time TO (sec)	Cycle time (sec)					Make span (s)	Total no. of machine tools required
		Workstation 1	Workstation 2	Workstation 3	Workstation 4	Workstation 5		
1	(1,2)	697.75	551.85	603.46	702.65	375.51	2931	9
2	(2,3)	889.75	710.85	769.46	906.68	485.47	3762	10
3	(3,4)	1326.98	869.85	1058.18	1114.64	220.69	4590	12
4	(4,5)	1279.71	1034.81	1107.42	1314.61	693.47	5430	14

However, a higher cutting speed increases wear of cutting tools. The effect of cutting parameters on cutting tool life is studied by Lajis *et al.* (29). A model is presented to calculate cutting tool life in end milling of hardened steel using the following equation:

$$TL = 16771 V^{-3.02} d^{-0.57} f^{-1.14}$$

They conclude that tool life decreases with increase in cutting speed followed by feed rate and depth of cut. Therefore, selection of cutting condition should be performed considering the cutting parameters. An expert system is proposed by Arezoo *et al.* (30) for selection of cutting tool and condition. It uses a knowledge based system to determine the optimum cutting tool and condition for an operation. The selection of cutting condition is another step of process planning which has not been considered here.

With reduction of tool life, the cutting tools wear out faster. A replacement with new cutting tools would be required. Normally, an extra spare tool would be plugged into the tool magazine to replace the worn-out tool before its life expires. For accuracy, this would result in additional tool change times, which is not really incorporated in the model. Due to shortage of data about the exact machining parameters being used by the benchmarked case studies, this part could not be incorporated and added as extra tool life constraints in the sequencing sub-model. However, one could argue as well that the effect of this might not be as significant, since tool change times are not as significant components as transportation time in the overall handling time objective function being taken. Another implication of the extra tool change is the cost. The total equipment cost needs to be considered. Future work in this field may consider a total cost of machine tools and cutting tools to arrive at a pareto optimal solution.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

A simple transfer line balancing problem is considered with a focus on process planning and line configuration. A hierarchical approach of grouping design features and sequencing machining operations is adopted. The objective is to minimize the handling time fraction of the cycle time consisting mainly of orientation change time and tool change time. Several technological constraints of inclusion, exclusion, precedence and takt time limit are considered. A revised tool magazine capacity constraint is considered for accurate representation. A balanced transfer line ensures maximum utilization of machine tool and higher productivity.

A new mixed integer linear programming model is proposed. A meta-heuristic is an effective technique to solve this problem for large instances. Simulated annealing is an efficient algorithm for process planning and operation sequencing as seen in literature. Hence, a problem-specific simulated annealing algorithm is developed to solve this problem.

Rigorous numerical experiments are conducted. Optimal results are obtained with MILP model for problems of small and medium size within a reasonable time. The solution time increases exponentially for large problems. The simulated annealing algorithm is able to solve problems of all sizes within a short time with near-optimal results. This performance with respect to optimality and computation time is better than that of similar methods considered in the literature.

A time-based approach has been followed in this thesis. An extension to this thesis may involve multi-objective optimization to solve the transfer line balancing problem. Both production time and equipment cost may be considered to reach a pareto optimal solution. Goal programming can be a useful tool for the formulation of a new mathematical model. Multi-threaded simulated annealing may be applied to develop a new problem-specific algorithm for multi-objective optimization.

BIBLIOGRAPHY

1. Essafi M, Delorme X, Dolgui A, Guschinskaya O. A MIP approach for balancing transfer line with complex industrial constraints. Computers & Industrial Engineering. 2010; 58(3):393-400.
2. Dolgui A, Ihnatsenka I. Branch and bound algorithm for a transfer line design problem: Stations with sequentially activated multi-spindle heads. European Journal of Operational Research. 2009; 197(3):1119-32.
3. Reddy SB, Shunmugam MS, Narendran TT. Operation sequencing in CAPP using genetic algorithms. International Journal of Production Research. 1999; 37(5):1063-74.
4. Kim I-T, Suh H-W. Optimal operation grouping and sequencing technique for multistage machining systems. International Journal of Production Research. 1998; 36(8):2061-81.
5. Halevi G, Weill RD. Principles of process planning: a logical approach. Springer; 1995.
6. Boysen N, Fliedner M, Scholl A. Assembly line balancing: Which model to use when? International Journal of Production Economics. 2008; 111(2):509-28.
7. Tolio T, Urgo M. Design of flexible transfer lines: A case-based reconfiguration cost assessment. Journal of Manufacturing Systems. 2013; 32:325-334.
8. Sharma IR. Latest trends in machining [Internet]. 2001 [updated 2001 Jan 1; cited 2014 Jan 23]. Available from: http://drishtikona.files.wordpress.com/2012/08/ch-all.pdf
9. Integration Definition for Function Modeling (IDEF0) [Internet] 1993 [updated 1993 December 21; cited 2014 Jan 15]. Available from: http://www.idef.com/pdf/idef0.pdf
10. Dolgui A, Guschinsky N, Levin G. Exact and heuristic algorithms for balancing transfer lines when a set of available spindle heads is given. International Transactions in Operational Research. 2008; 15(3):339-57.

11. Dolgui A, Guschinsky N, Levin G. Enhanced mixed integer programming model for a transfer line design problem. Computers & Industrial Engineering. 2012; 62:570-78.

12. Das K, Baki MF, Li X. Optimization of operation and changeover time for production planning and scheduling in a flexible manufacturing system. Computers & Industrial Engineering. 2009; 56(1):283-93.

13. Osman H, Baki MF. A Linearization and Decomposition Based Approach to Minimize the Non-Productive Time in Transfer Lines. World Academy of Science, Engineering and Technology. 2013:74.

14. Osman H, Baki MF. Balancing transfer lines using Benders decomposition and ant colony optimisation techniques. International Journal of Production Research. 2013: 1-17.

15. Zhang GW, Zhang SC, Xu YS. Research on flexible transfer line schematic design using hierarchical process planning. Journal of Materials Processing Technology. 2002; 129:629-633.

16. Masood, Syed. Line balancing and simulation of an automated production transfer line. Assembly Automation. 2006; 26(1):69-74.

17. Persi P, Ukovich W, Pesenti R, Nicolich M. A hierarchic approach to production planning and scheduling of a flexible manufacturing system. Robotics and Computer-Integrated Manufacturing. 1999;(1)15(5):373-85.

18. Sarin S, Chen C. The machine loading and tool allocation problem in a flexible manufacturing system. International Journal of Production Research. 1987; 25(7):1081-94.

19. Sinriech D, Rubinovitz J, Milo D, Nakbily G. Sequencing, scheduling and tooling single-stage multifunctional machines in a small batch environment. IIE Transactions. 2001; 33(10):897-911.

20. Ecker KH, Gupta JND. Scheduling tasks on a flexible manufacturing machine to minimize tool change delays. European Journal of Operational Research. 2005; 164(3):627-38.

21. Lin C-J, Wang H-P. Optimal operation planning and sequencing: minimization of tool changeovers. The International Journal of Production Research. 1993; 31(2):311-24.

22. McMullen PR, Frazier G. Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations. International Journal of Production Research. 1998; 36(10):2717-41.

23. Pandey V, Tiwari MK, Kumar S. An interactive approach to solve the operation sequencing problem using simulated annealing. The International Journal of Advanced Manufacturing Technology. 2005; 29(11-12):1212-31.

24. Azab A, ElMaraghy H. Sequential process planning: A hybrid optimal macro-level approach. Journal of Manufacturing Systems. 2007; 26(3-4):147-60.

25. Bhale S, Baki MF, Azab A. Grouping and Sequencing of Machining Operations for High Volume Transfer Lines. Proceedings of the 47th CIRP Conference on Manufacturing Systems; 2014 Apr 28-30; Windsor, Canada. Windsor: Canada; 2014. P. 413-18.

26. Suresh G, Sahu S. Stochastic assembly line balancing using simulated annealing. The International Journal of Production Research. 1994; 32.8:1801-10.

27. Fourer R, Gay D, Kernighan BW. The AMPL book. Duxbury Press, Pacific Grove; 2002.

28. Savitch WJ. Problem solving with C++: the object of programming / Walter Savitch. Reading, Mass.: Addison Wesley Longman; 1999.

29. Lajis MA, Karim ANM, Amin AKMN, Hafiz AMK, Turnad LG. Prediction of Tool Life in End Milling of Hardened Steel AISI D2. European Journal of Operations Research. 2008; 21(4):592-602.

30. Arezoo B, Ridgway K, Al-Ahmari AMA. Selection of cutting tools and conditions of machining operations using an expert system. Computers in Industry. 2000; 42:43-58.

APPENDIX A: AMPL PROGRAM FOR MATHEMATICAL MODEL: CASE STUDY 1

Grouping Sub-model

Model file

Format: .mod

```
reset;

option solver cplex;

set R;

set O{R};

set G;

param Or{R} >= 0;

param ORCT{r in R, rp in R} default 0;

param i{R,R} binary default 0;

param e{R,R} binary default 0;

var X {r in R,O[r],G} binary;

var Q {G} binary;

var Z {R,G} binary;

var B {R,G,R,G} binary;

var W1{g in G} = sum {r in R, rp in R} ORCT[r,rp] * B[r,g,rp,g];

var S{g in G} = sum {r in R} Or[r] * Z[r,g];

minimize objective:
```

```
sum{g in G} W1[g];

subject to eq_06 {r in R}:

sum {g in G} Z[r,g] = 1;

subject to eq_07 {g in G}:

sum {r in R} Z[r,g] >= 0;

subject to eq_08 {r in R, o in O[r], g in G}:

X[r,o,g] = Z[r,g];

subject to eq_09 {g in G, r in R, rp in R: rp>r and i[r,rp]=1}:

(Z[r,g] - Z[rp,g]) = 0;

subject to eq_10 {g in G, r in R, rp in R: rp>r and e[r,rp]=1}:

(Z[r,g] + Z[rp,g]) <= 1;

subject to eq_11 {g in G}:

sum {r in R} Z[r,g] <= 10000000 * Q[g];

subject to eq_13 {r in R, g in G, rp in R}:

Z[r,g] = sum{gp in G} B[r,g,rp,gp];

subject to eq_14 {r in R, g in G, rp in R, gp in G: rp>r}:

B[r,g,rp,gp] = B[rp,gp,r,g];
```

Data file

Format: .dat

```
param: R:      Or:=

1        4
```

```
2       4

3       3

4       3

5       2

6       4

7       3

8       3

9       2

10      2

11      4

12      4 ;

set O[1] := 1 2 3 4 ;

set O[2] := 5 6 7 8 ;

set O[3] := 9 10 11 ;

set O[4] := 12 13 14 ;

set O[5] := 15 16 ;

set O[6] := 17 18 19 20 ;

set O[7] := 21 22 23 ;

set O[8] := 24 25 26 ;

set O[9] := 27 28 ;

set O[10] := 29 30 ;
```

```
set O[11] := 31 32 33 34 ;

set O[12] := 35 36 37 38 ;

set G := 1 2 3 4 ;

let i[2,3] := 1;

let i[4,5] := 1;

let e[10,1] := 1;

let ORCT[1,4] :=  1.95;    let ORCT[1,5] :=  1.95;    let ORCT[1,6] :=  2.5;
let ORCT[1,7] :=  2.5;     let ORCT[1,8] :=  4;       let ORCT[1,9] :=  2.5;
let ORCT[1,10] :=  5;      let ORCT[1,11] :=  5;      let ORCT[1,12] :=  5;

let ORCT[2,4] :=  1.95;    let ORCT[2,5] :=  1.95;    let ORCT[2,6] :=  2.5;
let ORCT[2,7] :=  2.5;     let ORCT[2,8] :=  4;       let ORCT[2,9] :=  2.5;
let ORCT[2,10] :=  5;      let ORCT[2,11] :=  5;      let ORCT[2,12] :=  5;

let ORCT[3,4] :=  1.95;    let ORCT[3,5] :=  1.95;    let ORCT[3,6] :=  2.5;
let ORCT[3,7] :=  2.5;     let ORCT[3,8] :=  4;       let ORCT[3,9] :=  2.5;
let ORCT[3,10] :=  5;      let ORCT[3,11] :=  5;      let ORCT[3,12] :=  5;

let ORCT[4,1] :=  1.95;    let ORCT[4,2] :=  1.95;    let ORCT[4,3] :=  1.95;
let ORCT[4,6] :=  1.95;    let ORCT[4,7] :=  1.95;    let ORCT[4,8] :=  2.5;
let ORCT[4,9]:=  1.95;     let ORCT[4,10] :=  4;      let ORCT[4,11] :=  4;
let ORCT[4,12] :=  4;

let ORCT[5,1] :=  1.95;    let ORCT[5,2] :=  1.95;    let ORCT[5,3] :=  1.95;
let ORCT[5,6] :=  1.95;    let ORCT[5,7] :=  1.95;    let ORCT[5,8] :=  2.5;
let ORCT[5,9]:=  1.95;     let ORCT[5,10] :=  4;      let ORCT[5,11] :=  4;
let ORCT[5,12] :=  4;

let ORCT[6,1] :=  2.5;     let ORCT[6,2] :=  2.5;     let ORCT[6,3] :=  2.5;
let ORCT[6,4] :=  1.95;    let ORCT[6,5] :=  1.95;    let ORCT[6,8] :=  1.95;
let ORCT[6,10]:=  2.5;     let ORCT[6,11] :=  2.5;    let ORCT[6,12] :=  2.5;
```

let ORCT[7,1] := 2.5; let ORCT[7,2] := 2.5; let ORCT[7,3] := 2.5;
let ORCT[7,4] := 1.95; let ORCT[7,5] := 1.95; let ORCT[7,8] := 1.95;
let ORCT[7,10]:= 2.5; let ORCT[7,11] := 2.5; let ORCT[7,12] := 2.5;

let ORCT[8,1] := 4; let ORCT[8,2] := 4; let ORCT[8,3] := 4;
let ORCT[8,4] := 2.5; let ORCT[8,5] := 2.5; let ORCT[8,6] := 1.95;
let ORCT[8,7]:= 1.95; let ORCT[8,9] := 1.95; let ORCT[8,10] := 1.95;
let ORCT[8,11] := 1.95; let ORCT[8,12] := 1.95;

let ORCT[9,1] := 2.5; let ORCT[9,2] := 2.5; let ORCT[9,3] := 2.5;
let ORCT[9,4]:= 1.95; let ORCT[9,5] := 1.95; let ORCT[9,8] := 1.95;
let ORCT[9,10] := 2.5; let ORCT[9,11] := 2.5; let ORCT[9,12] := 2.5;

let ORCT[10,1] := 5; let ORCT[10,2] := 5; let ORCT[10,3] := 5;
let ORCT [10,4]:= 4; let ORCT[10,5] := 4; let ORCT[10,6] := 2.5;
let ORCT[10,7] := 2.5; let ORCT[10,8] := 1.95; let ORCT[10,9] := 2.5;

let ORCT[11,1] := 5; let ORCT[11,2] := 5; let ORCT[11,3] := 5;
let ORCT [11,4]:= 4; let ORCT[11,5] := 4; let ORCT[11,6] := 2.5;
let ORCT[11,7] := 2.5; let ORCT[11,8] := 1.95; let ORCT[11,9] := 2.5;

let ORCT[12,1] := 5; let ORCT[12,2] := 5; let ORCT[12,3] := 5;
let ORCT [12,4]:= 4; let ORCT[12,5] := 4; let ORCT[12,6] := 2.5;
let ORCT[12,7] := 2.5; let ORCT[12,8] := 1.95; let ORCT[12,9] := 2.5;

Sequencing Sub-model

Sample model and data files for group 1 are included below. The coding structure for other groups is similar except for the number of sequence positions, which differs for each group.

Model file

Format: .mod

```
reset;

option solver cplex;

set R;

set O{R};

set O1{R};

set O2{R};

set G;

set L;

set S{G};

set SP{G};

set S1{G};

set SP1{G};

set S2{G};

set SP2{G};

set S3{G};

set SP3{G};

set S4{G};

set SP4{G};

set S5{G};

set SP5{G};

set S6{G};
```

```
set SP6{G};

set S7{G};

set SP7{G};

set S8{G};

set SP8{G};

set S9{G};

set SP9{G};

set S10{G};

set SP10{G};

param Or{R} >= 0;

param Orf{R} >= 0;

param Orl{R} >= 0;

param D{R} >= 0;

param T{L} default 100;

param H{L} >= 0;

param E >= 0;

param A default 20;

param ORCT{r in R, rp in R} default 0;

param TLCT{r in R, o in O[r], rp in R, op in O[rp]} default 0;

param TO{r in R, o in O[r]} default 0;

param RT{r in R, o in O[r]} default 0;
```

```
param P{r in R, o in O[r], l in L} default 0;

param Z {R,G} default 0;

param NM{G} >= 0;

param i{R,R} binary default 0;

param t default 5;

var X {r in R, o in O[r], g in G, s in S[g]} binary;

var Y {L,G} binary;

var C {g in G, r in R, o in O[r], s in S[g], rp in R, op in O[rp], sp in S[g]} binary;

var N {G} integer >= 0;

var W1{G} = sum {r in R, rp in R, g in G, o in O[r], op in O[rp], s in SP[g]:rp!=r}
ORCT[r,rp] * C[g,r,o,s,rp,op,s+1] * Z[r,g] * Z[rp,g];

var W2{G} = sum{g in G, r in R, o in O[r], s in SP[g], op in O[r]} TLCT[r,o,r,op] *
C[g,r,o,s,r,op,s+1] * Z[r,g] + sum{g in G, r in R, o in O[r], rp in R, op in O[rp], s in
SP[g]:r!=rp} TLCT[r,o,rp,op] * C[g,r,o,s,rp,op,s+1] * Z[r,g] * Z[rp,g];

var W3{G} = sum {r in R, o in O[r], g in G, s in S[g]} D[r] * RT[r,o] * X[r,o,g,s];

var W4{G} = sum {r in R, o in O[r], g in G, s in S[g]} D[r] * TO[r,o] * X[r,o,g,s];

minimize objective1_non_productive_time:

sum {g in G} W1[g] + sum {g in G} W2[g] + sum{g in G} (N[g]-1) * t;

subject to eq_15 {g in G}:

N[g] <= NM[g];

subject to eq_16_001{r in R, o in O1[r], g in G, s in S1[g]}:

sum{sp in SP1[g]} (X[r,o,g,sp]) >= sum{sp in SP1[g]} (X[r,o+1,g,sp]);
```

```
subject to eq_16_002{r in R, o in O1[r], g in G, s in S2[g]}:

sum{sp in SP2[g]} (X[r,o,g,sp]) >= sum{sp in SP2[g]} (X[r,o+1,g,sp]);

subject to eq_16_003{r in R, o in O1[r], g in G, s in S3[g]}:

sum{sp in SP3[g]} (X[r,o,g,sp]) >= sum{sp in SP3[g]} (X[r,o+1,g,sp]);

subject to eq_16_004{r in R, o in O1[r], g in G, s in S4[g]}:

sum{sp in SP4[g]} (X[r,o,g,sp]) >= sum{sp in SP4[g]} (X[r,o+1,g,sp]);

subject to eq_16_005{r in R, o in O1[r], g in G, s in S5[g]}:

sum{sp in SP5[g]} (X[r,o,g,sp]) >= sum{sp in SP5[g]} (X[r,o+1,g,sp]);

subject to eq_16_006{r in R, o in O1[r], g in G, s in S6[g]}:

sum{sp in SP6[g]} (X[r,o,g,sp]) >= sum{sp in SP6[g]} (X[r,o+1,g,sp]);

subject to eq_16_007{r in R, o in O1[r], g in G, s in S7[g]}:

sum{sp in SP7[g]} (X[r,o,g,sp]) >= sum{sp in SP7[g]} (X[r,o+1,g,sp]);

subject to eq_16_008{r in R, o in O1[r], g in G, s in S8[g]}:

sum{sp in SP8[g]} (X[r,o,g,sp]) >= sum{sp in SP8[g]} (X[r,o+1,g,sp]);

subject to eq_16_009{r in R, o in O1[r], g in G, s in S9[g]}:

sum{sp in SP9[g]} (X[r,o,g,sp]) >= sum{sp in SP9[g]} (X[r,o+1,g,sp]);

subject to eq_16_010{r in R, o in O1[r], g in G, s in S10[g]}:

sum{sp in SP10[g]} (X[r,o,g,sp]) >= sum{sp in SP10[g]} (X[r,o+1,g,sp]);

subject to eq_17_001{r in R, rp in R, g in G, s in S1[g]:i[r,rp]=1}:

sum{sp in SP1[g]} (X[r,Orl[r],g,sp]) >= sum{sp in SP1[g]} (X[rp,Orf[rp],g,sp]);

subject to eq_17_002{r in R, rp in R, g in G, s in S2[g]:i[r,rp]=1}:
```

```
sum{sp in SP2[g]} (X[r,Orl[r],g,sp]) >= sum{sp in SP2[g]} (X[rp,Orf[rp],g,sp]);

subject to eq_17_003{r in R, rp in R, g in G, s in S3[g]:i[r,rp]=1}:

sum{sp in SP3[g]} (X[r,Orl[r],g,sp]) >= sum{sp in SP3[g]} (X[rp,Orf[rp],g,sp]);

subject to eq_17_004{r in R, rp in R, g in G, s in S4[g]:i[r,rp]=1}:

sum{sp in SP4[g]} (X[r,Orl[r],g,sp]) >= sum{sp in SP4[g]} (X[rp,Orf[rp],g,sp]);

subject to eq_17_005{r in R, rp in R, g in G, s in S5[g]:i[r,rp]=1}:

sum{sp in SP5[g]} (X[r,Orl[r],g,sp]) >= sum{sp in SP5[g]} (X[rp,Orf[rp],g,sp]);

subject to eq_17_006{r in R, rp in R, g in G, s in S6[g]:i[r,rp]=1}:

sum{sp in SP6[g]} (X[r,Orl[r],g,sp]) >= sum{sp in SP6[g]} (X[rp,Orf[rp],g,sp]);

subject to eq_17_007{r in R, rp in R, g in G, s in S7[g]:i[r,rp]=1}:

sum{sp in SP7[g]} (X[r,Orl[r],g,sp]) >= sum{sp in SP7[g]} (X[rp,Orf[rp],g,sp]);

subject to eq_17_008{r in R, rp in R, g in G, s in S8[g]:i[r,rp]=1}:

sum{sp in SP8[g]} (X[r,Orl[r],g,sp]) >= sum{sp in SP8[g]} (X[rp,Orf[rp],g,sp]);

subject to eq_17_009{r in R, rp in R, g in G, s in S9[g]:i[r,rp]=1}:

sum{sp in SP9[g]} (X[r,Orl[r],g,sp]) >= sum{sp in SP9[g]} (X[rp,Orf[rp],g,sp]);

subject to eq_17_010{r in R, rp in R, g in G, s in S10[g]:i[r,rp]=1}:

sum{sp in SP10[g]} (X[r,Orl[r],g,sp]) >= sum{sp in SP10[g]} (X[rp,Orf[rp],g,sp]);

subject to eq_18 {r in R, o in O[r], g in G}:

sum {s in S[g]} X[r,o,g,s] = Z[r,g];

subject to eq_19 {g in G, s in S[g]}:

sum {r in R, o in O[r]} X[r,o,g,s] = 1;
```

subject to eq_20 {g in G, s in S[g], l in L}:

sum {r in R, o in O[r]} P[r,o,l] * D[r] * TO[r,o] * X[r,o,g,s] <= T[l] * Y[l,g];

subject to eq_21 {g in G}:

sum {l in L} H[l] * Y[l,g] <= A * N[g];

subject to eq_22 :

sum{g in G} W1[g] + sum {g in G} W2[g] + sum{g in G} W3[g] + sum{g in G} W4[g] <= sum{g in G} E * N[g];

subject to eq_23 {g in G, s in S[g], rp in R, op in O[rp], sp in S[g]:s != sp}:

X[rp,op,g,sp] = sum{r in R, o in O[r]} C[g,r,o,s,rp,op,sp];

subject to eq_24 {g in G, s in S[g], sp in S[g], r in R, o in O[r], rp in R, op in O[rp]:s != sp}:

C[g,r,o,s,rp,op,sp] = C[g,rp,op,sp,r,o,s];

Data file

Format: .dat

param: R: Or:=

1 4

2 4

3 3

4 3

5 2

6 4

7 3

8 3

9 2

10 2

11 4

12 4 ;

param: L: H :=

1 4

2 2

3 2

4 2

5 2

6 1

7 1

8 1

9 1

10 1

11 3

12 3 ;

set O[1] := 1 2 3 4 ;

set O[2] := 5 6 7 8 ;

set O[3] := 9 10 11 ;

```
set O[4] := 12 13 14 ;

set O[5] := 15 16 ;

set O[6] := 17 18 19 20 ;

set O[7] := 21 22 23 ;

set O[8] := 24 25 26 ;

set O[9] := 27 28 ;

set O[10] := 29 30 ;

set O[11] := 31 32 33 34 ;

set O[12] := 35 36 37 38 ;

set O1[1] := 1 2 3 ;

set O1[2] := 5 6 7 ;

set O1[3] := 9 10 ;

set O1[4] := 12 13 ;

set O1[5] := 15 ;

set O1[6] := 17 18 19 ;

set O1[7] := 21 22 ;

set O1[8] := 24 25 ;

set O1[9] := 27 ;

set O1[10] := 29 ;

set O1[11] := 31 32 33 ;

set O1[12] := 35 36 37 ;
```

```
set G := 1 ;

set SP[1] := 1 2 3 4 5 6 7 ;

set S[1] := 1 2 3 4 5 6 7 8 ;

set S1[1] := 1 ;

set S2[1] := 2 ;

set S3[1] := 3 ;

set S4[1] := 4 ;

set S5[1] := 5 ;

set S6[1] := 6 ;

set S7[1] := 7 ;

set S8[1] := 8 ;

set SP1[1] := 1 ;

set SP2[1] := 1 2 ;

set SP3[1] := 1 2 3 ;

set SP4[1] := 1 2 3 4 ;

set SP5[1] := 1 2 3 4 5 ;

set SP6[1] := 1 2 3 4 5 6 ;

set SP7[1] := 1 2 3 4 5 6 7 ;

set SP8[1] := 1 2 3 4 5 6 7 8 ;

let Orf[1] := 1;

let Orf[2] := 5;
```

```
let Orf[3] := 9;

let Orf[4] := 12;

let Orf[5] := 15;

let Orf[6] := 17;

let Orf[7] := 21;

let Orf[8] := 24;

let Orf[9] := 27;

let Orf[10] := 29;

let Orf[11] := 31;

let Orf[12] := 35;

let Orl[1] := 4;

let Orl[2] := 8;

let Orl[3] := 11;

let Orl[4] := 14;

let Orl[5] := 16;

let Orl[6] := 20;

let Orl[7] := 23;

let Orl[8] := 26;

let Orl[9] := 28;

let Orl[10] := 30;

let Orl[11] := 34;
```

```
let Orl[12] := 38;

let Z[4,1] := 1 ;

let Z[5,1] := 1 ;

let Z[8,1] := 1 ;

let i[2,3] := 1;

let i[4,5] := 1;

let E := 400;

let NM[1] := 5 ;

let TO[1,1] := 2.88 ;

let TO[1,2] := 1.25 ;

let TO[1,3] := 1.71 ;

let TO[1,4] := 2.93 ;

let TO[2,5] := 1.54 ;

let TO[2,6] := 1.57 ;

let TO[2,7] := 2.62 ;

let TO[2,8] := 1.7 ;

let TO[3,9] := 2.75 ;

let TO[3,10] := 1.79 ;

let TO[3,11] := 2.89 ;

let TO[4,12] := 1.95 ;

let TO[4,13] := 1.2 ;
```

```
let TO[4,14] := 1.65 ;

let TO[5,15] := 2.68 ;

let TO[5,16] := 2.3 ;

let TO[6,17] := 1.97 ;

let TO[6,18] := 3.37 ;

let TO[6,19] := 1.36 ;

let TO[6,20] := 2.74 ;

let TO[7,21] := 1.35 ;

let TO[7,22] := 1.22 ;

let TO[7,23] := 3.09 ;

let TO[8,24] := 1.2 ;

let TO[8,25] := 1.62 ;

let TO[8,26] := 2.06 ;

let TO[9,27] := 1.9 ;

let TO[9,28] := 3.3 ;

let TO[10,29] := 2.78 ;

let TO[10,30] := 1.31 ;

let TO[11,31] := 3.43 ;

let TO[11,32] := 3.42 ;

let TO[11,33] := 3.39 ;

let TO[11,34] := 2.5 ;
```

let TO[12,35] := 2.64 ;

let TO[12,36] := 2.26 ;

let TO[12,37] := 2.45 ;

let TO[12,38] := 2.32 ;

let RT[1,1] := 1.73 ;

let RT[1,2] := 0.75 ;

let RT[1,3] := 1.03 ;

let RT[1,4] := 1.8 ;

let RT[2,5] := 0.92 ;

let RT[2,6] := 0.94 ;

let RT[2,7] := 1.57 ;

let RT[2,8] := 1 ;

let RT[3,9] := 1.65 ;

let RT[3,10] := 1.07 ;

let RT[3,11] := 1.73 ;

let RT[4,12] := 1.17 ;

let RT[4,13] := 0.75 ;

let RT[4,14] := 0.99 ;

let RT[5,15] := 1.61 ;

let RT[5,16] := 1.38 ;

