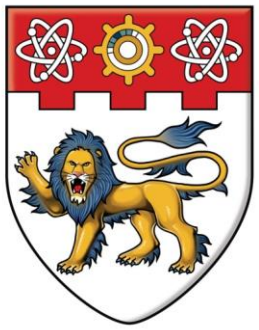

NANYANG TECHNOLOGICAL UNIVERSITY

META-HEURISTICS FOR SCHEDULING FLEXIBLE JOB SHOP PROBLEMS WITH CONSTRAINTS

GAO KAIZHOU

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

2015

META-HEURISTICS FOR SCHEDULING FLEXIBLE JOB SHOP PROBLEMS WITH CONSTRAINTS

GAO KAIZHOU

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirements for the degree of
Doctor of Philosophy

2015

Statement of Originality

I hereby certify that the content of this thesis is the result of work done by me and has not been submitted for a higher degree to any other University or Institution.

…… ……....

Date GAO KAIZHOU

Acknowledgments

I would like to take this opportunity to express my gratitude to those who contributed to the successful completion of my postgraduate studies and this research thesis. I would thank them all, but there are some people who need special recognition.

First, I want to thank my supervisor, Associate Professor Ponnuthurai Nagaratnam Suganthan. He has provided his careful guidance, stimulating suggestions, exact insights and profound knowledge in supervising my research work throughout my Ph.D study. Without his precise and appropriate instructions at every step during my research, I could not accomplish my postgraduate study smoothly.

Thank to my co-supervisor, Dr Chong Chin Soon, for advice, discussion and helping me in reviewing and modifying my papers. He is always prudent in every detail. I really appreciate all his generous help.

I also want to express my thanks to Dr. Pan Quan-Ke, Mr. Chua TayJin, and Mr. Cai TianXiang. During the collaboration with them on several research works, they have provided valuable suggestions as well as shared precious experiences on those work.

I would like to thank all my good friends and lab-mates including Mr. Yang ChengBin, my roommate, for the fun and joy we have had during the past three years.

Meanwhile, thanks a lot to my dear parents, my older brother, my sister in law, my nephews and nieces.

Finally, thanks with lots of love to my wife, Wang MeiFang, and my son, Gao MingYu. They encouraged and supported me to complete the Ph.D study with their patience and love.

Contents

Summary

Flexible job-shop scheduling problem (FJSSP) is an extension of the classical job shop scheduling problem (JSSP) with practical applications. FJSSP has been proven as an NP-hard problem. Many researchers have focused on FJSSP in recent years.

FJSSP includes two sub-problems: 1. machine assignment that is to select a machine from a set of candidate machines for operations; 2. operation sequencing that is to schedule the operations on machines to obtain a feasible solution. In the classical JSSP problem, one operation can be processed on only one machine. Hence, the FJSSP is more difficult than the classical JSSP. There is a great variety of real-world problems that can be modeled as FJSSP, e.g., optimization of crane operations, simulation and optimization of transport systems, and scheduling of manufacturing and remanufacturing systems.

In this thesis, the scheduling problems in the remanufacturing industry are modeled as FJSSP with multiple constraints. The constraints are new job insertion and uncertain processing time. The uncertain processing time constraint is built into two models, one is based on most probable processing time and another is based on fuzzy processing time. A rescheduling operator is executed when a new job is inserted or processing time becomes larger than the most probable processing time. The problem size may change dynamically during scheduling and rescheduling stages. The objectives include minimizing maximum completion time (Makespan), minimizing maximum machine workload, minimizing the average of earliness and tardiness and so on.

Compared to exact methods, approximate methods, especially meta-heuristics, are better for solving FJSSP because meta-heuristics can obtain satisfactory solutions in a reasonable time. Meta-heuristics are more suitable for large-scale FJSSP. A music-inspired algorithm, discrete harmony search (DHS), and a nature-inspired algorithm, artificial bee colony (ABC), are investigated and improved to solve FJSSP with constraints.