let RT[6,17] := 1.18 ;

```
let RT[6,18] := 2.02 ;

let RT[6,19] := 0.82 ;

let RT[6,20] := 1.4 ;

let RT[7,21] := 0.81 ;

let RT[7,22] := 0.73 ;

let RT[7,23] := 1.85 ;

let RT[8,24] := 0.72 ;

let RT[8,25] := 0.97 ;

let RT[8,26] := 1.24 ;

let RT[9,27] := 1.14 ;

let RT[9,28] := 1.98 ;

let RT[10,29] := 1.67 ;

let RT[10,30] := 0.79 ;

let RT[11,31] := 2.05 ;

let RT[11,32] := 2.05 ;

let RT[11,33] := 2.03 ;

let RT[11,34] := 1.5 ;

let RT[12,35] := 1.58 ;

let RT[12,36] := 1.36 ;

let RT[12,37] := 1.47 ;

let RT[12,38] := 1.4 ;
```

```
let P[1,1,1] := 1 ;

let P[1,2,2] := 1 ;

let P[1,3,3] := 1 ;

let P[1,4,4] := 1 ;

let P[2,5,1] := 1 ;

let P[2,6,2] := 1 ;

let P[2,7,4] := 1 ;

let P[2,8,5] := 1 ;

let P[3,9,1] := 1 ;

let P[3,10,4] := 1 ;

let P[3,11,5] := 1 ;

let P[4,12,6] := 1 ;

let P[4,13,7] := 1 ;

let P[4,14,8] := 1 ;

let P[5,15,9] := 1 ;

let P[5,16,10] := 1 ;

let P[6,17,9] := 1 ;

let P[6,18,10] := 1 ;

let P[6,19,11] := 1 ;

let P[6,20,12] := 1 ;

let P[7,21,10] := 1 ;
```

```
let P[7,22,11] := 1 ;

let P[7,23,12] := 1 ;

let P[8,24,3] := 1 ;

let P[8,25,4] := 1 ;

let P[8,26,5] := 1 ;

let P[9,27,3] := 1 ;

let P[9,28,4] := 1 ;

let P[10,29,1] := 1 ;

let P[10,30,2] := 1 ;

let P[11,31,11] := 1 ;

let P[11,32,12] := 1 ;

let P[11,33,5] := 1 ;

let P[11,34,6] := 1 ;

let P[12,35,8] := 1 ;

let P[12,36,9] := 1 ;

let P[12,37,10] := 1 ;

let P[12,38,11] := 1 ;

let D[1] := 16 ;

let D[2] := 4 ;

let D[3] := 8 ;

let D[4] := 8 ;
```

```
let D[5] := 4 ;

let D[6] := 4 ;

let D[7] := 4 ;

let D[8] := 20 ;

let D[9] := 20 ;

let D[10] := 8 ;

let D[11] := 8 ;

let D[12] := 4 ;

let ORCT[1,4] :=  1.95;    let ORCT[1,5] :=  1.95;    let ORCT[1,6] :=  2.5;
let ORCT[1,7] :=  2.5;     let ORCT[1,8] :=  4;       let ORCT[1,9] :=  2.5;
let ORCT[1,10] :=  5;      let ORCT[1,11] :=  5;      let ORCT[1,12] :=  5;

let ORCT[2,4] :=  1.95;    let ORCT[2,5] :=  1.95;    let ORCT[2,6] :=  2.5;
let ORCT[2,7] :=  2.5;     let ORCT[2,8] :=  4;       let ORCT[2,9] :=  2.5;
let ORCT[2,10] :=  5;      let ORCT[2,11] :=  5;      let ORCT[2,12] :=  5;

let ORCT[3,4] :=  1.95;    let ORCT[3,5] :=  1.95;    let ORCT[3,6] :=  2.5;
let ORCT[3,7] :=  2.5;     let ORCT[3,8] :=  4;       let ORCT[3,9] :=  2.5;
let ORCT[3,10] :=  5;      let ORCT[3,11] :=  5;      let ORCT[3,12] :=  5;

let ORCT[4,1] :=  1.95;    let ORCT[4,2] :=  1.95;    let ORCT[4,3] :=  1.95;
let ORCT[4,6] :=  1.95;    let ORCT[4,7] :=  1.95;    let ORCT[4,8] :=  2.5;
let ORCT[4,9]:=  1.95;     let ORCT[4,10] :=  4;      let ORCT[4,11] :=  4;
let ORCT[4,12] :=  4;

let ORCT[5,1] :=  1.95;    let ORCT[5,2] :=  1.95;    let ORCT[5,3] :=  1.95;
let ORCT[5,6] :=  1.95;    let ORCT[5,7] :=  1.95;    let ORCT[5,8] :=  2.5;
let ORCT[5,9]:=  1.95;     let ORCT[5,10] :=  4;      let ORCT[5,11] :=  4;
let ORCT[5,12] :=  4;
```

```
let ORCT[6,1] := 2.5;      let ORCT[6,2] := 2.5;      let ORCT[6,3] := 2.5;
let ORCT[6,4] := 1.95;     let ORCT[6,5] := 1.95;     let ORCT[6,8] := 1.95;
let ORCT[6,10]:= 2.5;      let ORCT[6,11] := 2.5;     let ORCT[6,12] := 2.5;

let ORCT[7,1] := 2.5;      let ORCT[7,2] := 2.5;      let ORCT[7,3] := 2.5;
let ORCT[7,4] := 1.95;     let ORCT[7,5] := 1.95;     let ORCT[7,8] := 1.95;
let ORCT[7,10]:= 2.5;      let ORCT[7,11] := 2.5;     let ORCT[7,12] := 2.5;

let ORCT[8,1] := 4;        let ORCT[8,2] := 4;        let ORCT[8,3] := 4;
let ORCT[8,4] := 2.5;      let ORCT[8,5] := 2.5;      let ORCT[8,6] := 1.95;
let ORCT[8,7]:= 1.95;      let ORCT[8,9] := 1.95;     let ORCT[8,10] := 1.95;
let ORCT[8,11] := 1.95;    let ORCT[8,12] := 1.95;

let ORCT[9,1] := 2.5;      let ORCT[9,2] := 2.5;      let ORCT[9,3] := 2.5;
let ORCT[9,4]:= 1.95;      let ORCT[9,5] := 1.95;     let ORCT[9,8] := 1.95;
let ORCT[9,10] := 2.5;     let ORCT[9,11] := 2.5;     let ORCT[9,12] := 2.5;

let ORCT[10,1] := 5;       let ORCT[10,2] := 5;       let ORCT[10,3] := 5;
let ORCT [10,4]:= 4;       let ORCT[10,5] := 4;       let ORCT[10,6] := 2.5;
let ORCT[10,7] := 2.5;     let ORCT[10,8] := 1.95;    let ORCT[10,9] := 2.5;

let ORCT[11,1] := 5;       let ORCT[11,2] := 5;       let ORCT[11,3] := 5;
let ORCT [11,4]:= 4;       let ORCT[11,5] := 4;       let ORCT[11,6] := 2.5;
let ORCT[11,7] := 2.5;     let ORCT[11,8] := 1.95;    let ORCT[11,9] := 2.5;

let ORCT[12,1] := 5;       let ORCT[12,2] := 5;       let ORCT[12,3] := 5;
let ORCT [12,4]:= 4;       let ORCT[12,5] := 4;       let ORCT[12,6] := 2.5;
let ORCT[12,7] := 2.5;     let ORCT[12,8] := 1.95;    let ORCT[12,9] := 2.5;

let TLCT[1,1,1,1] := 0;    let TLCT[1,1,1,2] := 10;   let TLCT[1,1,1,3] := 6;
let TLCT[1,1,1,4] := 1;    let TLCT[1,1,2,5] := 0;    let TLCT[1,1,2,6] := 9;
let TLCT[1,1,2,7] := 1;    let TLCT[1,1,2,8] := 8;    let TLCT[1,1,3,9] := 0;
let TLCT[1,1,3,10] := 4;   let TLCT[1,1,3,11] := 4;   let TLCT[1,1,4,12] := 5;
let TLCT[1,1,4,13] := 2;   let TLCT[1,1,4,14] := 8;   let TLCT[1,1,5,15] := 9;
let TLCT[1,1,5,16] := 6;   let TLCT[1,1,6,17] := 3;   let TLCT[1,1,6,18] := 5;
```

```
let TLCT[1,1,6,19] := 2;   let TLCT[1,1,6,20] := 1;   let TLCT[1,1,7,21] := 5;
let TLCT[1,1,7,22] := 7;   let TLCT[1,1,7,23] := 8;   let TLCT[1,1,8,24] := 7;
let TLCT[1,1,8,25] := 6;   let TLCT[1,1,8,26] := 10;  let TLCT[1,1,9,27] := 2;
let TLCT[1,1,9,28] := 9;   let TLCT[1,1,10,29] := 0;  let TLCT[1,1,10,30] := 6;
let TLCT[1,1,11,31] := 2;  let TLCT[1,1,11,32] := 2;  let TLCT[1,1,11,33] := 10;
let TLCT[1,1,11,34] := 3;  let TLCT[1,1,12,35] := 4;  let TLCT[1,1,12,36] := 10;
let TLCT[1,1,12,37] := 10; let TLCT[1,1,12,38] := 7;

let TLCT[1,2,1,1] := 8;    let TLCT[1,2,1,2] := 0;    let TLCT[1,2,1,3] := 2;
let TLCT[1,2,1,4] := 1;    let TLCT[1,2,2,5] := 6;    let TLCT[1,2,2,6] := 0;
let TLCT[1,2,2,7] := 4;    let TLCT[1,2,2,8] := 1;    let TLCT[1,2,3,9] := 3;
let TLCT[1,2,3,10] := 9;   let TLCT[1,2,3,11] := 2;   let TLCT[1,2,4,12] := 2;
let TLCT[1,2,4,13] := 4;   let TLCT[1,2,4,14] := 8;   let TLCT[1,2,5,15] := 3;
let TLCT[1,2,5,16] := 6;   let TLCT[1,2,6,17] := 4;   let TLCT[1,2,6,18] := 6;
let TLCT[1,2,6,19] := 2;   let TLCT[1,2,6,20] := 4;   let TLCT[1,2,7,21] := 6;
let TLCT[1,2,7,22] := 6;   let TLCT[1,2,7,23] := 1;   let TLCT[1,2,8,24] := 3;
let TLCT[1,2,8,25] := 5;   let TLCT[1,2,8,26] := 8;   let TLCT[1,2,9,27] := 9;
let TLCT[1,2,9,28] := 6;   let TLCT[1,2,10,29] := 2;  let TLCT[1,2,10,30] := 0;
let TLCT[1,2,11,31] := 8;  let TLCT[1,2,11,32] := 2;  let TLCT[1,2,11,33] := 7;
let TLCT[1,2,11,34] := 2;  let TLCT[1,2,12,35] := 5;  let TLCT[1,2,12,36] := 5;
let TLCT[1,2,12,37] := 9;  let TLCT[1,2,12,38] := 5;

let TLCT[1,3,1,1] := 6;    let TLCT[1,3,1,2] := 9;    let TLCT[1,3,1,3] := 0;
let TLCT[1,3,1,4] := 10;   let TLCT[1,3,2,5] := 2;    let TLCT[1,3,2,6] := 1;
let TLCT[1,3,2,7] := 6;    let TLCT[1,3,2,8] := 3;    let TLCT[1,3,3,9] := 1;
let TLCT[1,3,3,10] := 9;   let TLCT[1,3,3,11] := 3;   let TLCT[1,3,4,12] := 6;
let TLCT[1,3,4,13] := 2;   let TLCT[1,3,4,14] := 6;   let TLCT[1,3,5,15] := 6;
let TLCT[1,3,5,16] := 7;   let TLCT[1,3,6,17] := 10;  let TLCT[1,3,6,18] := 5;
let TLCT[1,3,6,19] := 7;   let TLCT[1,3,6,20] := 1;   let TLCT[1,3,7,21] := 9;
let TLCT[1,3,7,22] := 9;   let TLCT[1,3,7,23] := 7;   let TLCT[1,3,8,24] := 0;
let TLCT[1,3,8,25] := 5;   let TLCT[1,3,8,26] := 2;   let TLCT[1,3,9,27] := 0;
let TLCT[1,3,9,28] := 7;   let TLCT[1,3,10,29] := 10; let TLCT[1,3,10,30] := 3;
```

```
let TLCT[1,3,11,31] := 6;  let TLCT[1,3,11,32] := 5;  let TLCT[1,3,11,33] := 4;
let TLCT[1,3,11,34] := 1;  let TLCT[1,3,12,35] := 7;  let TLCT[1,3,12,36] := 6;
let TLCT[1,3,12,37] := 5;  let TLCT[1,3,12,38] := 7;

let TLCT[1,4,1,1] := 9;    let TLCT[1,4,1,2] := 8;    let TLCT[1,4,1,3] := 7;
let TLCT[1,4,1,4] := 0;    let TLCT[1,4,2,5] := 10;   let TLCT[1,4,2,6] := 4;
let TLCT[1,4,2,7] := 0;    let TLCT[1,4,2,8] := 9;    let TLCT[1,4,3,9] := 8;
let TLCT[1,4,3,10] := 0;   let TLCT[1,4,3,11] := 3;   let TLCT[1,4,4,12] := 4;
let TLCT[1,4,4,13] := 6;   let TLCT[1,4,4,14] := 6;   let TLCT[1,4,5,15] := 9;
let TLCT[1,4,5,16] := 6;   let TLCT[1,4,6,17] := 1;   let TLCT[1,4,6,18] := 1;
let TLCT[1,4,6,19] := 4;   let TLCT[1,4,6,20] := 1;   let TLCT[1,4,7,21] := 10;
let TLCT[1,4,7,22] := 4;   let TLCT[1,4,7,23] := 9;   let TLCT[1,4,8,24] := 7;
let TLCT[1,4,8,25] := 0;   let TLCT[1,4,8,26] := 4;   let TLCT[1,4,9,27] := 2;
let TLCT[1,4,9,28] := 0;   let TLCT[1,4,10,29] := 10; let TLCT[1,4,10,30] := 8;
let TLCT[1,4,11,31] := 8;  let TLCT[1,4,11,32] := 1;  let TLCT[1,4,11,33] := 6;
let TLCT[1,4,11,34] := 8;  let TLCT[1,4,12,35] := 7;  let TLCT[1,4,12,36] := 9;
let TLCT[1,4,12,37] := 4;  let TLCT[1,4,12,38] := 6;

let TLCT[2,5,1,1] := 0;    let TLCT[2,5,1,2] := 1;    let TLCT[2,5,1,3] := 6;
let TLCT[2,5,1,4] := 1;    let TLCT[2,5,2,5] := 0;    let TLCT[2,5,2,6] := 9;
let TLCT[2,5,2,7] := 8;    let TLCT[2,5,2,8] := 8;    let TLCT[2,5,3,9] := 0;
let TLCT[2,5,3,10] := 1;   let TLCT[2,5,3,11] := 5;   let TLCT[2,5,4,12] := 10;
let TLCT[2,5,4,13] := 10;  let TLCT[2,5,4,14] := 2;   let TLCT[2,5,5,15] := 2;
let TLCT[2,5,5,16] := 4;   let TLCT[2,5,6,17] := 8;   let TLCT[2,5,6,18] := 6;
let TLCT[2,5,6,19] := 3;   let TLCT[2,5,6,20] := 2;   let TLCT[2,5,7,21] := 2;
let TLCT[2,5,7,22] := 9;   let TLCT[2,5,7,23] := 10;  let TLCT[2,5,8,24] := 5;
let TLCT[2,5,8,25] := 8;   let TLCT[2,5,8,26] := 3;   let TLCT[2,5,9,27] := 5;
let TLCT[2,5,9,28] := 2;   let TLCT[2,5,10,29] := 0;  let TLCT[2,5,10,30] := 2;
let TLCT[2,5,11,31] := 2;  let TLCT[2,5,11,32] := 7;  let TLCT[2,5,11,33] := 1;
let TLCT[2,5,11,34] := 8;  let TLCT[2,5,12,35] := 5;  let TLCT[2,5,12,36] := 10;
let TLCT[2,5,12,37] := 4;  let TLCT[2,5,12,38] := 4;
```

```
let TLCT[2,6,1,1] :=  9;      let TLCT[2,6,1,2] := 0;       let TLCT[2,6,1,3] :=  8;
let TLCT[2,6,1,4] :=  1;      let TLCT[2,6,2,5] :=  5;      let TLCT[2,6,2,6] :=  0;
let TLCT[2,6,2,7] :=  5;      let TLCT[2,6,2,8] :=  1;      let TLCT[2,6,3,9] :=  2;
let TLCT[2,6,3,10] :=  4;     let TLCT[2,6,3,11] :=  7;     let TLCT[2,6,4,12] :=  1;
let TLCT[2,6,4,13] :=  1;     let TLCT[2,6,4,14] :=  10;    let TLCT[2,6,5,15] :=  10;
let TLCT[2,6,5,16] :=  8;     let TLCT[2,6,6,17] :=  5;     let TLCT[2,6,6,18] :=  7;
let TLCT[2,6,6,19] :=  7;     let TLCT[2,6,6,20] :=  6;     let TLCT[2,6,7,21] :=  8;
let TLCT[2,6,7,22] :=  6;     let TLCT[2,6,7,23] :=  10;    let TLCT[2,6,8,24] :=  1;
let TLCT[2,6,8,25] :=  1;     let TLCT[2,6,8,26] :=  4;     let TLCT[2,6,9,27] :=  3;
let TLCT[2,6,9,28] :=  10;    let TLCT[2,6,10,29] :=  4;    let TLCT[2,6,10,30] :=  0;
let TLCT[2,6,11,31] :=  7;    let TLCT[2,6,11,32] :=  3;    let TLCT[2,6,11,33] :=  3;
let TLCT[2,6,11,34] :=  7;    let TLCT[2,6,12,35] :=  8;    let TLCT[2,6,12,36] :=  7;
let TLCT[2,6,12,37] :=  10;   let TLCT[2,6,12,38] :=  8;

let TLCT[2,7,1,1] :=  4;      let TLCT[2,7,1,2] := 1;       let TLCT[2,7,1,3] :=  5;
let TLCT[2,7,1,4] :=  0;      let TLCT[2,7,2,5] :=  8;      let TLCT[2,7,2,6] :=  5;
let TLCT[2,7,2,7] :=  0;      let TLCT[2,7,2,8] :=  7;      let TLCT[2,7,3,9] :=  8;
let TLCT[2,7,3,10] :=  0;     let TLCT[2,7,3,11] :=  8;     let TLCT[2,7,4,12] :=  5;
let TLCT[2,7,4,13] :=  6;     let TLCT[2,7,4,14] :=  7;     let TLCT[2,7,5,15] :=  1;
let TLCT[2,7,5,16] :=  1;     let TLCT[2,7,6,17] :=  4;     let TLCT[2,7,6,18] :=  8;
let TLCT[2,7,6,19] :=  10;    let TLCT[2,7,6,20] :=  10;    let TLCT[2,7,7,21] :=  7;
let TLCT[2,7,7,22] :=  7;     let TLCT[2,7,7,23] :=  2;     let TLCT[2,7,8,24] :=  7;
let TLCT[2,7,8,25] :=  0;     let TLCT[2,7,8,26] :=  6;     let TLCT[2,7,9,27] :=  1;
let TLCT[2,7,9,28] :=  0;     let TLCT[2,7,10,29] :=  5;    let TLCT[2,7,10,30] :=  6;
let TLCT[2,7,11,31] :=  8;    let TLCT[2,7,11,32] :=  6;    let TLCT[2,7,11,33] :=  5;
let TLCT[2,7,11,34] :=  3;    let TLCT[2,7,12,35] :=  7;    let TLCT[2,7,12,36] :=  9;
let TLCT[2,7,12,37] :=  9;    let TLCT[2,7,12,38] :=  10;

let TLCT[2,8,1,1] :=  4;      let TLCT[2,8,1,2] := 1;       let TLCT[2,8,1,3] :=  10;
let TLCT[2,8,1,4] :=  5;      let TLCT[2,8,2,5] :=  1;      let TLCT[2,8,2,6] :=  4;
let TLCT[2,8,2,7] :=  2;      let TLCT[2,8,2,8] :=  0;      let TLCT[2,8,3,9] :=  7;
let TLCT[2,8,3,10] :=  4;     let TLCT[2,8,3,11] :=  0;     let TLCT[2,8,4,12] :=  9;
```

```
let TLCT[2,8,4,13] := 2;   let TLCT[2,8,4,14] := 5;   let TLCT[2,8,5,15] := 9;
let TLCT[2,8,5,16] := 8;   let TLCT[2,8,6,17] := 6;   let TLCT[2,8,6,18] := 8;
let TLCT[2,8,6,19] := 9;   let TLCT[2,8,6,20] := 7;   let TLCT[2,8,7,21] := 5;
let TLCT[2,8,7,22] := 10;  let TLCT[2,8,7,23] := 0;   let TLCT[2,8,8,24] := 5;
let TLCT[2,8,8,25] := 2;   let TLCT[2,8,8,26] := 0;   let TLCT[2,8,9,27] := 10;
let TLCT[2,8,9,28] := 3;   let TLCT[2,8,10,29] := 7;  let TLCT[2,8,10,30] := 10;
let TLCT[2,8,11,31] := 4;  let TLCT[2,8,11,32] := 1;  let TLCT[2,8,11,33] := 0;
let TLCT[2,8,11,34] := 6;  let TLCT[2,8,12,35] := 1;  let TLCT[2,8,12,36] := 7;
let TLCT[2,8,12,37] := 2;  let TLCT[2,8,12,38] := 1;

let TLCT[3,9,1,1] := 0;    let TLCT[3,9,1,2] := 2;    let TLCT[3,9,1,3] := 8;
let TLCT[3,9,1,4] := 4;    let TLCT[3,9,2,5] := 0;    let TLCT[3,9,2,6] := 2;
let TLCT[3,9,2,7] := 7;    let TLCT[3,9,2,8] := 3;    let TLCT[3,9,3,9] := 0;
let TLCT[3,9,3,10] := 3;   let TLCT[3,9,3,11] := 5;   let TLCT[3,9,4,12] := 2;
let TLCT[3,9,4,13] := 9;   let TLCT[3,9,4,14] := 10;  let TLCT[3,9,5,15] := 7;
let TLCT[3,9,5,16] := 7;   let TLCT[3,9,6,17] := 5;   let TLCT[3,9,6,18] := 4;
let TLCT[3,9,6,19] := 4;   let TLCT[3,9,6,20] := 7;   let TLCT[3,9,7,21] := 6;
let TLCT[3,9,7,22] := 7;   let TLCT[3,9,7,23] := 4;   let TLCT[3,9,8,24] := 8;
let TLCT[3,9,8,25] := 9;   let TLCT[3,9,8,26] := 9;   let TLCT[3,9,9,27] := 8;
let TLCT[3,9,9,28] := 6;   let TLCT[3,9,10,29] := 0;  let TLCT[3,9,10,30] := 6;
let TLCT[3,9,11,31] := 4;  let TLCT[3,9,11,32] := 10; let TLCT[3,9,11,33] := 6;
let TLCT[3,9,11,34] := 1;  let TLCT[3,9,12,35] := 4;  let TLCT[3,9,12,36] := 2;
let TLCT[3,9,12,37] := 2;  let TLCT[3,9,12,38] := 7;

let TLCT[3,10,1,1] := 1;   let TLCT[3,10,1,2] := 9;   let TLCT[3,10,1,3] := 10;
let TLCT[3,10,1,4] := 0;   let TLCT[3,10,2,5] := 2;   let TLCT[3,10,2,6] := 1;
let TLCT[3,10,2,7] := 0;   let TLCT[3,10,2,8] := 6;   let TLCT[3,10,3,9] := 10;
let TLCT[3,10,3,10] := 0;  let TLCT[3,10,3,11] := 1;  let TLCT[3,10,4,12] := 5;
let TLCT[3,10,4,13] := 1;  let TLCT[3,10,4,14] := 10; let TLCT[3,10,5,15] := 4;
let TLCT[3,10,5,16] := 10; let TLCT[3,10,6,17] := 8;  let TLCT[3,10,6,18] := 8;
let TLCT[3,10,6,19] := 7;  let TLCT[3,10,6,20] := 2;  let TLCT[3,10,7,21] := 7;
let TLCT[3,10,7,22] := 5;  let TLCT[3,10,7,23] := 8;  let TLCT[3,10,8,24] := 3;
```

```
let TLCT[3,10,8,25] :=  0;   let TLCT[3,10,8,26] :=  9;   let TLCT[3,10,9,27] :=  10;
let TLCT[3,10,9,28] :=  0;   let TLCT[3,10,10,29] :=  5; let TLCT[3,10,10,30] :=  2;
let TLCT[3,10,11,31] :=  6; let TLCT[3,10,11,32] :=  7; let TLCT[3,10,11,33] :=  9;
let TLCT[3,10,11,34] :=  2; let TLCT[3,10,12,35] :=  10;let TLCT[3,10,12,36] :=  5;
let TLCT[3,10,12,37] :=  10;let TLCT[3,10,12,38] :=  1;

let TLCT[3,11,1,1] :=  7;     let TLCT[3,11,1,2] := 2;      let TLCT[3,11,1,3] :=  2;
let TLCT[3,11,1,4] :=  6;     let TLCT[3,11,2,5] :=  8;     let TLCT[3,11,2,6] :=  7;
let TLCT[3,11,2,7] :=  9;     let TLCT[3,11,2,8] :=  0;     let TLCT[3,11,3,9] :=  9;
let TLCT[3,11,3,10] :=  10; let TLCT[3,11,3,11] :=  0;   let TLCT[3,11,4,12] :=  1;
let TLCT[3,11,4,13] :=  4;   let TLCT[3,11,4,14] :=  8;   let TLCT[3,11,5,15] :=  9;
let TLCT[3,11,5,16] :=  4;   let TLCT[3,11,6,17] :=  8;   let TLCT[3,11,6,18] :=  1;
let TLCT[3,11,6,19] :=  3;   let TLCT[3,11,6,20] :=  1;   let TLCT[3,11,7,21] :=  4;
let TLCT[3,11,7,22] :=  8;   let TLCT[3,11,7,23] :=  10; let TLCT[3,11,8,24] :=  2;
let TLCT[3,11,8,25] :=  1;   let TLCT[3,11,8,26] :=  0;   let TLCT[3,11,9,27] :=  7;
let TLCT[3,11,9,28] :=  7;   let TLCT[3,11,10,29] :=  7; let TLCT[3,11,10,30] :=  8;
let TLCT[3,11,11,31] :=  7; let TLCT[3,11,11,32] :=  10;let TLCT[3,11,11,33] :=  0;
let TLCT[3,11,11,34] :=  7; let TLCT[3,11,12,35] :=  9; let TLCT[3,11,12,36] :=  1;
let TLCT[3,11,12,37] :=  3; let TLCT[3,11,12,38] :=  10;

let TLCT[4,12,1,1] :=  6;     let TLCT[4,12,1,2] := 6;      let TLCT[4,12,1,3] :=  1;
let TLCT[4,12,1,4] :=  1;     let TLCT[4,12,2,5] :=  1;     let TLCT[4,12,2,6] :=  5;
let TLCT[4,12,2,7] :=  8;     let TLCT[4,12,2,8] :=  8;     let TLCT[4,12,3,9] :=  2;
let TLCT[4,12,3,10] :=  3;   let TLCT[4,12,3,11] :=  4;   let TLCT[4,12,4,12] :=  0;
let TLCT[4,12,4,13] := 10;  let TLCT[4,12,4,14] :=  4;   let TLCT[4,12,5,15] :=  6;
let TLCT[4,12,5,16] :=  2;   let TLCT[4,12,6,17] :=  2;   let TLCT[4,12,6,18] :=  8;
let TLCT[4,12,6,19] :=  1;   let TLCT[4,12,6,20] :=  5;   let TLCT[4,12,7,21] :=  6;
let TLCT[4,12,7,22] :=  3;   let TLCT[4,12,7,23] :=  7;   let TLCT[4,12,8,24] :=  3;
let TLCT[4,12,8,25] :=  2;   let TLCT[4,12,8,26] :=  2;   let TLCT[4,12,9,27] :=  2;
let TLCT[4,12,9,28] :=  1;   let TLCT[4,12,10,29] :=  2; let TLCT[4,12,10,30] :=  6;
let TLCT[4,12,11,31] :=  10;let TLCT[4,12,11,32] :=  5; let TLCT[4,12,11,33] :=  7;
```

let TLCT[4,12,11,34] := 0; let TLCT[4,12,12,35] := 10;let TLCT[4,12,12,36] := 2;
let TLCT[4,12,12,37] := 3; let TLCT[4,12,12,38] := 5;

let TLCT[4,13,1,1] := 3; let TLCT[4,13,1,2] := 3; let TLCT[4,13,1,3] := 1;
let TLCT[4,13,1,4] := 7; let TLCT[4,13,2,5] := 2; let TLCT[4,13,2,6] := 6;
let TLCT[4,13,2,7] := 4; let TLCT[4,13,2,8] := 7; let TLCT[4,13,3,9] := 5;
let TLCT[4,13,3,10] := 5; let TLCT[4,13,3,11] := 6; let TLCT[4,13,4,12] := 8;
let TLCT[4,13,4,13] := 0; let TLCT[4,13,4,14] := 5; let TLCT[4,13,5,15] := 9;
let TLCT[4,13,5,16] := 3; let TLCT[4,13,6,17] := 4; let TLCT[4,13,6,18] := 2;
let TLCT[4,13,6,19] := 2; let TLCT[4,13,6,20] := 1; let TLCT[4,13,7,21] := 9;
let TLCT[4,13,7,22] := 6; let TLCT[4,13,7,23] := 3; let TLCT[4,13,8,24] := 9;
let TLCT[4,13,8,25] := 5; let TLCT[4,13,8,26] := 5; let TLCT[4,13,9,27] := 9;
let TLCT[4,13,9,28] := 5; let TLCT[4,13,10,29] := 3; let TLCT[4,13,10,30] := 2;
let TLCT[4,13,11,31] := 2; let TLCT[4,13,11,32] := 2; let TLCT[4,13,11,33] := 5;
let TLCT[4,13,11,34] := 7; let TLCT[4,13,12,35] := 6; let TLCT[4,13,12,36] := 9;
let TLCT[4,13,12,37] := 1; let TLCT[4,13,12,38] := 6;

let TLCT[4,14,1,1] := 7; let TLCT[4,14,1,2] := 2; let TLCT[4,14,1,3] := 1;
let TLCT[4,14,1,4] := 6; let TLCT[4,14,2,5] := 7; let TLCT[4,14,2,6] := 8;
let TLCT[4,14,2,7] := 6; let TLCT[4,14,2,8] := 3; let TLCT[4,14,3,9] := 7;
let TLCT[4,14,3,10] := 10; let TLCT[4,14,3,11] := 4; let TLCT[4,14,4,12] := 9;
let TLCT[4,14,4,13] := 2; let TLCT[4,14,4,14] := 0; let TLCT[4,14,5,15] := 2;
let TLCT[4,14,5,16] := 7; let TLCT[4,14,6,17] := 6; let TLCT[4,14,6,18] := 1;
let TLCT[4,14,6,19] := 5; let TLCT[4,14,6,20] := 3; let TLCT[4,14,7,21] := 6;
let TLCT[4,14,7,22] := 5; let TLCT[4,14,7,23] := 9; let TLCT[4,14,8,24] := 9;
let TLCT[4,14,8,25] := 7; let TLCT[4,14,8,26] := 4; let TLCT[4,14,9,27] := 4;
let TLCT[4,14,9,28] := 3; let TLCT[4,14,10,29] := 9; let TLCT[4,14,10,30] := 1;
let TLCT[4,14,11,31] := 6; let TLCT[4,14,11,32] := 7; let TLCT[4,14,11,33] := 1;
let TLCT[4,14,11,34] := 9; let TLCT[4,14,12,35] := 0;let TLCT[4,14,12,36] := 10;
let TLCT[4,14,12,37] := 3; let TLCT[4,14,12,38] := 5;

let TLCT[5,15,1,1] := 10; let TLCT[5,15,1,2] := 7; let TLCT[5,15,1,3] := 6;
let TLCT[5,15,1,4] := 6; let TLCT[5,15,2,5] := 7; let TLCT[5,15,2,6] := 5;

let TLCT[5,15,2,7] := 4; let TLCT[5,15,2,8] := 6; let TLCT[5,15,3,9] := 6;
let TLCT[5,15,3,10] := 8; let TLCT[5,15,3,11] := 8; let TLCT[5,15,4,12] := 1;
let TLCT[5,15,4,13] := 6; let TLCT[5,15,4,14] := 3; let TLCT[5,15,5,15] := 0;
let TLCT[5,15,5,16] := 3; let TLCT[5,15,6,17] := 0; let TLCT[5,15,6,18] := 9;
let TLCT[5,15,6,19] := 7; let TLCT[5,15,6,20] := 10; let TLCT[5,15,7,21] := 4;
let TLCT[5,15,7,22] := 2; let TLCT[5,15,7,23] := 6; let TLCT[5,15,8,24] := 5;
let TLCT[5,15,8,25] := 9; let TLCT[5,15,8,26] := 1; let TLCT[5,15,9,27] := 9;
let TLCT[5,15,9,28] := 6; let TLCT[5,15,10,29] := 9; let TLCT[5,15,10,30] := 8;
let TLCT[5,15,11,31] := 2; let TLCT[5,15,11,32] := 7; let TLCT[5,15,11,33] := 4;
let TLCT[5,15,11,34] := 2; let TLCT[5,15,12,35] := 10;let TLCT[5,15,12,36] := 0;
let TLCT[5,15,12,37] := 3; let TLCT[5,15,12,38] := 1;

let TLCT[5,16,1,1] := 4; let TLCT[5,16,1,2] := 9; let TLCT[5,16,1,3] := 3;
let TLCT[5,16,1,4] := 3; let TLCT[5,16,2,5] := 5; let TLCT[5,16,2,6] := 1;
let TLCT[5,16,2,7] := 4; let TLCT[5,16,2,8] := 8; let TLCT[5,16,3,9] := 7;
let TLCT[5,16,3,10] := 10; let TLCT[5,16,3,11] := 1; let TLCT[5,16,4,12] := 8;
let TLCT[5,16,4,13] := 1; let TLCT[5,16,4,14] := 4; let TLCT[5,16,5,15] := 10;
let TLCT[5,16,5,16] := 0; let TLCT[5,16,6,17] := 2; let TLCT[5,16,6,18] := 0;
let TLCT[5,16,6,19] := 3; let TLCT[5,16,6,20] := 10; let TLCT[5,16,7,21] := 0;
let TLCT[5,16,7,22] := 2; let TLCT[5,16,7,23] := 10; let TLCT[5,16,8,24] := 9;
let TLCT[5,16,8,25] := 5; let TLCT[5,16,8,26] := 2; let TLCT[5,16,9,27] := 2;
let TLCT[5,16,9,28] := 8; let TLCT[5,16,10,29] := 5; let TLCT[5,16,10,30] := 8;
let TLCT[5,16,11,31] := 1; let TLCT[5,16,11,32] := 5; let TLCT[5,16,11,33] := 9;
let TLCT[5,16,11,34] := 1; let TLCT[5,16,12,35] := 6; let TLCT[5,16,12,36] := 2;
let TLCT[5,16,12,37] := 0; let TLCT[5,16,12,38] := 2;

let TLCT[6,17,1,1] := 4; let TLCT[6,17,1,2] := 6; let TLCT[6,17,1,3] := 10;
let TLCT[6,17,1,4] := 8; let TLCT[6,17,2,5] := 1; let TLCT[6,17,2,6] := 3;
let TLCT[6,17,2,7] := 1; let TLCT[6,17,2,8] := 4; let TLCT[6,17,3,9] := 2;
let TLCT[6,17,3,10] := 10; let TLCT[6,17,3,11] := 2; let TLCT[6,17,4,12] := 5;
let TLCT[6,17,4,13] := 1; let TLCT[6,17,4,14] := 4; let TLCT[6,17,5,15] := 0;
let TLCT[6,17,5,16] := 6; let TLCT[6,17,6,17] := 0; let TLCT[6,17,6,18] := 4;

```
let TLCT[6,17,6,19] := 10; let TLCT[6,17,6,20] := 1;  let TLCT[6,17,7,21] := 5;
let TLCT[6,17,7,22] := 3;  let TLCT[6,17,7,23] := 5;  let TLCT[6,17,8,24] := 3;
let TLCT[6,17,8,25] := 6;  let TLCT[6,17,8,26] := 5;  let TLCT[6,17,9,27] := 1;
let TLCT[6,17,9,28] := 9;  let TLCT[6,17,10,29] := 6; let TLCT[6,17,10,30] := 4;
let TLCT[6,17,11,31] := 5; let TLCT[6,17,11,32] := 6; let TLCT[6,17,11,33] := 6;
let TLCT[6,17,11,34] := 9; let TLCT[6,17,12,35] := 9; let TLCT[6,17,12,36] := 0;
let TLCT[6,17,12,37] := 7; let TLCT[6,17,12,38] := 8;

let TLCT[6,18,1,1] := 2;   let TLCT[6,18,1,2] := 3;   let TLCT[6,18,1,3] := 3;
let TLCT[6,18,1,4] := 8;   let TLCT[6,18,2,5] := 8;   let TLCT[6,18,2,6] := 9;
let TLCT[6,18,2,7] := 5;   let TLCT[6,18,2,8] := 5;   let TLCT[6,18,3,9] := 8;
let TLCT[6,18,3,10] := 4;  let TLCT[6,18,3,11] := 7;  let TLCT[6,18,4,12] := 2;
let TLCT[6,18,4,13] := 9;  let TLCT[6,18,4,14] := 10; let TLCT[6,18,5,15] := 5;
let TLCT[6,18,5,16] := 0;  let TLCT[6,18,6,17] := 8;  let TLCT[6,18,6,18] := 0;
let TLCT[6,18,6,19] := 6;  let TLCT[6,18,6,20] := 5;  let TLCT[6,18,7,21] := 0;
let TLCT[6,18,7,22] := 3;  let TLCT[6,18,7,23] := 6;  let TLCT[6,18,8,24] := 10;
let TLCT[6,18,8,25] := 5;  let TLCT[6,18,8,26] := 4;  let TLCT[6,18,9,27] := 9;
let TLCT[6,18,9,28] := 9;  let TLCT[6,18,10,29] := 10;let TLCT[6,18,10,30] := 1;
let TLCT[6,18,11,31] := 3; let TLCT[6,18,11,32] := 3; let TLCT[6,18,11,33] := 7;
let TLCT[6,18,11,34] := 9; let TLCT[6,18,12,35] := 9;let TLCT[6,18,12,36] := 10;
let TLCT[6,18,12,37] := 0; let TLCT[6,18,12,38] := 3;

let TLCT[6,19,1,1] := 10;  let TLCT[6,19,1,2] := 9;   let TLCT[6,19,1,3] := 5;
let TLCT[6,19,1,4] := 9;   let TLCT[6,19,2,5] := 5;   let TLCT[6,19,2,6] := 2;
let TLCT[6,19,2,7] := 6;   let TLCT[6,19,2,8] := 8;   let TLCT[6,19,3,9] := 4;
let TLCT[6,19,3,10] := 3;  let TLCT[6,19,3,11] := 5;  let TLCT[6,19,4,12] := 5;
let TLCT[6,19,4,13] := 2;  let TLCT[6,19,4,14] := 3;  let TLCT[6,19,5,15] := 10;
let TLCT[6,19,5,16] := 1;  let TLCT[6,19,6,17] := 5;  let TLCT[6,19,6,18] := 7;
let TLCT[6,19,6,19] := 0;  let TLCT[6,19,6,20] := 6;  let TLCT[6,19,7,21] := 1;
let TLCT[6,19,7,22] := 0;  let TLCT[6,19,7,23] := 8;  let TLCT[6,19,8,24] := 2;
let TLCT[6,19,8,25] := 1;  let TLCT[6,19,8,26] := 6;  let TLCT[6,19,9,27] := 9;
let TLCT[6,19,9,28] := 8;  let TLCT[6,19,10,29] := 4; let TLCT[6,19,10,30] := 3;
```

```
let TLCT[6,19,11,31] :=  0; let TLCT[6,19,11,32] :=  3; let TLCT[6,19,11,33] :=  7;
let TLCT[6,19,11,34] :=  9; let TLCT[6,19,12,35] :=  3; let TLCT[6,19,12,36] :=  1;
let TLCT[6,19,12,37] :=  4; let TLCT[6,19,12,38] :=  0;

let TLCT[6,20,1,1] :=  3;    let TLCT[6,20,1,2] := 10;    let TLCT[6,20,1,3] :=  5;
let TLCT[6,20,1,4] :=  7;    let TLCT[6,20,2,5] :=  4;    let TLCT[6,20,2,6] :=  7;
let TLCT[6,20,2,7] :=  3;    let TLCT[6,20,2,8] :=  5;    let TLCT[6,20,3,9] :=  6;
let TLCT[6,20,3,10] :=  1;   let TLCT[6,20,3,11] :=  4;   let TLCT[6,20,4,12] :=  2;
let TLCT[6,20,4,13] :=  2;   let TLCT[6,20,4,14] :=  9;   let TLCT[6,20,5,15] :=  5;
let TLCT[6,20,5,16] :=  6;   let TLCT[6,20,6,17] :=  7;   let TLCT[6,20,6,18] :=  10;
let TLCT[6,20,6,19] :=  8;   let TLCT[6,20,6,20] :=  0;   let TLCT[6,20,7,21] :=  9;
let TLCT[6,20,7,22] :=  3;   let TLCT[6,20,7,23] :=  0;   let TLCT[6,20,8,24] :=  7;
let TLCT[6,20,8,25] :=  3;   let TLCT[6,20,8,26] :=  4;   let TLCT[6,20,9,27] :=  10;
let TLCT[6,20,9,28] :=  2;   let TLCT[6,20,10,29] :=  9; let TLCT[6,20,10,30] :=  3;
let TLCT[6,20,11,31] :=  8; let TLCT[6,20,11,32] :=  0;let TLCT[6,20,11,33] :=  10;
let TLCT[6,20,11,34] :=  10;let TLCT[6,20,12,35] :=  8; let TLCT[6,20,12,36] :=  9;
let TLCT[6,20,12,37] :=  10;let TLCT[6,20,12,38] :=  5;

let TLCT[7,21,1,1] :=  1;    let TLCT[7,21,1,2] := 4;     let TLCT[7,21,1,3] :=  7;
let TLCT[7,21,1,4] :=  10;   let TLCT[7,21,2,5] :=  10;   let TLCT[7,21,2,6] :=  5;
let TLCT[7,21,2,7] :=  6;    let TLCT[7,21,2,8] :=  9;    let TLCT[7,21,3,9] :=  4;
let TLCT[7,21,3,10] :=  9;   let TLCT[7,21,3,11] :=  9;   let TLCT[7,21,4,12] :=  7;
let TLCT[7,21,4,13] :=  4;   let TLCT[7,21,4,14] :=  8;   let TLCT[7,21,5,15] :=  10;
let TLCT[7,21,5,16] :=  0;   let TLCT[7,21,6,17] :=  10; let TLCT[7,21,6,18] :=  0;
let TLCT[7,21,6,19] :=  7;   let TLCT[7,21,6,20] :=  4;   let TLCT[7,21,7,21] :=  0;
let TLCT[7,21,7,22] :=  2;   let TLCT[7,21,7,23] :=  1;   let TLCT[7,21,8,24] :=  10;
let TLCT[7,21,8,25] :=  9;   let TLCT[7,21,8,26] :=  4;   let TLCT[7,21,9,27] :=  2;
let TLCT[7,21,9,28] :=  9;   let TLCT[7,21,10,29] :=  9; let TLCT[7,21,10,30] :=  7;
let TLCT[7,21,11,31] :=  3; let TLCT[7,21,11,32] :=  7; let TLCT[7,21,11,33] :=  6;
let TLCT[7,21,11,34] :=  7; let TLCT[7,21,12,35] :=  7; let TLCT[7,21,12,36] :=  4;
let TLCT[7,21,12,37] :=  0; let TLCT[7,21,12,38] :=  4;
```

```
let TLCT[7,22,1,1] :=  3;    let TLCT[7,22,1,2] := 1;     let TLCT[7,22,1,3] :=  10;
let TLCT[7,22,1,4] :=  6;    let TLCT[7,22,2,5] :=  6;    let TLCT[7,22,2,6] :=  8;
let TLCT[7,22,2,7] :=  5;    let TLCT[7,22,2,8] :=  2;    let TLCT[7,22,3,9] :=  4;
let TLCT[7,22,3,10] :=  2;   let TLCT[7,22,3,11] :=  9;   let TLCT[7,22,4,12] :=  2;
let TLCT[7,22,4,13] :=  10;  let TLCT[7,22,4,14] :=  7;   let TLCT[7,22,5,15] :=  4;
let TLCT[7,22,5,16] :=  10;  let TLCT[7,22,6,17] :=  10;  let TLCT[7,22,6,18] :=  9;
let TLCT[7,22,6,19] :=  0;   let TLCT[7,22,6,20] :=  4;   let TLCT[7,22,7,21] :=  7;
let TLCT[7,22,7,22] :=  0;   let TLCT[7,22,7,23] :=  5;   let TLCT[7,22,8,24] :=  5;
let TLCT[7,22,8,25] :=  10;  let TLCT[7,22,8,26] :=  1;   let TLCT[7,22,9,27] :=  6;
let TLCT[7,22,9,28] :=  7;   let TLCT[7,22,10,29] :=  4;  let TLCT[7,22,10,30] :=  4;
let TLCT[7,22,11,31] :=  0;  let TLCT[7,22,11,32] :=  9;  let TLCT[7,22,11,33] :=  5;
let TLCT[7,22,11,34] :=  6;  let TLCT[7,22,12,35] :=  7;  let TLCT[7,22,12,36] :=  5;
let TLCT[7,22,12,37] :=  5;  let TLCT[7,22,12,38] :=  0;

let TLCT[7,23,1,1] :=  2;    let TLCT[7,23,1,2] := 8;     let TLCT[7,23,1,3] :=  10;
let TLCT[7,23,1,4] :=  10;   let TLCT[7,23,2,5] :=  9;    let TLCT[7,23,2,6] :=  3;
let TLCT[7,23,2,7] :=  10;   let TLCT[7,23,2,8] :=  6;    let TLCT[7,23,3,9] :=  8;
let TLCT[7,23,3,10] :=  7;   let TLCT[7,23,3,11] :=  7;   let TLCT[7,23,4,12] :=  10;
let TLCT[7,23,4,13] :=  8;   let TLCT[7,23,4,14] :=  6;   let TLCT[7,23,5,15] :=  5;
let TLCT[7,23,5,16] :=  2;   let TLCT[7,23,6,17] :=  1;   let TLCT[7,23,6,18] :=  3;
let TLCT[7,23,6,19] :=  9;   let TLCT[7,23,6,20] :=  0;   let TLCT[7,23,7,21] :=  7;
let TLCT[7,23,7,22] :=  6;   let TLCT[7,23,7,23] :=  0;   let TLCT[7,23,8,24] :=  5;
let TLCT[7,23,8,25] :=  8;   let TLCT[7,23,8,26] :=  8;   let TLCT[7,23,9,27] :=  1;
let TLCT[7,23,9,28] :=  8;   let TLCT[7,23,10,29] :=  5;  let TLCT[7,23,10,30] :=  1;
let TLCT[7,23,11,31] :=  6;  let TLCT[7,23,11,32] :=  0;  let TLCT[7,23,11,33] :=  5;
let TLCT[7,23,11,34] :=  4;  let TLCT[7,23,12,35] :=  3;  let TLCT[7,23,12,36] :=  9;
let TLCT[7,23,12,37] :=  1;  let TLCT[7,23,12,38] :=  1;

let TLCT[8,24,1,1] :=  5;    let TLCT[8,24,1,2] := 7;     let TLCT[8,24,1,3] :=  0;
let TLCT[8,24,1,4] :=  10;   let TLCT[8,24,2,5] :=  10;   let TLCT[8,24,2,6] :=  8;
let TLCT[8,24,2,7] :=  9;    let TLCT[8,24,2,8] :=  9;    let TLCT[8,24,3,9] :=  4;
let TLCT[8,24,3,10] :=  1;   let TLCT[8,24,3,11] :=  2;   let TLCT[8,24,4,12] :=  4;
```

```
let TLCT[8,24,4,13] :=  3;   let TLCT[8,24,4,14] :=  1;   let TLCT[8,24,5,15] :=  10;
let TLCT[8,24,5,16] :=  9;   let TLCT[8,24,6,17] :=  8;   let TLCT[8,24,6,18] :=  3;
let TLCT[8,24,6,19] :=  1;   let TLCT[8,24,6,20] :=  7;   let TLCT[8,24,7,21] :=  4;
let TLCT[8,24,7,22] :=  10; let TLCT[8,24,7,23] :=  7;   let TLCT[8,24,8,24] :=  0;
let TLCT[8,24,8,25] :=  10; let TLCT[8,24,8,26] :=  3;   let TLCT[8,24,9,27] :=  0;
let TLCT[8,24,9,28] :=  2;   let TLCT[8,24,10,29] :=  1; let TLCT[8,24,10,30] :=  4;
let TLCT[8,24,11,31] :=  8; let TLCT[8,24,11,32] :=  9; let TLCT[8,24,11,33] :=  2;
let TLCT[8,24,11,34] :=  1; let TLCT[8,24,12,35] :=  2; let TLCT[8,24,12,36] :=  8;
let TLCT[8,24,12,37] :=  9; let TLCT[8,24,12,38] :=  3;

let TLCT[8,25,1,1] :=  10;   let TLCT[8,25,1,2] := 8;      let TLCT[8,25,1,3] :=  3;
let TLCT[8,25,1,4] :=  0;     let TLCT[8,25,2,5] :=  9;     let TLCT[8,25,2,6] :=  3;
let TLCT[8,25,2,7] :=  0;     let TLCT[8,25,2,8] :=  6;     let TLCT[8,25,3,9] :=  2;
let TLCT[8,25,3,10] :=  0;   let TLCT[8,25,3,11] :=  4;   let TLCT[8,25,4,12] :=  6;
let TLCT[8,25,4,13] :=  3;   let TLCT[8,25,4,14] :=  6;   let TLCT[8,25,5,15] :=  8;
let TLCT[8,25,5,16] :=  5;   let TLCT[8,25,6,17] :=  1;   let TLCT[8,25,6,18] :=  3;
let TLCT[8,25,6,19] :=  2;   let TLCT[8,25,6,20] :=  5;   let TLCT[8,25,7,21] :=  2;
let TLCT[8,25,7,22] :=  4;   let TLCT[8,25,7,23] :=  10; let TLCT[8,25,8,24] :=  1;
let TLCT[8,25,8,25] :=  0;   let TLCT[8,25,8,26] :=  9;   let TLCT[8,25,9,27] :=  5;
let TLCT[8,25,9,28] :=  0;   let TLCT[8,25,10,29] :=  9; let TLCT[8,25,10,30] :=  8;
let TLCT[8,25,11,31] :=  1; let TLCT[8,25,11,32] :=  6; let TLCT[8,25,11,33] :=  5;
let TLCT[8,25,11,34] :=  6; let TLCT[8,25,12,35] :=  1; let TLCT[8,25,12,36] :=  9;
let TLCT[8,25,12,37] :=  1; let TLCT[8,25,12,38] :=  4;

let TLCT[8,26,1,1] :=  7;     let TLCT[8,26,1,2] := 8;      let TLCT[8,26,1,3] :=  7;
let TLCT[8,26,1,4] :=  7;     let TLCT[8,26,2,5] :=  9;     let TLCT[8,26,2,6] :=  7;
let TLCT[8,26,2,7] :=  3;     let TLCT[8,26,2,8] :=  0;     let TLCT[8,26,3,9] :=  1;
let TLCT[8,26,3,10] :=  8;   let TLCT[8,26,3,11] :=  0;   let TLCT[8,26,4,12] :=  9;
let TLCT[8,26,4,13] :=  6;   let TLCT[8,26,4,14] :=  10; let TLCT[8,26,5,15] :=  4;
let TLCT[8,26,5,16] :=  7;   let TLCT[8,26,6,17] :=  10; let TLCT[8,26,6,18] :=  10;
let TLCT[8,26,6,19] :=  5;   let TLCT[8,26,6,20] :=  4;   let TLCT[8,26,7,21] :=  2;
let TLCT[8,26,7,22] :=  9;   let TLCT[8,26,7,23] :=  7;   let TLCT[8,26,8,24] :=  9;
```

```
let TLCT[8,26,8,25] := 6;  let TLCT[8,26,8,26] := 0;  let TLCT[8,26,9,27] := 10;
let TLCT[8,26,9,28] := 9;  let TLCT[8,26,10,29] := 5; let TLCT[8,26,10,30] := 8;
let TLCT[8,26,11,31] := 1; let TLCT[8,26,11,32] := 1; let TLCT[8,26,11,33] := 0;
let TLCT[8,26,11,34] := 7; let TLCT[8,26,12,35] := 4; let TLCT[8,26,12,36] := 1;
let TLCT[8,26,12,37] := 4; let TLCT[8,26,12,38] := 4;

let TLCT[9,27,1,1] := 6;   let TLCT[9,27,1,2] := 2;   let TLCT[9,27,1,3] := 0;
let TLCT[9,27,1,4] := 1;   let TLCT[9,27,2,5] := 6;   let TLCT[9,27,2,6] := 1;
let TLCT[9,27,2,7] := 7;   let TLCT[9,27,2,8] := 2;   let TLCT[9,27,3,9] := 6;
let TLCT[9,27,3,10] := 3;  let TLCT[9,27,3,11] := 1;  let TLCT[9,27,4,12] := 4;
let TLCT[9,27,4,13] := 7;  let TLCT[9,27,4,14] := 4;  let TLCT[9,27,5,15] := 9;
let TLCT[9,27,5,16] := 4;  let TLCT[9,27,6,17] := 8;  let TLCT[9,27,6,18] := 2;
let TLCT[9,27,6,19] := 6;  let TLCT[9,27,6,20] := 4;  let TLCT[9,27,7,21] := 10;
let TLCT[9,27,7,22] := 10; let TLCT[9,27,7,23] := 9;  let TLCT[9,27,8,24] := 0;
let TLCT[9,27,8,25] := 4;  let TLCT[9,27,8,26] := 5;  let TLCT[9,27,9,27] := 0;
let TLCT[9,27,9,28] := 2;  let TLCT[9,27,10,29] := 3; let TLCT[9,27,10,30] := 4;
let TLCT[9,27,11,31] := 7; let TLCT[9,27,11,32] := 8; let TLCT[9,27,11,33] := 6;
let TLCT[9,27,11,34] := 9; let TLCT[9,27,12,35] := 8; let TLCT[9,27,12,36] := 7;
let TLCT[9,27,12,37] := 8; let TLCT[9,27,12,38] := 6;

let TLCT[9,28,1,1] := 9;   let TLCT[9,28,1,2] := 4;   let TLCT[9,28,1,3] := 1;
let TLCT[9,28,1,4] := 0;   let TLCT[9,28,2,5] := 6;   let TLCT[9,28,2,6] := 2;
let TLCT[9,28,2,7] := 0;   let TLCT[9,28,2,8] := 4;   let TLCT[9,28,3,9] := 6;
let TLCT[9,28,3,10] := 0;  let TLCT[9,28,3,11] := 4;  let TLCT[9,28,4,12] := 2;
let TLCT[9,28,4,13] := 6;  let TLCT[9,28,4,14] := 9;  let TLCT[9,28,5,15] := 9;
let TLCT[9,28,5,16] := 3;  let TLCT[9,28,6,17] := 2;  let TLCT[9,28,6,18] := 8;
let TLCT[9,28,6,19] := 3;  let TLCT[9,28,6,20] := 4;  let TLCT[9,28,7,21] := 6;
let TLCT[9,28,7,22] := 3;  let TLCT[9,28,7,23] := 7;  let TLCT[9,28,8,24] := 3;
let TLCT[9,28,8,25] := 0;  let TLCT[9,28,8,26] := 4;  let TLCT[9,28,9,27] := 2;
let TLCT[9,28,9,28] := 0;  let TLCT[9,28,10,29] := 7; let TLCT[9,28,10,30] := 6;
let TLCT[9,28,11,31] := 5; let TLCT[9,28,11,32] := 3; let TLCT[9,28,11,33] := 8;
```