Two encoding and decoding methods are developed for solution representation. Several simple and ensemble heuristics are employed to initialize population. An effective heuristic, named MinEnd heuristic is proposed to generate a high quality initial solution. Dynamic grouping strategies of harmony memory and new crossover method for improvising new harmonies are proposed. Effective local search operators are proposed to improve algorithms' performance. For the rescheduling operator, three re-scheduling strategies are proposed and compared. A two-stage discrete harmony search

algorithm and a two-stage artificial bee colony algorithm are proposed for rescheduling FJSSP with remanufacturing constraints.

Extensive experiments have been conducted to test the performance of the proposed algorithms for solving FJSSP with remanufacturing constraints. The instance sets used in this thesis include three sets of benchmark instances and two sets from remanufacturing industry. Single objective, multi-objective and Pareto-based multi-objective formulations are considered in this thesis. The proposed algorithms are compared to more than ten existing heuristics and algorithms. The comparisons with literature show the effectiveness and efficiency of the proposed heuristics and algorithms.

List of Figure

List of Table

List of Abbreviations

JSP	job shop problem
JSSP	job shop scheduling problem
FJSSP	flexible job shop scheduling problem
FIFO	first in first out
SPT	short processing time
EDD	earliest due date first
CDR	composite dispatch rules
GA	genetic algorithm
ACO	ant colony optimization
PSO	particle swarm optimization
DE	differential evaluation
HS	harmony search
DHS	discrete harmony search
PGDHS	Pareto-based grouping discrete harmony search
ABC	artificial bee colony
SA	simulated annealing
TS	tabu search
VNS	variable neighborhood search
HM	harmony memory
HMS	harmony memory size
HMCR	harmony memory consideration rate
PAR	pitch adjustment rate
MA	machine assignment vector
OS	operation sequence vector
TFN	triangle fuzzy number
QM	quality metric
DM	diversification metric

List of Symbols

J	set of jobs
M	set of machines
$O_{i,j}$	operation j of job i
$c_{i,j}$	the completion time of operation $O_{i,j}$
c_i	the completion time of job J_i
C_M	the maximum completion time, Makespan
E/T	the mean of earliness and tardiness
RPI	the relative percentage increase
C_W	the maximum machine workload
$ARPI$	the average relative percentage increase
n	number of job
m	number of machine
Di	the due date of job J_i
X_i	i^{th} harmony vector in the HM
X_{new}	new harmony

Chapter 1

Introduction

Remanufacturing is the process of disassembly and recovery at the module level and, eventually, at the component level. It requires the repair or replacement of worn out or obsolete components and modules. Parts subject to degradation affecting the performance or the expected life of the whole are replaced. Remanufacturing is a form of a product recovery process that differs from other recovery processes in its completeness: a remanufactured machine should match the same customer expectation as new machines. Lund [1] was the first one to define and discuss the remanufacturing problem. Krupp [2] perfected the concept of remanufacturing and predicted the future of remanufacturing engineering. There are seven characteristics of remanufacturing as follows [3]:

1. Routing uncertainty and processing time uncertainty. The condition of recovered parts is unknown until inspected. This leads to stochastic routing and lead times.
2. Balancing returns with demand. There is a problem of imperfect correlation between the supply of cores and the demand for remanufacturing units.
3. The disassembly of returned products. The disassembly and subsequent release of parts to the remanufacturing operations requires a high degree of coordination with reassembly to avoid high inventory levels or poor customer service.

4. Uncertainty in materials recovered. Some parts that are not recoverable must be replaced by new parts. This characteristic complicates resource planning and scheduling since the volumes of parts recovered and required processing will be a stochastic variable. Remanufacturing production schedule is to be tightly integrated with dependent demand inventory.
5. The requirement for reverse logistics network. It is the determination of how products will be collected and transported to the operation, involving a set of parallel decisions, such as the number of collection centers, method and frequency of transport.
6. Materials/parts matching requirements. Units are often composed of serial number specific parts and components, along with common ones. Coordination between disassembly and reassembly is critical if customer due dates are to be met.
7. Uncertainty in the timing and the quality of returns. The number of products returned and the timing are factors that cannot be controlled by remanufacturers, due to factors, such as stage of product lifecycle and the rate of technological change.