let TLCT[9,28,11,34] := 2; let TLCT[9,28,12,35] := 10;let TLCT[9,28,12,36] := 6;
let TLCT[9,28,12,37] := 1; let TLCT[9,28,12,38] := 3;

let TLCT[10,29,1,1] := 0; let TLCT[10,29,1,2] := 8; let TLCT[10,29,1,3] := 9;
let TLCT[10,29,1,4] := 3; let TLCT[10,29,2,5] := 0; let TLCT[10,29,2,6] := 4;
let TLCT[10,29,2,7] := 2; let TLCT[10,29,2,8] := 10; let TLCT[10,29,3,9] := 0;
let TLCT[10,29,3,10] := 7; let TLCT[10,29,3,11] := 2; let TLCT[10,29,4,12] := 3;
let TLCT[10,29,4,13] := 7; let TLCT[10,29,4,14] := 7; let TLCT[10,29,5,15] := 7;
let TLCT[10,29,5,16] := 5; let TLCT[10,29,6,17] := 6; let TLCT[10,29,6,18] := 1;
let TLCT[10,29,6,19] := 2; let TLCT[10,29,6,20] := 4; let TLCT[10,29,7,21] := 5;
let TLCT[10,29,7,22] := 4; let TLCT[10,29,7,23] := 6; let TLCT[10,29,8,24] := 9;
let TLCT[10,29,8,25] := 5; let TLCT[10,29,8,26] := 8; let TLCT[10,29,9,27] := 3;
let TLCT[10,29,9,28] := 4;let TLCT[10,29,10,29] := 0;let TLCT[10,29,10,30] :=
9;

let TLCT[10,29,11,31] := 4;let TLCT[10,29,11,32] := 1;let TLCT[10,29,11,33] :=
3;

let TLCT[10,29,11,34] := 5;let TLCT[10,29,12,35] := 7;let TLCT[10,29,12,36] :=
3; let TLCT[10,29,12,37] := 9;let TLCT[10,29,12,38] := 2;

let TLCT[10,30,1,1] := 8; let TLCT[10,30,1,2] := 0; let TLCT[10,30,1,3] := 1;
let TLCT[10,30,1,4] := 8; let TLCT[10,30,2,5] := 6; let TLCT[10,30,2,6] := 0;
let TLCT[10,30,2,7] := 4; let TLCT[10,30,2,8] := 2; let TLCT[10,30,3,9] := 4;
let TLCT[10,30,3,10] := 1; let TLCT[10,30,3,11] := 6; let TLCT[10,30,4,12] := 7;
let TLCT[10,30,4,13] := 1; let TLCT[10,30,4,14] := 3; let TLCT[10,30,5,15] := 7;
let TLCT[10,30,5,16] := 7; let TLCT[10,30,6,17] := 3; let TLCT[10,30,6,18] := 8;
let TLCT[10,30,6,19] := 10;let TLCT[10,30,6,20] := 6; let TLCT[10,30,7,21] := 3;
let TLCT[10,30,7,22] := 2; let TLCT[10,30,7,23] := 4; let TLCT[10,30,8,24] := 6;
let TLCT[10,30,8,25] := 7; let TLCT[10,30,8,26] := 10;let TLCT[10,30,9,27] := 2;
let TLCT[10,30,9,28] := 9;let TLCT[10,30,10,29] := 10;let TLCT[10,30,10,30] :=
0;

let TLCT[10,30,11,31] := 3;let TLCT[10,30,11,32] := 9;let TLCT[10,30,11,33] := 4;

let TLCT[10,30,11,34] := 6;let TLCT[10,30,12,35] := 6;let TLCT[10,30,12,36] := 3;

let TLCT[10,30,12,37] := 10;let TLCT[10,30,12,38] := 2;

let TLCT[11,31,1,1] := 4; let TLCT[11,31,1,2] := 5; let TLCT[11,31,1,3] := 8;
let TLCT[11,31,1,4] := 3; let TLCT[11,31,2,5] := 10; let TLCT[11,31,2,6] := 1;
let TLCT[11,31,2,7] := 10; let TLCT[11,31,2,8] := 1; let TLCT[11,31,3,9] := 2;
let TLCT[11,31,3,10] := 10;let TLCT[11,31,3,11] := 9; let TLCT[11,31,4,12] := 7;
let TLCT[11,31,4,13] := 3; let TLCT[11,31,4,14] := 2; let TLCT[11,31,5,15] := 3;
let TLCT[11,31,5,16] := 8; let TLCT[11,31,6,17] := 9; let TLCT[11,31,6,18] := 2;
let TLCT[11,31,6,19] := 0; let TLCT[11,31,6,20] := 4; let TLCT[11,31,7,21] := 10;

let TLCT[11,31,7,22] := 0;let TLCT[11,31,7,23] := 3;let TLCT[11,31,8,24] := 7;

let TLCT[11,31,8,25] := 4; let TLCT[11,31,8,26] := 10;let TLCT[11,31,9,27] := 5;
let TLCT[11,31,9,28] := 10;let TLCT[11,31,10,29] := 5;let TLCT[11,31,10,30] := 8;

let TLCT[11,31,11,31] := 0;let TLCT[11,31,11,32] := 1;let TLCT[11,31,11,33] := 6;

let TLCT[11,31,11,34] := 2;let TLCT[11,31,12,35] := 10;let TLCT[11,31,12,36] := 3;

let TLCT[11,31,12,37] := 1;let TLCT[11,31,12,38] := 0;

let TLCT[11,32,1,1] := 6; let TLCT[11,32,1,2] := 7; let TLCT[11,32,1,3] := 9;
let TLCT[11,32,1,4] := 8; let TLCT[11,32,2,5] := 7; let TLCT[11,32,2,6] := 4;
let TLCT[11,32,2,7] := 7; let TLCT[11,32,2,8] := 8; let TLCT[11,32,3,9] := 4;
let TLCT[11,32,3,10] := 6; let TLCT[11,32,3,11] := 9; let TLCT[11,32,4,12] := 6;
let TLCT[11,32,4,13] := 1; let TLCT[11,32,4,14] := 5; let TLCT[11,32,5,15] := 5;

let TLCT[11,32,5,16] := 7; let TLCT[11,32,6,17] := 7; let TLCT[11,32,6,18] := 6; let TLCT[11,32,6,19] := 6; let TLCT[11,32,6,20] := 0; let TLCT[11,32,7,21] := 1; let TLCT[11,32,7,22] := 8; let TLCT[11,32,7,23] := 0; let TLCT[11,32,8,24] := 5; let TLCT[11,32,8,25] := 9; let TLCT[11,32,8,26] := 8; let TLCT[11,32,9,27] := 2; let TLCT[11,32,9,28] := 6; let TLCT[11,32,10,29] := 9;let TLCT[11,32,10,30] := 10;

let TLCT[11,32,11,31] := 4;let TLCT[11,32,11,32] := 0;let TLCT[11,32,11,33] := 6;

let TLCT[11,32,11,34] := 10;let TLCT[11,32,12,35] := 5; let TLCT[11,32,12,36] := 4;

let TLCT[11,32,12,37] := 8;let TLCT[11,32,12,38] := 9;

let TLCT[11,33,1,1] := 7; let TLCT[11,33,1,2] := 1; let TLCT[11,33,1,3] := 6; let TLCT[11,33,1,4] := 7; let TLCT[11,33,2,5] := 10; let TLCT[11,33,2,6] := 9; let TLCT[11,33,2,7] := 7; let TLCT[11,33,2,8] := 0; let TLCT[11,33,3,9] := 5; let TLCT[11,33,3,10] := 5; let TLCT[11,33,3,11] := 0; let TLCT[11,33,4,12] := 9; let TLCT[11,33,4,13] := 7; let TLCT[11,33,4,14] := 6; let TLCT[11,33,5,15] := 2; let TLCT[11,33,5,16] := 6; let TLCT[11,33,6,17] := 8; let TLCT[11,33,6,18] := 10;

let TLCT[11,33,6,19] := 9; let TLCT[11,33,6,20] := 8; let TLCT[11,33,7,21] := 1; let TLCT[11,33,7,22] := 1; let TLCT[11,33,7,23] := 7; let TLCT[11,33,8,24] := 4; let TLCT[11,33,8,25] := 8; let TLCT[11,33,8,26] := 0; let TLCT[11,33,9,27] := 5; let TLCT[11,33,9,28] := 8; let TLCT[11,33,10,29] := 6;let TLCT[11,33,10,30] := 4;

let TLCT[11,33,11,31] := 5;let TLCT[11,33,11,32] := 7;let TLCT[11,33,11,33] := 0;

let TLCT[11,33,11,34] := 3; let TLCT[11,33,12,35] := 1;let TLCT[11,33,12,36] := 5;

let TLCT[11,33,12,37] := 1;let TLCT[11,33,12,38] := 6;

let TLCT[11,34,1,1] := 9; let TLCT[11,34,1,2] := 5; let TLCT[11,34,1,3] := 7;
let TLCT[11,34,1,4] := 1; let TLCT[11,34,2,5] := 3; let TLCT[11,34,2,6] := 4;
let TLCT[11,34,2,7] := 1; let TLCT[11,34,2,8] := 8; let TLCT[11,34,3,9] := 4;
let TLCT[11,34,3,10] := 8; let TLCT[11,34,3,11] := 3; let TLCT[11,34,4,12] := 0;
let TLCT[11,34,4,13] := 3; let TLCT[11,34,4,14] := 3; let TLCT[11,34,5,15] := 9;
let TLCT[11,34,5,16] := 5; let TLCT[11,34,6,17] := 7; let TLCT[11,34,6,18] := 10;

let TLCT[11,34,6,19] := 2; let TLCT[11,34,6,20] := 8; let TLCT[11,34,7,21] := 6;
let TLCT[11,34,7,22] := 7; let TLCT[11,34,7,23] := 9; let TLCT[11,34,8,24] := 9;
let TLCT[11,34,8,25] := 7; let TLCT[11,34,8,26] := 2; let TLCT[11,34,9,27] := 2;
let TLCT[11,34,9,28] := 4; let TLCT[11,34,10,29] := 9;let TLCT[11,34,10,30] := 3;

let TLCT[11,34,11,31] := 8;let TLCT[11,34,11,32] := 3;let TLCT[11,34,11,33] := 4;

let TLCT[11,34,11,34] := 0;let TLCT[11,34,12,35] := 5;let TLCT[11,34,12,36] := 7;

let TLCT[11,34,12,37] := 7;let TLCT[11,34,12,38] := 4;

let TLCT[12,35,1,1] := 9; let TLCT[12,35,1,2] := 1; let TLCT[12,35,1,3] := 6;
let TLCT[12,35,1,4] := 10; let TLCT[12,35,2,5] := 7; let TLCT[12,35,2,6] := 5;
let TLCT[12,35,2,7] := 7; let TLCT[12,35,2,8] := 6; let TLCT[12,35,3,9] := 1;
let TLCT[12,35,3,10] := 9; let TLCT[12,35,3,11] := 4; let TLCT[12,35,4,12] := 8;
let TLCT[12,35,4,13] := 9; let TLCT[12,35,4,14] := 0; let TLCT[12,35,5,15] := 8;
let TLCT[12,35,5,16] := 7; let TLCT[12,35,6,17] := 2; let TLCT[12,35,6,18] := 4;
let TLCT[12,35,6,19] := 6; let TLCT[12,35,6,20] := 1; let TLCT[12,35,7,21] := 8;
let TLCT[12,35,7,22] := 6; let TLCT[12,35,7,23] := 6; let TLCT[12,35,8,24] := 5;
let TLCT[12,35,8,25] := 1; let TLCT[12,35,8,26] := 8; let TLCT[12,35,9,27] := 2;

let TLCT[12,35,9,28] := 2; let TLCT[12,35,10,29] := 3;let TLCT[12,35,10,30] := 5;

let TLCT[12,35,11,31] := 8;let TLCT[12,35,11,32] := 9;let TLCT[12,35,11,33] := 1;

let TLCT[12,35,11,34] := 10;let TLCT[12,35,12,35] := 0;let TLCT[12,35,12,36] := 8;

let TLCT[12,35,12,37] := 1;let TLCT[12,35,12,38] := 9;.

let TLCT[12,36,1,1] := 2; let TLCT[12,36,1,2] := 3; let TLCT[12,36,1,3] := 6;
let TLCT[12,36,1,4] := 8; let TLCT[12,36,2,5] := 1; let TLCT[12,36,2,6] := 4;
let TLCT[12,36,2,7] := 10; let TLCT[12,36,2,8] := 6; let TLCT[12,36,3,9] := 5;
let TLCT[12,36,3,10] := 5; let TLCT[12,36,3,11] := 7; let TLCT[12,36,4,12] := 3;
let TLCT[12,36,4,13] := 3; let TLCT[12,36,4,14] := 3; let TLCT[12,36,5,15] := 0;
let TLCT[12,36,5,16] := 6; let TLCT[12,36,6,17] := 0; let TLCT[12,36,6,18] := 2;
let TLCT[12,36,6,19] := 8; let TLCT[12,36,6,20] := 5; let TLCT[12,36,7,21] := 8;
let TLCT[12,36,7,22] := 10;let TLCT[12,36,7,23] := 6; let TLCT[12,36,8,24] := 3;
let TLCT[12,36,8,25] := 5; let TLCT[12,36,8,26] := 4; let TLCT[12,36,9,27] := 8;
let TLCT[12,36,9,28] := 7; let TLCT[12,36,10,29] := 6;let TLCT[12,36,10,30] := 6;

let TLCT[12,36,11,31] := 2;let TLCT[12,36,11,32] := 8;let TLCT[12,36,11,33] := 3;

let TLCT[12,36,11,34] := 6;let TLCT[12,36,12,35] := 9;let TLCT[12,36,12,36] := 0;

let TLCT[12,36,12,37] := 2;let TLCT[12,36,12,38] := 7;

let TLCT[12,37,1,1] := 2; let TLCT[12,37,1,2] := 9; let TLCT[12,37,1,3] := 6;
let TLCT[12,37,1,4] := 7; let TLCT[12,37,2,5] := 1; let TLCT[12,37,2,6] := 2;
let TLCT[12,37,2,7] := 1; let TLCT[12,37,2,8] := 7; let TLCT[12,37,3,9] := 7;
let TLCT[12,37,3,10] := 2; let TLCT[12,37,3,11] := 1; let TLCT[12,37,4,12] := 1;

let TLCT[12,37,4,13] := 7; let TLCT[12,37,4,14] := 9; let TLCT[12,37,5,15] := 1; let TLCT[12,37,5,16] := 0; let TLCT[12,37,6,17] := 4; let TLCT[12,37,6,18] := 0; let TLCT[12,37,6,19] := 4; let TLCT[12,37,6,20] := 2; let TLCT[12,37,7,21] := 0; let TLCT[12,37,7,22] := 2; let TLCT[12,37,7,23] := 1; let TLCT[12,37,8,24] := 5; let TLCT[12,37,8,25] := 7; let TLCT[12,37,8,26] := 3; let TLCT[12,37,9,27] := 9; let TLCT[12,37,9,28] := 6; let TLCT[12,37,10,29] := 3;let TLCT[12,37,10,30] := 4;

let TLCT[12,37,11,31] := 8;let TLCT[12,37,11,32] := 8;let TLCT[12,37,11,33] := 6;

let TLCT[12,37,11,34] := 2; let TLCT[12,37,12,35] := 4;let TLCT[12,37,12,36] := 3;

let TLCT[12,37,12,37] := 0;let TLCT[12,37,12,38] := 3;

let TLCT[12,38,1,1] := 10; let TLCT[12,38,1,2] := 10; let TLCT[12,38,1,3] := 3; let TLCT[12,38,1,4] := 5; let TLCT[12,38,2,5] := 4; let TLCT[12,38,2,6] := 1; let TLCT[12,38,2,7] := 7; let TLCT[12,38,2,8] := 1; let TLCT[12,38,3,9] := 3; let TLCT[12,38,3,10] := 3; let TLCT[12,38,3,11] := 4; let TLCT[12,38,4,12] := 5; let TLCT[12,38,4,13] := 3; let TLCT[12,38,4,14] := 8; let TLCT[12,38,5,15] := 1; let TLCT[12,38,5,16] := 5; let TLCT[12,38,6,17] := 10;let TLCT[12,38,6,18] := 3; let TLCT[12,38,6,19] := 0; let TLCT[12,38,6,20] := 6; let TLCT[12,38,7,21] := 7; let TLCT[12,38,7,22] := 0; let TLCT[12,38,7,23] := 9; let TLCT[12,38,8,24] := 6; let TLCT[12,38,8,25] := 3; let TLCT[12,38,8,26] := 4; let TLCT[12,38,9,27] := 6; let TLCT[12,38,9,28] := 10;let TLCT[12,38,10,29] := 5;let TLCT[12,38,10,30] := 7;

let TLCT[12,38,11,31] := 0;let TLCT[12,38,11,32] := 5;let TLCT[12,38,11,33] := 2;

let TLCT[12,38,11,34] := 10;let TLCT[12,38,12,35] := 1;let TLCT[12,38,12,36] := 8;

let TLCT[12,38,12,37] := 3;let TLCT[12,38,12,38] := 0;

APPENDIX B: C++ PROGRAM FOR SA ALGORITHM: CASE STUDY 1

Header file

Format: .h

```
#include <iostream>
#include <conio.h>
#include <random>
#include <time.h>
#include <cmath>

using namespace std;

class Operation
{
public:
	int R;
	int D;
	int F;
	int P;
	float TO;
	float RT;
	int I;
	int E;
	int G;
};

extern Operation O[38];
extern Operation BG[38];
extern float Face[5][5];
extern float Tool[38][38];
extern int Q[38];
extern int BS[38];
extern int Nbg[4];
extern int H[12];
extern float Ebg;
extern float Ebs;
extern float t;
```

```
extern float takt;
extern float T[12];
extern int counter1;
extern int counter2;
extern int A;
extern int NM;
extern int totaloper;
float grouping_energy();
float sequencing_energy();
void display_result();
void best_grouping_solution (float, int []);
void best_sequencing_solution (float);
void declare();
int randint(int, int);
```

Source file 1

Format: .cpp

```
#include "data.h"

Operation O[38];
Operation BG[38];
float Face[5][5] =
{{0,1.95,2.5,4,5},{1.95,0,1.95,2.5,4},{2.5,1.95,0,1.95,2.5},{4,2.5,1.95,0,1.95},{5
,4,2.5,1.95,0}};
float Tool[38][38] =
{{0,10,6,1,0,9,1,8,0,4,4,5,2,8,9,6,3,5,2,1,5,7,8,7,6,10,2,9,0,6,2,2,10,3,4,10,10,7
},{8,0,2,1,6,0,4,1,3,9,2,2,4,8,3,6,4,6,2,4,6,6,1,3,5,8,9,6,2,0,8,2,7,2,5,5,9,5},{6
,9,0,10,2,1,6,3,1,9,3,6,2,6,6,7,10,5,7,1,9,9,7,0,5,2,0,7,10,3,6,5,4,1,7,6,5,7},{9,
8,7,0,10,4,0,9,8,0,3,4,6,6,9,6,1,1,4,1,10,4,9,7,0,4,2,0,10,8,8,1,6,8,7,9,4,6},{0,1
,6,1,0,9,8,8,0,1,5,10,10,2,2,4,8,6,3,2,2,9,10,5,8,3,5,2,0,2,2,7,1,8,5,10,4,4},{9,0
,8,1,5,0,5,1,2,4,7,1,1,10,10,8,5,7,7,6,8,6,10,1,1,4,3,10,4,0,7,3,3,7,8,7,10,8},{4,
1,5,0,8,5,0,7,8,0,8,5,6,7,1,1,4,8,10,10,7,7,2,7,0,6,1,0,5,6,8,6,5,3,7,9,9,10},{4,1
,10,5,1,4,2,0,7,4,0,9,2,5,9,8,6,8,9,7,5,10,0,5,2,0,10,3,7,10,4,1,0,6,1,7,2,1},{0,2
,8,4,0,2,7,3,0,3,5,2,9,10,7,7,5,4,4,7,6,7,4,8,9,9,8,6,0,6,4,10,6,1,4,2,2,7},{1,9,1
0,0,2,1,0,6,10,0,1,5,1,10,4,10,8,8,7,2,7,5,8,3,0,9,10,0,5,2,6,7,9,2,10,5,10,1},{7,
2,2,6,8,7,9,0,9,10,0,1,4,8,9,4,8,1,3,1,4,8,10,2,1,0,7,7,7,8,7,10,0,7,9,1,3,10},{6,
6,1,1,1,5,8,8,2,3,4,0,10,4,6,2,2,8,1,5,6,3,7,3,2,2,2,1,2,6,10,5,7,0,10,2,3,5},{3,3
,1,7,2,6,4,7,5,5,6,8,0,5,9,3,4,2,2,1,9,6,3,9,5,5,9,5,3,2,2,2,5,7,6,9,1,6},{7,2,1,6
```

```
,7,8,6,3,7,10,4,9,2,0,2,7,6,1,5,3,6,5,9,9,7,4,4,3,9,1,6,7,1,9,0,10,3,5},{10,7,6,6,
7,5,4,6,6,8,8,1,6,3,0,3,0,9,7,10,4,2,6,5,9,1,9,6,9,8,2,7,4,2,10,0,3,1},{4,9,3,3,5,
1,4,8,7,10,1,8,1,4,10,0,2,0,3,10,0,2,10,9,5,2,2,8,5,8,1,5,9,1,6,2,0,2},{4,6,10,8,1
,3,1,4,2,10,2,5,1,4,0,6,0,4,10,1,5,3,5,3,6,5,1,9,6,4,5,6,6,9,9,0,7,8},{2,3,3,8,8,9
,5,5,8,4,7,2,9,10,5,0,8,0,6,5,0,3,6,10,5,4,9,9,10,1,3,3,7,9,9,10,0,3},{10,9,5,9,5,
2,6,8,4,3,5,5,2,3,10,1,5,7,0,6,1,0,8,2,1,6,9,8,4,3,0,3,7,9,3,1,4,0},{3,10,5,7,4,7,
3,5,6,1,4,2,2,9,5,6,7,10,8,0,9,3,0,7,3,4,10,2,9,3,8,0,10,10,8,9,10,5},{1,4,7,10,10
,5,6,9,4,9,9,7,4,8,10,0,10,0,7,4,0,2,1,10,9,4,2,9,9,7,3,7,6,7,7,4,0,4},{3,1,10,6,6
,8,5,2,4,2,9,2,10,7,4,10,10,9,0,4,7,0,5,5,10,1,6,7,4,4,0,9,5,6,7,5,5,0},{2,8,10,10
,9,3,10,6,8,7,7,10,8,6,5,2,1,3,9,0,7,6,0,5,8,8,1,8,5,1,6,0,5,4,3,9,1,1},{5,7,0,10,
10,8,9,9,4,1,2,4,3,1,10,9,8,3,1,7,4,10,7,0,10,3,0,2,1,4,8,9,2,1,2,8,9,3},{10,8,3,0
,9,3,0,6,2,0,4,6,3,6,8,5,1,3,2,5,2,4,10,1,0,9,5,0,9,8,1,6,5,6,1,9,1,4},{7,8,7,7,9,
7,3,0,1,8,0,9,6,10,4,7,10,10,5,4,2,9,7,9,6,0,10,9,5,8,1,1,0,7,4,1,4,4},{6,2,0,1,6,
1,7,2,6,3,1,4,7,4,9,4,8,2,6,4,10,10,9,0,4,5,0,2,3,4,7,8,6,9,8,7,8,6},{9,4,1,0,6,2,
0,4,6,0,4,2,6,9,9,3,2,8,3,4,6,3,7,3,0,4,2,0,7,6,5,3,8,2,10,6,1,3},{0,8,9,3,0,4,2,1
0,0,7,2,3,7,7,7,5,6,1,2,4,5,4,6,9,5,8,3,4,0,9,4,1,3,5,7,3,9,2},{8,0,1,8,6,0,4,2,4,
1,6,7,1,3,7,7,3,8,10,6,3,2,4,6,7,10,2,9,10,0,3,9,4,6,6,3,10,2},{4,5,8,3,10,1,10,1,
2,10,9,7,3,2,3,8,9,2,0,4,10,0,3,7,4,10,5,10,5,8,0,1,6,2,10,3,1,0},{6,7,9,8,7,4,7,8
,4,6,9,6,1,5,5,7,7,6,6,0,1,8,0,5,9,8,2,6,9,10,4,0,6,10,5,4,8,9},{7,1,6,7,10,9,7,0,
5,5,0,9,7,6,2,6,8,10,9,8,1,1,7,4,8,0,5,8,6,4,5,7,0,3,1,5,1,6},{9,5,7,1,3,4,1,8,4,8
,3,0,3,3,9,5,7,10,2,8,6,7,9,9,7,2,2,4,9,3,8,3,4,0,5,7,7,4},{9,1,6,10,7,5,7,6,1,9,4
,8,9,0,8,7,2,4,6,1,8,6,6,5,1,8,2,2,3,5,8,9,1,10,0,8,1,9},{2,3,6,8,1,4,10,6,5,5,7,3
,3,3,0,6,0,2,8,5,8,10,6,3,5,4,8,7,6,6,2,8,3,6,9,0,2,7},{2,9,6,7,1,2,1,7,7,2,1,1,7,
9,1,0,4,0,4,2,0,2,1,5,7,3,9,6,3,4,8,8,6,2,4,3,0,3},{10,10,3,5,4,1,7,1,3,3,4,5,3,8,
1,5,10,3,0,6,7,0,9,6,3,4,6,10,5,7,0,5,2,10,1,8,3,0}};
int Q[38] =
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
int BS[38] =
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
int Nbg[4] = {1,1,1,1};
int H[12] = {4,2,2,2,2,1,1,1,1,1,3,3};
int A = 75;
int NM = 5;
int totaloper = 38;
float Ebg = 1600;
float Ebs = 500;
float t = 5;
float takt = 400;
float T[12] = {2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000};
```

```
float grouping_energy()
{

	float En=0; float Eng[4]={0,0,0,0}; int Hg[4]={0,0,0,0}; int decision=0;
int N[4] = {1,1,1,1};
	//ORCT
	int F1=0; int F2=0;
	for (int i=1; i<=4; i++)
	{
		for(int j=1; j<=totaloper-1; j++)
		{
			if(O[j].G==i)
				{
					F1=O[j].F;
					for(int k=j+1; k<=totaloper; k++)
					{if(O[k].G==i) {F2=O[k].F; Eng[i-1] = Eng[i-1] +
Face[F1-1][F2-1]; break;}}

				}
		}
	}
	for(int i=1; i<=4;i++)
	{
		En = En + Eng[i-1];
	}
	//TLCT
	int T1=0; int T2=0;
	for (int i=1; i<=4; i++)
	{
		for(int j=1; j<=totaloper-1; j++)
		{
			if(O[j].G==i)
				{
					T1=j;
					for(int k=j+1; k<=totaloper; k++)
					{if(O[k].G==i) {T2=k; Eng[i-1] = Eng[i-1] +
Tool[T1-1][T2-1]; break;}}

				}
```

```
		}
	}
	//OPT
	for (int i=1; i<=4; i++)
	{
		for(int j=1; j<=totaloper; j++)
		{
			if(O[j].G==i)
				{
					Eng[i-1] = Eng[i-1] + O[j].D *
(O[j].TO+O[j].RT);
				}
		}
	}
	//Tool life
	for (int i=1; i<=4; i++)
	{
		int Y[12]={0,0,0,0,0,0,0,0,0,0,0,0};
		float TO[12]={0,0,0,0,0,0,0,0,0,0,0,0};
		for(int j=1; j<=totaloper; j++)
		{
			if(O[j].G==i)
				{
					Y[O[j].P-1]=1;
					TO[O[j].P-1]=TO[O[j].P-1] + O[j].TO*O[j].D;
				}
		}
		for(int k=1; k<=12; k++)
		{
			if(Y[k-1]==1)
				{
					Hg[i-1] = Hg[i-1] + H[k-1];
				}
			if(TO[k-1]>T[k-1])
			{
				decision=1;
			}
		}
```

```
	}
	//Inclusion and exclusion

	for(int i=1; i<=totaloper; i++)
	{
		for (int j=1; j<=totaloper; j++)
		{
			if(O[i].I==O[j].R&&O[i].G!=O[j].G) {decision=1;}
			if(O[i].E==O[j].R&&O[i].G==O[j].G) {decision=1;}
		}
	}
	//No. of machines
	for (int i=1; i<=4; i++)
	{
		if(Eng[i-1]>(takt*N[i-1])&&Eng[i-1]<2*(takt*N[i-1]))
			N[i-1]++;
		else if(Eng[i-1]>2*(takt*N[i-1]))
			N[i-1] = N[i-1] + 2;
		else if (((takt*N[i-1])-Eng[i-1])>takt)
			N[i-1]--;
	}
	//TT
	for (int i=1; i<=4; i++)
	{
		Eng[i-1] = Eng[i-1] + (N[i-1]-1)*t;
		En = En + (N[i-1]-1)*t;
	}
	//Takt time, Tool magazine limit and No. of machines
	for (int i=1; i<=4; i++)
	{
		if(Eng[i-1]>(takt*N[i-1])||Hg[i-1]>(A*N[i-1])||N[i-1]>NM)
			{
				decision=1;
			}
	}

	if (decision==0) {best_grouping_solution(En,N); return En;}
	else return 1;
}
```

```
float sequencing_energy()
{
	float En=0;

	for(int i=1; i<=4; i++)
	{
		for (int j=1; j<=totaloper; j++)
		{
			if(BG[Q[j-1]].G==i)
				{   int F1=0; int F2=0; int T1=0; int T2=0;

					for(int k=j+1; k<=totaloper; k++)
					{if(BG[Q[k-1]].G==i)
						{
							F1=O[Q[j-1]].F; F2=O[Q[k-1]].F; En
= En + Face[F1-1][F2-1];
							T1=Q[j-1]; T2=Q[k-1]; En = En +
Tool[T1-1][T2-1]; break;
						}
					}

				}
		}
	}

	best_sequencing_solution(En);
	return En;
}

void display_result()
{
	float ORCT[4]={0,0,0,0}; float TLCT[4]={0,0,0,0}; float OPT[4]={0,0,0,0};
	for(int i=1; i<=4; i++)
	{
		cout << endl << " Workstation No= " << i << endl;
		for (int j=1; j<=totaloper; j++)
		{
			if(BG[BS[j-1]].G==i)
```

```
                        {   cout << "[" << O[BS[j-1]].R << ", " << BS[j-1] <<
"] ";
                                OPT[i-1] = OPT[i-1] + O[BS[j-1]].D * (O[BS[j-
1]].TO+O[BS[j-1]].RT);
                                int F1=0; int F2=0; int T1=0; int T2=0;

                                for(int k=j+1; k<=totaloper; k++)
                                {if(BG[BS[k-1]].G==i)
                                        {
                                                F1=O[BS[j-1]].F; F2=O[BS[k-1]].F;
ORCT[i-1] = ORCT[i-1] + Face[F1-1][F2-1];
                                                T1=BS[j-1]; T2=BS[k-1]; TLCT[i-1]
= TLCT[i-1] + Tool[T1-1][T2-1]; break;
                                        }
                                }

                        }
                }
                cout << endl << "N= " << Nbg[i-1] << " ORCT= " << ORCT[i-1] << "
TLCT= " << TLCT[i-1] << "  OPT= " << OPT[i-1] << " TT= " << (Nbg[i-1]-1)*t << "
Idle time= " << Nbg[i-1]*takt-ORCT[i-1]-TLCT[i-1]-OPT[i-1]-(Nbg[i-1]-1)*t << endl;
        }

        cout << endl << "Takt time= " << takt << " Total ORCT= " <<
ORCT[0]+ORCT[1]+ORCT[2]+ORCT[3] << " Total TLCT= " <<
TLCT[0]+TLCT[1]+TLCT[2]+TLCT[3] << " Total OPT= " << OPT[0]+OPT[1]+OPT[2]+OPT[3]
<< " Total TT= " << (Nbg[0]+Nbg[1]+Nbg[2]+Nbg[3]-4)*t << " Makespan= " <<
ORCT[0]+ORCT[1]+ORCT[2]+ORCT[3]+TLCT[0]+TLCT[1]+TLCT[2]+TLCT[3]+OPT[0]+OPT[1]+OPT[
2]+OPT[3]+(Nbg[0]+Nbg[1]+Nbg[2]+Nbg[3]-4)*t << endl;
}


void best_grouping_solution (float Enew, int N[])
{
        if (Enew < Ebg)
        {
                Ebg=Enew;
                for(int i=1; i<=totaloper; i++)
                {
```

```
                        BG[i].G=O[i].G;
                        BG[i].F=O[i].F;
                }
                for(int i=1; i<=4; i++)
                {
                        Nbg[i-1]=N[i-1];
                }
        }

}

void best_sequencing_solution (float Enew)
{
        if (Enew < Ebs)
        {
                Ebs=Enew;
                for(int i=1; i<=totaloper; i++)
                {
                        BS[i-1]=Q[i-1];
                }
        }

}

int randint(int floor, int ceiling)
{
                int range = ceiling+1-floor;
                int num = floor + int ((range*rand()) / (RAND_MAX+1.0));
                return num;
}

void declare()
{
O[1].R=1;    O[1].D=16;   O[1].F=1;   O[1].P=1;   O[1].TO= 2.88;   O[1].RT= 1.73;
O[1].I=0;   O[1].E=6;
O[2].R=1;    O[2].D=16;   O[2].F=1;   O[2].P=2;   O[2].TO= 1.25;   O[2].RT= 0.75;
O[2].I=0;   O[2].E=6;
O[3].R=1;    O[3].D=16;   O[3].F=1;   O[3].P=3;   O[3].TO= 1.71;   O[3].RT= 1.03;
O[3].I=0;   O[3].E=6;
```

```
O[4].R=1;    O[4].D=16;   O[4].F=1;   O[4].P=4;   O[4].TO= 2.93;   O[4].RT= 1.80;
O[4].I=0;   O[4].E=6;
O[5].R=2;    O[5].D=4;    O[5].F=1;   O[5].P=1;   O[5].TO= 1.54;   O[5].RT= 0.92;
O[5].I=0;   O[5].E=0;
O[6].R=2;    O[6].D=4;    O[6].F=1;   O[6].P=2;   O[6].TO= 1.57;   O[6].RT= 0.94;
O[6].I=0;   O[6].E=0;
O[7].R=2;    O[7].D=4;    O[7].F=1;   O[7].P=4;   O[7].TO= 2.62;   O[7].RT= 1.57;
O[7].I=0;   O[7].E=0;
O[8].R=2;    O[8].D=4;    O[8].F=1;   O[8].P=5;   O[8].TO= 1.70;   O[8].RT= 1.00;
O[8].I=0;   O[8].E=0;
O[9].R=3;    O[9].D=8;    O[9].F=1;   O[9].P=1;   O[9].TO= 2.75;   O[9].RT= 1.65;
O[9].I=0;   O[9].E=0;
O[10].R=3;   O[10].D=8;   O[10].F=1;  O[10].P=4;  O[10].TO= 1.79;  O[10].RT= 1.07;
O[10].I=0;  O[10].E=0;
O[11].R=3;   O[11].D=8;   O[11].F=1;  O[11].P=5;  O[11].TO= 2.89;  O[11].RT= 1.73;
O[11].I=0;  O[11].E=0;
O[12].R=4;   O[12].D=8;   O[12].F=2;  O[12].P=6;  O[12].TO= 1.95;  O[12].RT= 1.17;
O[12].I=5;  O[12].E=0;
O[13].R=4;   O[13].D=8;   O[13].F=2;  O[13].P=7;  O[13].TO= 1.20;  O[13].RT= 0.75;
O[13].I=5;  O[13].E=0;
O[14].R=4;   O[14].D=8;   O[14].F=2;  O[14].P=8;  O[14].TO= 1.65;  O[14].RT= 0.99;
O[14].I=5;  O[14].E=0;
O[15].R=5;   O[15].D=4;   O[15].F=2;  O[15].P=9;  O[15].TO= 2.68;  O[15].RT= 1.61;
O[15].I=0;  O[15].E=0;
O[16].R=5;   O[16].D=4;   O[16].F=2;  O[16].P=10; O[16].TO= 2.30;  O[16].RT= 1.38;
O[16].I=0;  O[16].E=0;
O[17].R=6;   O[17].D=4;   O[17].F=3;  O[17].P=9;  O[17].TO= 1.97;  O[17].RT= 1.18;
O[17].I=8;  O[17].E=0;
O[18].R=6;   O[18].D=4;   O[18].F=3;  O[18].P=10; O[18].TO= 3.37;  O[18].RT= 2.02;
O[18].I=8;  O[18].E=0;
O[19].R=6;   O[19].D=4;   O[19].F=3;  O[19].P=11; O[19].TO= 1.36;  O[19].RT= 0.82;
O[19].I=8;  O[19].E=0;
O[20].R=6;   O[20].D=4;   O[20].F=3;  O[20].P=12; O[20].TO= 2.74;  O[20].RT= 1.40;
O[20].I=8;  O[20].E=0;
O[21].R=7;   O[21].D=4;   O[21].F=3;  O[21].P=10; O[21].TO= 1.35;  O[21].RT= 0.81;
O[21].I=0;  O[21].E=12;
O[22].R=7;   O[22].D=4;   O[22].F=3;  O[22].P=11; O[22].TO= 1.22;  O[22].RT= 0.73;
O[22].I=0;  O[22].E=12;
```

```
O[23].R=7;   O[23].D=4;   O[23].F=3;  O[23].P=12; O[23].TO= 3.09;  O[23].RT= 1.85;
O[23].I=0;  O[23].E=12;
O[24].R=8;   O[24].D=20;  O[24].F=4;  O[24].P=3;  O[24].TO= 1.20;  O[24].RT= 0.72;
O[24].I=0;  O[24].E=0;
O[25].R=8;   O[25].D=20;  O[25].F=4;  O[25].P=4;  O[25].TO= 1.62;  O[25].RT= 0.97;
O[25].I=0;  O[25].E=0;
O[26].R=8;   O[26].D=20;  O[26].F=4;  O[26].P=5;  O[26].TO= 2.06;  O[26].RT= 1.24;
O[26].I=0;  O[26].E=0;
O[27].R=9;   O[27].D=20;  O[27].F=3;  O[27].P=3;  O[27].TO= 1.90;  O[27].RT= 1.14;
O[27].I=0;  O[27].E=0;
O[28].R=9;   O[28].D=20;  O[28].F=3;  O[28].P=4;  O[28].TO= 3.30;  O[28].RT= 1.98;
O[28].I=0;  O[28].E=0;
O[29].R=10;  O[29].D=8;   O[29].F=5;  O[29].P=1;  O[29].TO= 2.78;  O[29].RT= 1.67;
O[29].I=11; O[29].E=0;
O[30].R=10;  O[30].D=8;   O[30].F=5;  O[30].P=2;  O[30].TO= 1.31;  O[30].RT= 0.79;
O[30].I=11; O[30].E=0;
O[31].R=11;  O[31].D=8;   O[31].F=5;  O[31].P=11; O[31].TO= 3.43;  O[31].RT= 2.05;
O[31].I=0;  O[31].E=0;
O[32].R=11;  O[32].D=8;   O[32].F=5;  O[32].P=12; O[32].TO= 3.42;  O[32].RT= 2.05;
O[32].I=0;  O[32].E=0;
O[33].R=11;  O[33].D=8;   O[33].F=5;  O[33].P=5;  O[33].TO= 3.39;  O[33].RT= 2.03;
O[33].I=0;  O[33].E=0;
O[34].R=11;  O[34].D=8;   O[34].F=5;  O[34].P=6;  O[34].TO= 2.50;  O[34].RT= 1.50;
O[34].I=0;  O[34].E=0;
O[35].R=12;  O[35].D=4;   O[35].F=5;  O[35].P=8;  O[35].TO= 2.64;  O[35].RT= 1.58;
O[35].I=0;  O[35].E=0;
O[36].R=12;  O[36].D=4;   O[36].F=5;  O[36].P=9;  O[36].TO= 2.26;  O[36].RT= 1.36;
O[36].I=0;  O[36].E=0;
O[37].R=12;  O[37].D=4;   O[37].F=5;  O[37].P=10; O[37].TO= 2.45;  O[37].RT= 1.47;
O[37].I=0;  O[37].E=0;
O[38].R=12;  O[38].D=4;   O[38].F=5;  O[38].P=11; O[38].TO= 2.32;  O[38].RT= 1.40;
O[38].I=0;  O[38].E=0;

//Generate grouping

       int steps=12/4; int index_step=0; int decision=0;
       int R1[3]={-1,-1,-1}, R2[3]={-1,-1,-1}, R3[4]={-1,-1,-1,-1}, R4[3]={-1,-1,-
1};
       int R=0; int I=0;
```

```
for(int i=1; i<=totaloper; i++)
{
	O[i].G=0;
}

//G1
//String
for(int i=1; i<=totaloper; i++)
{
	if(O[i].I!=0)
	{
		for(int j=1; j<=totaloper; j++)
		{
			if(O[j].R==O[i].R||O[j].I==O[i].R||O[j].R==O[i].I)
			{
				O[j].G=1;
				if(O[j].R!=R1[0]&&O[j].R!=R1[1]&&O[j].R!=R1[2])
				{R1[index_step]=O[j].R; index_step++;}
			}

		}
	break;
	}
}
//Filler
for (int i=1; i<=totaloper; i++)
	{
		if(O[i].I==0&&index_step<steps)
		{
			decision=0;
					for(int j=1; j<=totaloper; j++)
					{
						for(int k=0; k<=index_step; k++)

						{

if((O[j].E==O[i].R)&&(O[j].R==R1[k])) decision=1;
```

```
                                        }

                                }
                                if (decision==0)
                                        {
                                                for(int k=1; k<=totaloper;
k++)
                                                {
                                                        if(O[k].R==O[i].R)
O[k].G=1;
                                                }

        if(O[i].R!=R1[0]&&O[i].R!=R1[1]&&O[i].R!=R1[2])
                                                {R1[index_step]=O[i].R;
index_step++;}
                                        }
                                else
if(O[i].I==R1[0]||O[i].I==R1[1]||O[i].I==R1[2])
                                        {
                                                for(int k=1; k<=totaloper;
k++)
                                                {
                                                        if(O[k].R==O[i].R)
O[k].G=1;
                                                }

        if(O[i].R!=R1[0]&&O[i].R!=R1[1]&&O[i].R!=R1[2])
                                                {R1[index_step]=O[i].R;
index_step++;}
                                        }
                                }
                        }

        //G2
        //String
        index_step=0;
        for (int i=1; i<=totaloper; i++)
        {
                if(O[i].I!=0&&O[i].G==0)
```

```
		{
			for(int j=1; j<=totaloper; j++)
			{
				if(O[j].R==O[i].R||O[j].I==O[i].R||O[j].R==O[i].I)
				{
					O[j].G=2;
					if(O[j].R!=R2[0]&&O[j].R!=R2[1]&&O[j].R!=R2[2])
					{R2[index_step]=O[j].R; index_step++;}

				}
			}
		break;
		}
	}
	//Filler
	for(int i=1; i<=totaloper; i++)
	{
			if(O[i].I==0&&O[i].G==0&&index_step<steps)
			{
				decision=0;
					for(int j=1; j<=totaloper; j++)
					{
						for(int k=0; k<=index_step; k++)

						{

	if((O[j].E==O[i].R)&&(O[j].R==R2[k])) decision=1;
						}

					}
					if (decision==0)
						{
							for(int k=1; k<=totaloper;
k++)

							{
								if(O[k].R==O[i].R)
O[k].G=2;

							}
```

```
        if(O[i].R!=R2[0]&&O[i].R!=R2[1]&&O[i].R!=R2[2])
                                                        {R2[index_step]=O[i].R;
index_step++;}
                                                }
                                            else
if(O[i].I==R2[0]||O[i].I==R2[1]||O[i].I==R2[2])
                                                {
                                                        for(int k=1; k<=totaloper;
k++)
                                                        {
                                                                if(O[k].R==O[i].R)
O[k].G=2;
                                                        }

        if(O[i].R!=R2[0]&&O[i].R!=R2[1]&&O[i].R!=R2[2])
                                                        {R2[index_step]=O[i].R;
index_step++;}
                                                }
                                }
        }

        //G3
        //String
        index_step=0;
        for (int i=1; i<=totaloper; i++)
        {
                if(O[i].I!=0&&O[i].G==0)
                {
                        for(int j=1; j<=totaloper; j++)
                        {
                                if(O[j].R==O[i].R||O[j].I==O[i].R||O[j].R==O[i].I)
                                {
                                        O[j].G=3;
                                        if(O[j].R!=R3[0]&&O[j].R!=R3[1]&&O[j].R!=R3[2])
                                        {R3[index_step]=O[j].R; index_step++;}
                                }
                        }
                break;
```

```
		}
	}
	//Filler
	for (int i=1; i<=totaloper; i++)
		{
			if(O[i].I==0&&O[i].G==0&&index_step<steps+1)
			{
				decision=0;
					for(int j=1; j<=totaloper; j++)
					{
						for(int k=0; k<=index_step; k++)

						{

	if((O[j].E==O[i].R)&&(O[j].R==R3[k])) decision=1;
						}

					}
					if (decision==0)
						{
							for(int k=1; k<=totaloper;
k++)
							{
								if(O[k].R==O[i].R)
O[k].G=3;
							}

	if(O[i].R!=R3[0]&&O[i].R!=R3[1]&&O[i].R!=R3[2])
							{R3[index_step]=O[i].R;
index_step++;}
						}
					else
if(O[i].I==R3[0]||O[i].I==R3[1]||O[i].I==R3[2])
						{
							for(int k=1; k<=totaloper;
k++)
							{
								if(O[k].R==O[i].R)
O[k].G=3;
```

```
                                                        }

        if(O[i].R!=R3[0]&&O[i].R!=R3[1]&&O[i].R!=R3[2])
                                                        {R3[index_step]=O[i].R;
index_step++;}
                                                    }
                                }
                        }

        //G4
        //String
        index_step=0;
        for (int i=1; i<=totaloper; i++)
        {
                if(O[i].I!=0&&O[i].G==0)
                {
                        for(int j=1; j<=totaloper; j++)
                        {
                                if(O[j].R==O[i].R||O[j].I==O[i].R||O[j].R==O[i].I)
                                {
                                        O[j].G=4;
                                        if(O[j].R!=R4[0]&&O[j].R!=R4[1]&&O[j].R!=R4[2])
                                        {R4[index_step]=O[j].R; index_step++;}
                                }
                        }
                break;
                }
        }
        //Filler
        for (int i=1; i<=totaloper; i++)
                {
                        if(O[i].I==0&&O[i].G==0&&index_step<steps)
                        {
                                decision=0;
                                        for(int j=1; j<=totaloper; j++)
                                        {
                                                for(int k=0; k<=index_step; k++)

                                                {
```

```
	if((O[j].E==O[i].R)&&(O[j].R==R4[k])) decision=1;
                                                }

                                        }
                                        if (decision==0)
                                                {
                                                        for(int k=1; k<=totaloper;
k++)
                                                        {
                                                                if(O[k].R==O[i].R)
O[k].G=4;
                                                        }

	if(O[i].R!=R4[0]&&O[i].R!=R4[1]&&O[i].R!=R4[2])
                                                        {R4[index_step]=O[i].R;
index_step++;}
                                                }
                                        else
if(O[i].I==R4[0]||O[i].I==R4[1]||O[i].I==R4[2])
                                                {
                                                        for(int k=1; k<=totaloper;
k++)
                                                        {
                                                                if(O[k].R==O[i].R)
O[k].G=4;
                                                        }

	if(O[i].R!=R4[0]&&O[i].R!=R4[1]&&O[i].R!=R4[2])
                                                        {R4[index_step]=O[i].R;
index_step++;}
                                                }
                                }
                        }


	for(int i=1; i<=totaloper; i++)
	{
```

```
            cout << " Oper no = " << i << " R= " << O[i].R << " Group = " <<
O[i].G << endl;
       }
}
```