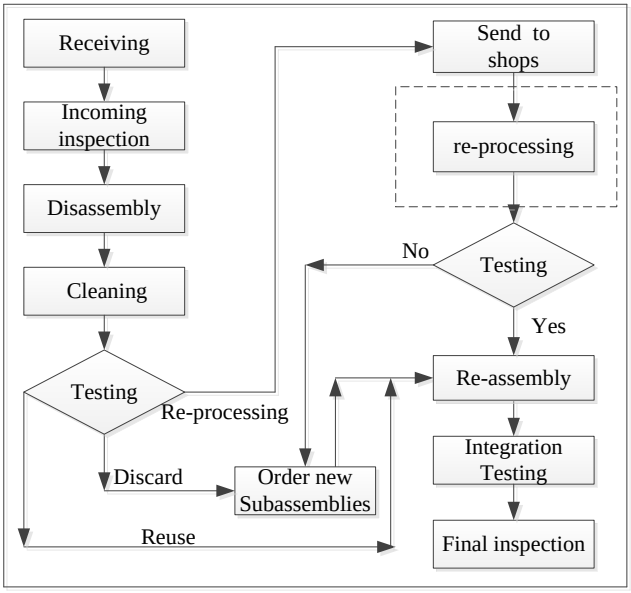

Fig. 1 - 1 The flow chart of remanufacturing system

Fig. 1 - 1 shows the process of remanufacturing, which is developed from the remanufacturing model in literature [4]. For remanufacturing, each return production has a set of subassemblies. After cleaning and testing each subassembly, the subassemblies can be classified into one of the three groups: (1) to be reused, (2) to be discarded and order new one or (3) to be re-processed. The subassembly that can be processed will be sent to shop floor for re-processing. After testing, the re-processed subassemblies will be re-assembled for final inspection.

In recent years, many researchers focused on remanufacturing theory research and industry application. Guide Jr [5] reported the production planning and control in remanufacturing for industry practice and research needs. Guide Jr [6] analyzed the capacity planning in remanufacturing environment. Guide [7] examined the impact of product structure complexity on other managerial and operational decisions in a remanufacturing environment. Depuy [4] proposed a methodology for production planning within facilities involved in the remanufacturing of products. Teunter [8] developed a polynomial-time dynamic programming heuristic for the minimization of cost in manufacturing and remanufacturing engineering. Grubbstrom [9] proposed a model to analyze how labor costs, material costs and budget influence the optimal production decisions in remanufacturing engineering. Guide Jr [10] considered the production planning when inputs have different and uncertain quality levels and discussed different decision variables in remanufacturing engineering. M.L. Junior [11] reviewed the literature on production planning and control in remanufacturing. Seventy-six papers were examined and classified in the literature. However, there are only a handful of papers on reprocessing scheduling in remanufacturing.

The literature on Production Planning & Scheduling in remanufacturing is dominated by quantitative approaches. The majority of these quantitative models developed are generic; namely, not associated with any kind of specific remanufacturing operation. The presence of complicating characteristics does not necessarily mean it is an area that has been fully addressed. In fact, because most studies are theoretical, the need for more practical research on each PPS activity based on the complicating characteristics is evident.

The re-processing of returns in remanufacturing can be modelled as FJSSP with constraints. The characteristics 1) and 7) can be modelled as constraints for scheduling re-processing. Hence, the re-processing scheduling problem in remanufacturing can be solved as FJSSP scheduling problem with two constraints. One constraint is that processing time may become larger or smaller in re-processing shop floor for characteristic 1). Another constraint is new job coming and inserting into scheduled sequences for characteristic 7).