Source file 2

Format: .cpp

```
#include "data.h"
void main()
{
       clock_t start_time=clock();

       declare();

       double e = 2.718281828;
       double lambda = 0.99;

       float Enew; float Ecs=0;
       int counter1=0; int counter2=0;
       int x; int x1; int x2; int x3;
       int temp1=0; int temp2=0; int temp3=0; float decision=0;

       int R[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

       float Ec = Enew = grouping_energy();
       for (double T=1000000; T>=1; T *= lambda)
                     {
                            for (int n=1; n<=50; n++)
                            {

                                          temp1=0; temp2=0; temp3=0; decision=0;

                                          x=randint(1,2);
                                          switch(x)
                                          {
                                          case 1:
```

```
x1=randint(1,4);
switch(x1)
{
case 1:
       for(int i=1; i<=totaloper;
i++)
       {
              if(O[i].G==2)
              { temp1=O[i].R;
break;}
       }
       for(int i=1; i<=totaloper;
i++)
       {
       if(O[i].G==2&&O[i].R==temp1)
                     O[i].G=1;
       }
       for(int i=1; i<=totaloper;
i++)
       {
       if(O[i].G==1&&O[i].R>temp2)
                     temp2=O[i].R;
       }
       for(int i=1; i<=totaloper;
i++)
       {
       if(O[i].G==1&&O[i].R==temp2)
                     O[i].G=2;
       }
       decision =
grouping_energy();
       if (decision>1)
Enew=decision;
       else if (decision==1)
```

```
                                                        {
                                                               for(int i=1;
i<=totaloper; i++)
                                                               {
       if(O[i].G==1&&O[i].R==temp1)
       O[i].G=2;
                                                               }
                                                               for(int i=1;
i<=totaloper; i++)
                                                               {
       if(O[i].G==2&&O[i].R==temp2)
       O[i].G=1;
                                                               }
                                                        }
                                                        break;
                                                 case 2:
                                                        for(int i=1; i<=totaloper;
i++)
                                                        {
                                                               if(O[i].G==3)
                                                               { temp1=O[i].R;
break;}
                                                        }
                                                        for(int i=1; i<=totaloper;
i++)
                                                        {
       if(O[i].G==3&&O[i].R==temp1)
                                                                      O[i].G=2;
                                                        }
                                                        for(int i=1; i<=totaloper;
i++)
                                                        {
```

```
        if(O[i].G==2&&O[i].R>temp2)
                                                        temp2=O[i].R;
                                                }
                                                for(int i=1; i<=totaloper;
i++)
                                                {

        if(O[i].G==2&&O[i].R==temp2)
                                                        O[i].G=3;
                                                }
                                                decision =
grouping_energy();
                                                if (decision>1)
Enew=decision;
                                                else if (decision==1)
                                                {
                                                        for(int i=1;
i<=totaloper; i++)
                                                        {

        if(O[i].G==2&&O[i].R==temp1)

        O[i].G=3;
                                                        }
                                                        for(int i=1;
i<=totaloper; i++)
                                                        {

        if(O[i].G==3&&O[i].R==temp2)

        O[i].G=2;
                                                        }
                                                }
                                                break;
                                        case 3:
                                                for(int i=1; i<=totaloper;
i++)
                                                {
```

```
break;}

i++)

        if(O[i].G==4&&O[i].R==temp1)

i++)

        if(O[i].G==3&&O[i].R>temp2)

i++)

        if(O[i].G==3&&O[i].R==temp2)

grouping_energy();

Enew=decision;

i<=totaloper; i++)

        if(O[i].G==3&&O[i].R==temp1)

        O[i].G=4;

        if(O[i].G==4)
        { temp1=O[i].R;
}
for(int i=1; i<=totaloper;

{

                O[i].G=3;
}

for(int i=1; i<=totaloper;

{

                temp2=O[i].R;
}
for(int i=1; i<=totaloper;

{

                O[i].G=4;
}
decision =

if (decision>1)

else if (decision==1)
{
        for(int i=1;

        {
```

```
                                                        }
                                                        for(int i=1;
i<=totaloper; i++)
                                                        {
        if(O[i].G==4&&O[i].R==temp2)
        O[i].G=3;
                                                        }
                                                }
                                                break;
                                        case 4:
                                                for(int i=1; i<=totaloper;
i++)
                                                {
                                                        if(O[i].G==1)
                                                        { temp1=O[i].R;
break;}
                                                }
                                                for(int i=1; i<=totaloper;
i++)
                                                {
        if(O[i].G==1&&O[i].R==temp1)
                                                                O[i].G=4;
                                                }
                                                for(int i=1; i<=totaloper;
i++)
                                                {
        if(O[i].G==4&&O[i].R>temp2)
                                                                temp2=O[i].R;
                                                }
                                                for(int i=1; i<=totaloper;
i++)
                                                {
        if(O[i].G==4&&O[i].R==temp2)
```

```
                                                                O[i].G=1;
                                                        }
                                                        decision =
grouping_energy();
                                                        if (decision>1)
Enew=decision;
                                                        else if (decision==1)
                                                        {
                                                                for(int i=1;
i<=totaloper; i++)
                                                                {

        if(O[i].G==4&&O[i].R==temp1)

        O[i].G=1;
                                                                }
                                                                for(int i=1;
i<=totaloper; i++)
                                                                {

        if(O[i].G==1&&O[i].R==temp2)

        O[i].G=4;
                                                                }
                                                        }
                                                        break;
                                                default: break;
                                                }
                                                break;
                                        case 2:
                                                x2=randint(1,4);
                                                switch(x2)
                                                {
                                                case 1:

                                                        for(int i=1; i<=totaloper;
i++)
                                                        {
```

```
        if(O[i].G==1&&O[i].R>temp3)
                                                        temp3=O[i].R;
                                                }
                                                for(int i=1; i<=totaloper;
i++)
                                                {

        if(O[i].G==1&&O[i].R==temp3)
                                                        O[i].G=2;
                                                }
                                                decision =
grouping_energy();
                                                if (decision>1)
Enew=decision;
                                                else if (decision==1)
                                                {
                                                        for(int i=1;
i<=totaloper; i++)
                                                        {

        if(O[i].G==2&&O[i].R==temp3)

        O[i].G=1;
                                                        }
                                                }
                                                break;
                                        case 2:
                                                for(int i=1; i<=totaloper;
i++)
                                                {

        if(O[i].G==2&&O[i].R>temp3)
                                                        temp3=O[i].R;
                                                }
                                                for(int i=1; i<=totaloper;
i++)
                                                {
```

```
        if(O[i].G==2&&O[i].R==temp3)
                                                        O[i].G=3;
                                                }
                                                decision =
grouping_energy();
                                                if (decision>1)
Enew=decision;
                                                else if (decision==1)
                                                {
                                                        for(int i=1;
i<=totaloper; i++)
                                                        {
        if(O[i].G==3&&O[i].R==temp3)
        O[i].G=2;
                                                        }
                                                }
                                                break;
                                        case 3:
                                                for(int i=1; i<=totaloper;
i++)
                                                {
        if(O[i].G==3&&O[i].R>temp3)
                                                        temp3=O[i].R;
                                                }
                                                for(int i=1; i<=totaloper;
i++)
                                                {
        if(O[i].G==3&&O[i].R==temp3)
                                                        O[i].G=4;
                                                }
                                                decision =
grouping_energy();
```

```
                                                        if (decision>1)
Enew=decision;
                                                        else if (decision==1)
                                                        {
                                                               for(int i=1;
i<=totaloper; i++)
                                                               {
       if(O[i].G==4&&O[i].R==temp3)
       O[i].G=3;
                                                               }
                                                        }
                                                        break;
                                                 case 4:
                                                        for(int i=1; i<=totaloper;
i++)
                                                        {
       if(O[i].G==4&&O[i].R>temp3)
                                                                      temp3=O[i].R;
                                                        }
                                                        for(int i=1; i<=totaloper;
i++)
                                                        {
       if(O[i].G==4&&O[i].R==temp3)
                                                                      O[i].G=1;
                                                        }
                                                        decision =
grouping_energy();
                                                        if (decision>1)
Enew=decision;
                                                        else if (decision==1)
                                                        {
                                                               for(int i=1;
i<=totaloper; i++)
                                                               {
```

```
        if(O[i].G==1&&O[i].R==temp3)

        O[i].G=4;
                                                                }
                                                        }
                                                        break;
                                                default: break;
                                                }
                                        counter1++;
                                        if (Enew < Ec || (rand()/(double)RAND_MAX) <=
pow(e,-(Enew-Ec)/T))
                                        {
                                                Ec=Enew;
                                        }
                                }
                        }
                }

        int j=1;
        for (int i=1; i<=totaloper; i++)
        {
                if(BG[i].G==1)
                {Q[j-1]=i; j++;}
        }
        for (int i=1; i<=totaloper; i++)
        {
                if(BG[i].G==2)
                {Q[j-1]=i; j++;}
        }
        for (int i=1; i<=totaloper; i++)
        {
                if(BG[i].G==3)
                {Q[j-1]=i; j++;}
        }
        for (int i=1; i<=totaloper; i++)
        {
                if(BG[i].G==4)
                {Q[j-1]=i; j++;}
```

```
	}

	Ecs = sequencing_energy();

	for (float T=1000000; T>=1; T *= lambda)
			{
				for (int n=0; n<50; n++)
				{
					x3=randint(1,totaloper);
					int index=0; int x3p=0; int temp4=0; int
decision=0;
					for(int i=0; i<=totaloper-1; i++)
					{
						if(BG[Q[i]].G==BG[x3].G)
						{
							R[index]=Q[i]; index++;
						}
					}
					for(int i=0; i<index; i++)
					{
						if(R[i]==x3)
						{ x3p=i; }
					}
					for(int i=0; i<index; i++)
					{
						if(R[i]!=x3)
						{
							temp4=R[i];
							R[i]=x3;
							R[x3p]=temp4;
							decision=0;
							for(int j=0; j<index-1; j++)
							{
								for(int k=j+1; k<index;
k++)
								{

	if(O[R[j]].R==O[R[k]].R&&R[j]>R[k])//precedence within a feature
											decision=1;
```

```
                                                                else if
((O[R[j]].I==O[R[k]].R||O[R[k]].I==O[R[j]].R)&&R[j]>R[k])//precedence between
features
                                                                        decision=1;
                                                                }
                                                        }
                                                        if (decision==1) { R[i]=temp4;
R[x3p]=x3; }
                                                        else if (decision==0)
                                                                {
                                                                        int j=0;
                                                                        for(int k=0;
k<=totaloper-1; k++)
                                                                        {

        if(BG[Q[k]].G==BG[x3].G&&j<index)
                                                                                {Q[k]=R[j];
j++;}
                                                                        }

        Enew=sequencing_energy();
                                                                        R[i]=temp4;
R[x3p]=x3;
                                                                }
                                                        }
                                                }
                                                for (int i=0; i<=totaloper-1; i++)
                                                        {
                                                                Q[i] = BS[i];
                                                        }
                                                for (int i=0; i<index; i++)
                                                        {
                                                                R[i]=0;
                                                        }

                                                counter2++;
                                                Enew=sequencing_energy();
                                                if (Enew < Ecs || (rand()/(double)RAND_MAX) <=
pow(e,-(Enew-Ecs)/T))
```

```
                                {
                                        Ecs=Enew;
                                }
                        }

                }


        cout << " Best Grouping Objective= " << Ebg << " Last Grouping Objective= "
<< Ec << " Iterations= " << counter1 << endl;
        for (int i=1; i<=totaloper; i++)
                {
                        cout << "Operation No. " << i << "  R= " << O[i].R << " G= "
<< BG[i].G << endl;
                }

        cout << " Best Sequencing Objective= " << Ebs << " Last Sequencing
Objective= " << Ecs << " Iterations= " << counter2 << endl;
        for (int i=1; i<=totaloper; i++)
                {
                        cout << "Position No. " << i << " Operation no.= " << BS[i-1]
<< endl;
                }

        display_result();

        clock_t end_time=clock();
        cout << "elapsed time= " << (end_time-start_time)/CLOCKS_PER_SEC;
        char end;
        cin >> end;
}
```

APPENDIX C: GAP ANALYSIS FOR LITERATURE

Table 13: Gap analysis for relevant literature

Category	Problem	Objective	Method	Gap
Transfer line balancing (cost-based approach)	Equipment block allocation	Minimize line investment cost	Branch and bound algorithm (2)	Small problem size
			MILP and two heuristic algorithms (10)	Solution time is long
			Enhanced MILP (11)	Solution time is long
Transfer line balancing (time-based approach)	Machine loading, tool allocation and operation sequencing	Minimize handing time	MILP (12)	Small problem size
			Decomposition and linearization approach to MILP (13)	
			Ant colony and hybrid heuristic (14)	Solution time is long
			Analysis and simulation (16)	No feature grouping / operation sequencing
Transfer line balancing (minimize no. of machines)	Machine loading and operation sequencing	Minimize number of machines	MILP with algorithm (1)	Small problem size, No feature grouping / tool allocation
Transfer line design	Machine loading and tool allocation	Minimize equipment cost	MILP (7)	No feature grouping / operation sequencing
	Hierarchical process planning	Multi-criteria evaluation	Analysis (15)	No line balancing
FMS	Part grouping	Maximize machine	MILP (17)	No Feature

Scheduling	and batch scheduling	utilization		grouping
	Machine loading, tool allocation and part routing	Minimize total machining cost	MILP (18)	No operation sequencing
	Operations sequencing and tool allocation	Minimize tool change time	Heuristic algorithm and simulation (20)	Solution is near optimal
Line balancing / operation sequencing	Assembly line balancing	Minimize total cost, smoothness index and probability of lateness	Simulated annealing algorithm (22)	No tool allocation or operation sequencing
	Operation sequencing	Maximize tool changeover index, setup changeover index, motion continuity index and loose precedence index	Simulated annealing algorithm (23)	No machine loading or tool allocation
	Operation sequencing	Minimize non-cutting time	Simulated annealing algorithm (24)	No machine loading or tool allocation
	Assembly line balancing	Minimize smoothness index, minimize probability of line stopping	Simulated annealing algorithm (26)	No tool allocation or operation sequencing

APPENDIX D: TLCT MATRIX FOR CASE STUDY 1

Table 14: Tool change time matrix for Case Study 1 (Operations 1-19) (sec)

[DFU, Op]	[1,1]	[1,2]	[1,3]	[1,4]	[2,5]	[2,6]	[2,7]	[2,8]	[3,9]	[3,10]	[3,11]	[4,12]	[4,13]	[4,14]	[5,15]	[5,16]	[6,17]	[6,18]	[6,19]
[1,1]	0	8	6	9	0	9	4	4	0	1	7	6	3	7	10	4	4	2	10
[1,2]	10	0	9	8	1	0	1	1	2	9	2	6	3	2	7	9	6	3	9
[1,3]	6	2	0	7	6	8	5	10	8	10	2	1	1	1	6	3	10	3	5
[1,4]	1	1	10	0	1	1	0	5	4	0	6	1	7	6	6	3	8	8	9
[2,5]	0	6	2	10	0	5	8	1	0	2	8	1	2	7	7	5	1	8	5
[2,6]	9	0	1	4	9	0	5	4	2	1	7	5	6	8	5	1	3	9	2
[2,7]	1	4	6	0	8	5	0	2	7	0	9	8	4	6	4	4	1	5	6
[2,8]	8	1	3	9	8	1	7	0	3	6	0	8	7	3	6	8	4	5	8
[3,9]	0	3	1	8	0	2	8	7	0	10	9	2	5	7	6	7	2	8	4
[3,10]	4	9	9	0	1	4	0	4	3	0	10	3	5	10	8	10	10	4	3
[3,11]	4	2	3	3	5	7	8	0	5	1	0	4	6	4	8	1	2	7	5
[4,12]	5	2	6	4	10	1	5	9	2	5	1	0	8	9	1	8	5	2	5
[4,13]	2	4	2	6	10	1	6	2	9	1	4	10	0	2	6	1	1	9	2
[4,14]	8	8	6	6	2	10	7	5	10	10	8	4	5	0	3	4	4	10	3
[5,15]	9	3	6	9	2	10	1	9	7	4	9	6	9	2	0	10	0	5	10
[5,16]	6	6	7	6	4	8	1	8	7	10	4	2	3	7	3	0	6	0	1
[6,17]	3	4	10	1	8	5	4	6	5	8	8	2	4	6	0	2	0	8	5
[6,18]	5	6	5	1	6	7	8	8	4	8	1	8	2	1	9	0	4	0	7
[6,19]	2	2	7	4	3	7	10	9	4	7	3	1	2	5	7	3	10	6	0

[6, 20]	1	4	1	1	2	6	10	7	7	2	1	5	1	3	10	10	1	5	6
[7, 21]	5	6	9	10	2	8	7	5	6	7	4	6	9	6	4	0	5	0	1
[7, 22]	7	6	9	4	9	6	7	10	7	5	8	3	6	5	2	2	3	3	0
[7, 23]	8	1	7	9	10	10	2	0	4	8	10	7	3	9	6	10	5	6	8
[8, 24]	7	3	0	7	5	1	7	5	8	3	2	3	9	9	5	9	3	10	2
[8, 25]	6	5	5	0	8	1	0	2	9	0	1	2	5	7	9	5	6	5	1
[8, 26]	10	8	2	4	3	4	6	0	9	9	0	2	5	4	1	2	5	4	6
[9, 27]	2	9	0	2	5	3	1	10	8	10	7	2	9	4	9	2	1	9	9
[9, 28]	9	6	7	0	2	10	0	3	6	0	7	1	5	3	6	8	9	9	8
[10, 29]	0	2	10	10	0	4	5	7	0	5	7	2	3	9	9	5	6	10	4
[10, 30]	6	0	3	8	2	0	6	10	6	2	8	6	2	1	8	8	4	1	3
[11, 31]	2	8	6	8	2	7	8	4	4	6	7	10	2	6	2	1	5	3	0
[11, 32]	2	2	5	1	7	3	6	1	10	7	10	5	2	7	7	5	6	3	3
[11, 33]	10	7	4	6	1	3	5	0	6	9	0	7	5	1	4	9	6	7	7
[11, 34]	3	2	1	8	8	7	3	6	1	2	7	0	7	9	2	1	9	9	9
[12, 35]	4	5	7	7	5	8	7	1	4	10	9	10	6	0	10	6	9	9	3
[12, 36]	10	5	6	9	10	7	9	7	2	5	1	2	9	10	0	2	0	10	1
[12, 37]	10	9	5	4	4	10	9	2	2	10	3	3	1	3	3	0	7	0	4
[12, 38]	7	5	7	6	4	8	10	1	7	1	10	5	6	5	1	2	8	3	0

Table 15: Tool change time matrix for Case Study 1 (Operations 20-38) (sec)

[DFU, Op]	[6,20]	[7,21]	[7,22]	[7,23]	[8,24]	[8,25]	[8,26]	[9,27]	[9,28]	[10,29]	[10,30]	[11,31]	[11,32]	[11,33]	[11,34]	[12,35]	[12,36]	[12,37]	[12,38]
[6,20]	3	1	3	2	5	10	7	6	9	0	8	4	6	7	9	9	2	2	10
[7,21]	10	4	1	8	7	8	8	2	4	8	0	5	7	1	5	1	3	9	10
[7,22]	5	7	10	10	0	3	7	0	1	9	1	8	9	6	7	6	6	6	3
[7,23]	7	10	6	10	10	0	7	1	0	3	8	3	8	7	1	10	8	7	5
[8,24]	4	10	6	9	10	9	9	6	6	0	6	10	7	10	3	7	1	1	4
[8,25]	7	5	8	3	8	3	7	1	2	4	0	1	4	9	4	5	4	2	1
[8,26]	3	6	5	10	9	0	3	7	0	2	4	10	7	7	1	7	10	1	7
[9,27]	5	9	2	6	9	6	0	2	4	10	2	1	8	0	8	6	6	7	1
[9,28]	6	4	4	8	4	2	1	6	6	0	4	2	4	5	4	1	5	7	3
[10,29]	1	9	2	7	1	0	8	3	0	7	1	10	6	5	8	9	5	2	3
[10,30]	4	9	9	7	2	4	0	1	4	2	6	9	9	0	3	4	7	1	4
[11,31]	2	7	2	10	4	6	9	4	2	3	7	7	6	9	0	8	3	1	5
[11,32]	2	4	10	8	3	3	6	7	6	7	1	3	1	7	3	9	3	7	3
[11,33]	9	8	7	6	1	6	10	4	9	7	3	2	5	6	3	0	3	9	8

[11, 34]	5	10	4	5	10	8	4	9	9	7	7	3	5	2	9	8	0	1	1
[12, 35]	6	0	10	2	9	5	7	4	3	5	7	8	7	6	5	7	6	0	5
[12, 36]	7	10	10	1	8	1	10	8	2	6	3	9	7	8	7	2	0	4	10
[12, 37]	10	0	9	3	3	3	10	2	8	1	8	2	6	10	10	4	2	0	3
[12, 38]	8	7	0	9	1	2	5	6	3	2	10	0	6	9	2	6	8	4	0
[6, 20]	0	4	4	0	7	5	4	4	4	4	6	4	0	8	8	1	5	2	6
[7, 21]	9	0	7	7	4	2	2	10	6	5	3	10	1	1	6	8	8	0	7
[7, 22]	3	2	0	6	10	4	9	10	3	4	2	0	8	1	7	6	10	2	0
[7, 23]	0	1	5	0	7	10	7	9	7	6	4	3	0	7	9	6	6	1	9
[8, 24]	7	10	5	5	0	1	9	0	3	9	6	7	5	4	9	5	3	5	6
[8, 25]	3	9	10	8	10	0	6	4	0	5	7	4	9	8	7	1	5	7	3
[8, 26]	4	4	1	8	3	9	0	5	4	8	10	10	8	0	2	8	4	3	4
[9, 27]	10	2	6	1	0	5	10	0	2	3	2	5	2	5	2	2	8	9	6
[9, 28]	2	9	7	8	2	0	9	2	0	4	9	10	6	8	4	2	7	6	10
[10, 29]	9	9	4	5	1	9	5	3	7	0	10	5	9	6	9	3	6	3	5
[10, 30]	3	7	4	1	4	8	8	4	6	9	0	8	10	4	3	5	6	4	7

[11, 31]	8	3	0	6	8	1	1	7	5	4	3	0	4	5	8	8	2	8	0
[11, 32]	0	7	9	0	9	6	1	8	3	1	9	1	0	7	3	9	8	8	5
[11, 33]	10	6	5	5	2	5	0	6	8	3	4	6	6	0	4	1	3	6	2
[11, 34]	10	7	6	4	1	6	7	9	2	5	6	2	10	3	0	10	6	2	10
[12, 35]	8	7	7	3	2	1	4	8	10	7	6	10	5	1	5	0	9	4	1
[12, 36]	9	4	5	9	8	9	1	7	6	3	3	3	4	5	7	8	0	3	8
[12, 37]	10	0	5	1	9	1	4	8	1	9	10	1	8	1	7	1	2	0	3
[12, 38]	5	4	0	1	3	4	4	6	3	2	2	0	9	6	4	9	7	3	0

APPENDIX E: DETAIL RESULTS FOR CASE STUDY 2

Table 16: Computational results for problem 1

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	0.27 s	56.84	715.04	<0.1 s	56.84	715.04
2	0.23 s	58	504.1	<0.1 s	58	504.1
3	0.19 s	51.16	588.56	<0.1 s	51.16	588.56
4	0.17 s	68.73	582.13	<0.1 s	68.73	582.13
5	0.17 s	61.15	591.75	<0.1 s	61.15	591.75
6	0.17 s	61.85	733.45	<0.1 s	61.85	733.45
7	0.16 s	62.46	620.76	<0.1 s	62.46	620.76
8	0.17 s	51.57	537.27	<0.1 s	51.57	537.27
9	0.14 s	59.84	572.24	<0.1 s	59.84	572.24
10	0.19 s	61.16	707.16	<0.1 s	61.16	707.16
Mean	0.19 s	59.28	615.25	<0.1 s	59.28	615.25
Standard deviation	0.036	5.23	78.21	0	5.23	78.21

Table 17: Computational results for problem 2

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	0.22 s	37.33	416.53	<0.1 s	41.33	501.93
2	0.22 s	36.91	356.11	<0.1 s	39.76	358.76
3	0.13 s	33.56	409.6	<0.1 s	33.56	409.6
4	0.16 s	45.61	432.31	<0.1 s	46.2	432.9

5	0.14 s	36.82	442.52	<0.1 s	36.82	442.52
6	0.13 s	35.13	420.23	<0.1 s	41.43	442.63
7	0.11 s	46.03	527.53	<0.1 s	47.94	529.44
8	0.11 s	39.97	414.17	<0.1 s	40.19	414.39
9	0.12 s	49.76	396.66	<0.1 s	49.88	396.78
10	0.16 s	34.36	341.96	<0.1 s	35.47	343.07
Mean	0.15 s	39.55	415.76	<0.1 s	41.26	427.2
Standard deviation	0.04	5.63	50.46	0	5.36	57.45

Table 18: Computational results for problem 3

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	0.39 s	55.25	770.95	<0.1 s	55.25	770.95
2	0.27 s	70.77	743.67	<0.1 s	72.86	745.76
3	0.28 s	74.08	581.78	<0.1 s	74.33	582.03
4	0.25 s	59.83	659.03	<0.1 s	62.6	661.8
5	0.23 s	64.96	888.36	<0.1 s	68.97	892.37
6	0.22 s	67.83	661.33	<0.1 s	68.69	662.43
7	0.22 s	68.75	776.95	<0.1 s	70.84	779.04
8	0.2 s	68.69	709.49	<0.1 s	70.41	711.21
9	0.25 s	60.17	643.87	<0.1 s	63.38	647.08
10	0.27 s	60	620.5	<0.1 s	60.63	621.23
Mean	0.25 s	65.03	705.6	< 0.1 s	66.8	707.39
Standard deviation	0.05	5.98	91.13	0	6.07	91.75

Table 19: Computational results for problem 4

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	1.86 s	68.31	659.51	1 s	77.48	668.68
2	1.98 s	80.26	736.36	1 s	80.26	736.36
3	1.88 s	75.38	762.38	1 s	81.72	768.72
4	1.9 s	90.94	673.04	1 s	91.99	674.09
5	1.88 s	76.36	766.16	1 s	86.48	766.28
6	1.81 s	70.32	617.02	1 s	72.85	619.55
7	1.88 s	83.48	763.88	1 s	90.28	770.68
8	1.89 s	82.08	860.08	1 s	82.68	860.68
9	1.88 s	75.77	659.57	1 s	85.42	668.92
10	1.81 s	74.81	886.21	1 s	80.84	892.24
Mean	1.88 s	77.71	738.42	1 s	83	742.62
Standard deviation	0.05	6.64	88.25	0	5.76	87.64

Table 20: Computational results for problem 5

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	1.81 s	62.5	610	1 s	90.06	637.56
2	1.91 s	87.63	795.83	1 s	94.35	802.55
3	1.94 s	79.49	757.89	1 s	90.17	768.57
4	2.37 s	83.02	725.62	1 s	83.02	725.62
5	1.95 s	91.42	891.82	1 s	94.07	894.57
6	1.92 s	91.01	851.31	1 s	94.12	854.42

7	1.98 s	84.57	770.97	1 s	93.22	779.62
8	1.94 s	80.33	773.13	1 s	88.92	781.72
9	1.86 s	62.99	712.29	1 s	73.12	722.42
10	1.89 s	89.3	943.4	1 s	97.19	951.29
Mean	1.96 s	81.22	783.22	1 s	89.82	791.83
Standard deviation	0.15	10.57	95.16	0	7.06	90.56

Table 21: Computational results for problem 6

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	0.45 s	69.61	748.11	1 s	74.12	752.62
2	0.31 s	83.98	846.88	1 s	84.45	847.35
3	0.34 s	92.14	827.04	1 s	100.43	835.33
4	0.36 s	76.83	862.83	1 s	81.71	867.71
5	0.41 s	73.17	921.87	1 s	79.19	927.89
6	0.33 s	84.21	815.41	1 s	97.28	828.48
7	0.33 s	77.63	673.93	1 s	80.83	677.13
8	0.33 s	77.94	647.14	1 s	85.69	654.89
9	0.36 s	79.6	915.2	1 s	81.54	917.14
10	0.34 s	71.48	909.88	1 s	79.52	917.19
Mean	0.36	78.66	816.83	1 s	84.48	822.58
Standard deviation	0.04	6.75	98	0	8.23	97.76

Table 22: Computational results for problem 7

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	5 s	104.98	952.68	1 s	120.73	968.43
2	5.23 s	102.89	1004.29	1 s	112.07	1013.47
3	5.23 s	90.91	880.41	1 s	106.71	896.21
4	5.3 s	110.93	1156.13	1 s	118.74	1163.94
5	5.91 s	114.51	861.51	1 s	127.01	874.01
6	5.06 s	89.08	856.98	1 s	104.6	872.5
7	5.06 s	100.93	833.53	1 s	109.55	842.15
8	5.13 s	101.46	913.56	1 s	105.02	917.12
9	5.17 s	102.5	961.2	1 s	115.48	974.18
10	5.22 s	112.49	834.79	2 s	148.34	870.64
Mean	5.23 s	103.07	925.51	1.1 s	116.83	939.27
Standard deviation	0.26	8.41	99.48	0.32	13.27	96.18

Table 23: Computational results for problem 8

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	12.19 s	132.15	1358.85	2 s	151.96	1378.66
2	7.5 s	134.56	1328.16	3 s	141.46	1335.06
3	13.95 s	131.27	1100.37	2 s	147.97	1117.07
4	6.86 s	131.01	1339.31	2 s	140.87	1349.17
5	4.3 s	146.98	1285.38	2 s	148.93	1287.33
6	3.25 s	119.64	1233.74	3 s	137.26	1251.36

7	3.33 s	155.8	1335.6	2 s	160.29	1340.09
8	7.16 s	134.28	1441.08	2 s	147.5	1454.3
9	6.27 s	128.04	1273.04	2 s	141.91	1286.91
10	7.14 s	146.68	1488.08	2 s	153.02	1494.42
Mean	7.2 s	136.04	1318.36	2.2 s	147.12	1329.44
Standard deviation	3.5	10.67	107.58	0.42	6.93	105.7

Table 24: Computational results for problem 9

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	2.56 s	148.5	1450.7	2 s	164.9	1467.1
2	2.5 s	139.9	1314.9	2 s	146.4	1321.4
3	1.73 s	143.8	1695.8	2 s	172	1724
4	2.31 s	166.2	1615.3	3 s	166.2	1615.3
5	2.19 s	167.9	1426	2 s	170.3	1428.3
6	2.39 s	163.6	1572.3	2 s	169.5	1639.6
7	2.17 s	158.8	1590.4	3 s	160.4	1592
8	2.44 s	146.8	1501	2 s	161.9	1516.1
9	2.23 s	159.6	1345.1	2 s	169.9	1355.4
10	2.58 s	155.5	1459.3	2 s	162.2	1466
Mean	2.31 s	155.06	1497.08	2.2 s	164.37	1512.52
Standard deviation	0.25	9.8	121.46	0.42	7.48	129.23

Table 25: Computational results for problem 10

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	25.75 s	211.52	2031.52	4 s	232.03	2052.03
2	18.42 s	217.87	1947.17	4 s	219.91	1949.21
3	22.72 s	216.16	1878.76	4 s	225.8	1888.4
4	22.55 s	206.86	1674.56	5 s	223.29	1690.99
5	28.55 s	239	2079.8	4 s	239.27	2080.07
6	20.98 s	225.39	1906.79	4 s	237.09	1918.49
7	36.05 s	236.86	2092.66	4 s	257.96	2113.56
8	24.14 s	211.63	1936.23	5 s	220.21	1944.81
9	20.89 s	205.79	1973.19	5 s	233.05	2000.45
10	24.56 s	217.97	2025.67	5 s	229.26	2036.96
Mean	24.46 s	218.91	1954.64	4.4 s	231.79	1967.5
Standard deviation	4.95	11.56	121.52	0.52	11.36	121.58

Table 26: Computational results for problem 11

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	5.8 s	185.61	1949.51	2 s	187.74	1951.64
2	5.13 s	178.77	1903.67	3 s	189.89	1914.79
3	7.95 s	163.64	1661.24	3 s	185.49	1683.09
4	5.47 s	160.29	1715.79	2 s	179.49	1734.99
5	5.73 s	192.64	1670.74	3 s	205.76	1683.86
6	6.72 s	180.87	1627.87	3 s	199.05	1646.05

7	6.95 s	193.9	1857.7	3 s	210.7	1874.5
8	6 s	167.81	1540.91	3 s	199.95	1573.05
9	6.45 s	182.1	1815.9	3 s	196.6	1830.4
10	5.53 s	179.14	1658.34	3 s	200.6	1679.8
Mean	6.17 s	178.48	1740.16	2.8 s	195.53	1757.22
Standard deviation	0.85	11.42	133.65	0.42	9.69	127.09

Table 27: Computational results for problem 12

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	17.34 s	225.04	1924.14	4 s	246.22	1945.32
2	12.77 s	260.88	2114.58	4 s	276.58	2130.28
3	17.34 s	238.22	2292.22	3 s	240.53	2294.53
4	14.3 s	238.98	2253.88	5 s	251.88	2266.78
5	12.78 s	240.94	2319.64	4 s	251.73	2330.43
6	15.75 s	242.26	2306.96	4 s	264.95	2329.65
7	12.06 s	218.41	2239.61	4 s	243.02	2264.22
8	12.59 s	227.24	2183.44	4 s	248.7	2204.9
9	12.61 s	229.86	2389.46	4 s	248.32	2407.92
10	19.22 s	243.06	2388.36	4 s	271.13	2416.43
Mean	14.67 s	236.49	2241.23	4 s	254.3	2259.04
Standard deviation	2.56	11.96	141.15	0.47	12.27	139.96

Table 28: Computational results for problem 13

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	3.92 min	304.98	2889.08	6 s	322.44	2906.54
2	5.66 min	322.11	2872.51	6 s	367.88	2918.28
3	7.18 min	307.79	3030.99	6 s	310.69	3033.89
4	4.69 min	291.29	2670.39	7 s	308.37	2687.47
5	6.05 min	293.5	2933.7	8 s	319.46	2982.66
6	11.22 min	294.97	2802.67	7 s	314.06	2821.76
7	5.96 min	304.68	2500.28	6 s	315.94	2511.54
8	1.29 min	292.41	2961.21	6 s	349.85	3018.65
9	4.17 min	292.94	2854.94	7 s	302.68	2864.68
10	1.28 min	311.19	2683.79	7 s	318.93	2691.53
Mean	5.14 min	301.59	2819.96	6.6 s	323.03	2843.7
Standard deviation	2.89	10.26	159.72	0.7	20.20	168.13

Table 29: Computational results for problem 14

Test no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	5.73 h	373.07	3402.37	10 s	416.78	3446.08
2	4.88 h	370.36	3345.76	14 s	418.59	3393.99
3	4.27 h	377.37	3408.27	10 s	425.95	3456.85
4	3.46 h	393.95	3618.85	10 s	448.27	3673.17
5	8.97 h	362.63	3278.03	11 s	400.79	3316.19
6	4.54 h	385.55	3444.35	11 s	438.15	3496.95

7	3.59 h	388.51	3600.21	9 s	453.79	3665.49
8	4.56 h	390.86	3423.66	13 s	431.61	3464.41
9	2.71 h	372.17	3393.77	10 s	437.91	3459.51
10	1 h	389.44	3556.08	10 s	441.85	3608.45
Mean	4.37 h	380.39	3447.17	10.8 s	431.37	3498.11
Standard deviation	2.08	10.62	110.85	1.55	16.08	116.34

Table 30: Computational results for problem 15

Te0st no.	MILP model			Simulated annealing algorithm		
	Solution time	Handling time (s)	Makespan (s)	Solution time	Handling time (s)	Makespan (s)
1	>24 h	-	-	15 s	446.34	4167.48
2	>24 h	-	-	10 s	456.76	4295.72
3	>24 h	-	-	13 s	433.79	4224.2
4	>24 h	-	-	10 s	425.01	4307.78
5	>24 h	-	-	12 s	459.31	3876.95
6	>24 h	-	-	10 s	453.93	4204.65
7	>24 h	-	-	13 s	422.25	4522.25
8	>24 h	-	-	10 s	451.76	4101.23
9	>24 h	-	-	11 s	436.01	4309.29
10	>24 h	-	-	10 s	437.91	4390.82
Mean	>24 h	-	-	11.4 s	442.31	4240.03
Standard deviation	0	-	-	1.78	13.21	174.08

VITA AUCTORIS

NAME:	Soumitra Subhash Bhale
PLACE OF BIRTH:	Aurangabad, Maharashtra, India
YEAR OF BIRTH:	1989
EDUCATION:	MASc. University of Windsor, Windsor, ON 2014 B.E. University of Pune, Pune, India 2010 H.S.C. M.S.B.S.H.S.E., Aurangabad, India 2006 S.S.C. M.S.B.S.H.S.E., Aurangabad, India 2004
PUBLICATION:	Bhale S, Baki MF, Azab A. Grouping and Sequencing of Machining Operations for High Volume Transfer Lines. Proceedings of the 47th CIRP Conference on Manufacturing Systems; 2014 Apr 28-30; Windsor, Canada. Windsor: Canada; 2014. P. 413-18.