1.1 Flexible job shop scheduling problem

Job shop scheduling problem (JSSP) is to solve *n* jobs processed on *m* machines[12]. Each job comprises multiple operations while each machine can process multiple operations. Each machine can process only one operation at a time and each operation can be processed on only one machine at a time. The purpose of scheduling is to assign all operations on machines to achieve best scheduling

objectives. In JSSP, each job has a predetermined routing. The operations of the same job have precedence constraints and are processed in a sequential order.

The flexible job-shop scheduling problem (FJSSP) is an extension of classical JSSP with practical applications[13]. FJSSP is a combination of the job shop and the parallel machine environments. Instead of *m* machines in series, there are multiple work centers with each work center consisting of a number of identical machines in parallel. Each job has its own route to follow through the shop and requires processing at each work center on only one machine and any machine in the work center can process the job. FJSSP includes two sub-problems: 1. machine assignment to select a machine from a set of candidate machines for each operation; 2. operation sequencing to schedule the operations on all machines to obtain a feasible solution. In classical JSSP problem, one operation can be processed on only one machine. Hence, the FJSSP is more difficult than classical JSSP. There is a great variety of real-world problems that can be modeled as FJSSP, e.g., optimization of crane operations, simulation and optimization of transport systems, and scheduling in real manufacturing and remanufacturing systems.

1. Features of FJSSP

FJSSP has the following features:

- Computational Complexity

Comparing to classic JSSP problem, FJSSP has an additional step of selecting a processing machine from candidate machine set for each operation. Hence, FJSSP has higher computation complexity than classic JSSP problem. FJSSP has been proven as an NP-hard problem [14].

Discreteness

FJSSP is a typical discrete event problem. Operations are processed on a different instant of time and different machines. In FJSSP, a job arrival, a change in orders, an addition of machine and a machine breakdown are all discrete events. Hence, discrete event simulation, queuing theory are employed for FJSSP.

- Multi-objectives [15]

In different enterprise and different production environments, the objectives of FJSSP are different and varied. For completion time related criteria, FJSSP scheduling objectives include minimizing maximum completion time (Makespan), minimizing total completion time or average completion time and so on. For machine load related criteria, there are minimizing maximum machine load, minimizing total machine load or average machine load objectives. The due date related criteria are minimizing earliness or tardiness, minimizing average earliness or tardiness, minimizing total earliness and tardiness. There are also some other scheduling objectives in FJSSP, for example, cost related objectives.

- Multi-constraints

Generally, the routing of each job is known before processing and the operations of a job have strict precedence. At the same time, the candidate machines are also known and the operations must be processed on one machine based on the precedence. In addition, there are many other constraints in different industries and different production environments, for example, uncertain processing time, new job insertion, maintenance activity, limited resources, overlapping operations, sequence-dependent setup and transportation times and so on.

2. Categories of FJSSP

Based on the number of candidate machines, FJSSP can be divided into two categories: Total FJSSP and Partial FJSSP [16]. In Total FJSSP, each operation can be processed on any machine. In Partial FJSSP, some of the operations can be processed on all machines while the rest of operations can be processed on some of the machines from the machine set. In general, partial FJSSP is more applicable for the real production environment, as some machine cannot be used to process some of the operations.

3. Research approach on FJSSP

There are exact and approximate methods for solving FJSSP. Exact methods guarantee global optimal solutions. However, exact methods do not scale up with problem size, and hence are only suitable for small size problems and tend to be very slow. Approximate methods can obtain feasible solutions quickly but cannot guarantee global optimal solutions. Approximate methods are therefore very suitable for large size problems. They can find satisfactory solutions and solve real industry problems in limited time.

- Exact methods

Mathematical programming is the most used exact method for job shop scheduling problems, for example, Lagrangian relaxation and Decomposition method[17]. Most exact methods are based on a disjunctive graph. Branch-and-bound is also a commonly used method for FJSSP in early research [18]. Branch-and-bound consists of a systematic enumeration of all candidate solutions, where large subsets of fruitless candidates are discarded masse, by using upper and lower estimated bound of the problem being optimized.

- Approximate methods

Many approximate methods are proposed and employed for solving FJSSP in the past two decades. Priority dispatch rule is the earliest approximate method [19]. This method is based on the predetermined operation priority. Priority dispatch rule is simple and easy to realize. First in first out (FIFO) rule, shortest processing time (SPT) first rule and earliest due date first (EDD) rule are the usually employed priority dispatch rules. Some researchers have investigated composite dispatch rules

(CDR) [20]. These CDR are the heuristic combination of single dispatching rules that aim to inherit the advantages of them.

Many meta-heuristics are employed and developed for FJSSP in recent years, for example, Genetic algorithm (GA) [21], ant colony optimization (ACO) [22], particle swarm optimization (PSO) [23], differential evolution (DE) [24], harmony search algorithm (HS) [25], artificial bee colony (ABC) [26], Simulated annealing (SA) and Tabu search (TS) [27] etc. The advantage of meta-heuristic is to obtain multiple approximate solutions quickly.

Local search is also an effective approximate method. Local search is usually used to find local best solutions. Variable local search (VNS) [28] is employed by many researchers to improve the convergence of algorithm for solving FJSSP.

1.2 Motivation and purpose

FJSSP is very important in both academic and application fields. Many researchers focus on FJSSP due to its wide applicability and inherent difficulty. However, most existing literature focuses on the theoretical research, benchmark instance and performance of different methods and algorithms. Few pay attention to the practical environment, special fields and constraints. Hence, how to solve the FJSSP in the practical environment and industry constrains is a challenge to researchers and engineers.

This study is partly supported by Remanufacturing Scheduling System of A*Star (Agency for science, Technology and Research, Singapore) under Grant 112 290 4021.

This thesis modeled constrained FJSSP for remanufacturing industry. Two of the seven characteristics of remanufacturing are considered. The two characteristics are new job insertion, and the uncertain processing time of job operations. These two constraints are related to shop scheduling in remanufacturing environment. Two competitive meta-heuristics, discrete harmony search (DHS) and artificial bee colony (ABC) algorithms, are employed and developed for solving the FJSSP with remanufacturing constraints. The objectives are to minimize the completion time related criteria, machine workload related criteria and due date related criteria. The objectives of this thesis are to improve the production effectiveness of re-processing in remanufacturing industry and to reduce the material and production inventory cost. It can also make the enterprise improve response times and satisfying customer requirements better.

1.3 Original contributions

The major contributions throughout the whole PhD thesis research period are summarized as follows:

- We model the scheduling problem in remanufacturing industry using constrained FJSSP. Two scheduling related constraints, new job insertion and uncertain processing time, are considered as constraints. Rescheduling strategies are proposed for dealing with new job insertion constraints. The most probable processing time and triangle fuzzy number are employed to present the uncertain processing time.
- A discrete harmony search (DHS) algorithm is proposed to solve single objective, multi-objective and Pareto-based multi-objective FJSSP. We are the first to apply DHS algorithm on practical FJSSP with remanufacturing constraints. For FJSSP with two remanufacturing constraints, a two-stage DHS algorithm is proposed to execute scheduling and rescheduling operators. A fuzzy DHS is proposed to solve FJSSP with uncertain processing time. These DHS algorithms are employed to solve benchmark instances and real instances from remanufacturing industry. Many existing algorithms are compared to demonstrate improved performance of proposed DHS algorithms.
- An artificial bee colony (ABC) algorithm is proposed to solve FJSSP with new job insertion for minimizing makespan objective. We are the first to apply ABC algorithm on practical FJSSP with remanufacturing constraints. For fuzzy processing time constraint, a fuzzy ABC algorithm is proposed to minimize fuzzy maximum completion time and fuzzy maximum machine workload. A two-stage fuzzy ABC algorithm is proposed to solve FJSSP with uncertain processing time and new job insertion constraints. These ABC algorithms are compared to many existing algorithms to demonstrate improved performance.
- To improve the quality of initializing solutions, many simple heuristics are employed to initialize population. An effective heuristics, named MinEnd heuristic, is proposed to initialize fuzzy algorithms' population. To test performance, MinEnd heuristic is compared to several simple heuristics. Multiple ensemble heuristics are proposed to solve the FJSSP and compared to proposed DHS and ABC algorithms.

For the feature of FJSSP, two encoding and decoding methods are proposed to represent the solution. Dynamic grouping harmony memory and new crossover method are proposed to improve DHS algorithms performance. An ensemble local search method is proposed to improve the speed of convergence in proposed DHS and ABC algorithms. The extensive experimental comparisons show the effective and efficiency of these operators over the state-of-the-art.

1.4 Organization of the thesis

This thesis is structured into ten chapters, beginning with this introduction.

Chapter 2 presents harmony search algorithm and artificial bee colony algorithm in detail. Based on research approaches and scheduling objectives, we review the literature on solving FJSSP.

Chapter 3 models FJSSP. Two characteristics of remanufacturing industry, new job insertion and uncertain processing time, are modeled as two constraints of FJSSP. Two encoding and decoding methods are proposed to make the FJSSP solution suitable for DHS and ABC algorithms.

Chapter 4 proposes a dynamic group discrete harmony search algorithm for single objective and Pareto-based multi-objective FJSSP. Simple heuristics are employed for initializing population. Harmony memory dynamic grouping and local search methods are proposed to improve algorithm performance. To demonstrate improved performance, the experiments results for 15 benchmark instances are compared to existing algorithms.

Chapter 5 proposes a two-stage artificial bee colony algorithm (TABC) for scheduling FJSSP with new job insertion. An ensemble local search is proposed to improve the convergence of TABC. Three re-scheduling strategies are proposed for new job insertion. The three re-scheduling strategies are compared and discussed. To demonstrate improved performance, TABC are compared to several simple heuristics, hybrid heuristics and DHS algorithm proposed in Chapter 4.

Chapter 6 proposes a two-stage discrete harmony search (TDHS) algorithm for scheduling FJSSP with new job insertion and processing time increasing. Rescheduling strategy is executed when the processing time increases or the new job is inserted into existing scheduling solution. A local search algorithm is employed to speed up the convergence of the proposed TDHS algorithm. Computational results show that the real remanufacturing problems are solved effectively and efficiently. Scheduling and rescheduling results are satisfactory and can be used in practice.

Chapter 7 proposes a fuzzy DHS (FDHS) algorithm for single objective FJSSP with fuzzy processing time. Triangle fuzzy number (TFN) is used to show the uncertainty of processing time. A heuristics, named MinEnd, is proposed to initialize population. Three fuzzy number operators are employed to compare and rank triangle fuzzy number. The proposed FDHS algorithm is compared to six meta-heuristics for five benchmark instances. Eight instances from remanufacturing industry are solved by FDHS algorithm.

Chapter 8 proposes a fuzzy ABC (FABC) algorithm for multi-objective FJSSP with fuzzy processing time. The objectives are to minimize the maximum fuzzy completion time and the maximum fuzzy machine workload. The MinEnd heuristic in Chapter 8 is employed to initialize population. To test performance, FABC algorithm is compared to six meta-heuristics and FDHS algorithm for maximum fuzzy completion time criterion. For maximum fuzzy machine workload, FABC algorithm is compared to six heuristics.

Chapter 9 proposes a two-stage fuzzy ABC (TFABC) algorithm for FJSSP with fuzzy processing time and new job insertion constraints. Eight instances and nineteen insertions are solved by two-stage fuzzy ABC algorithm. The results and comparisons show the effectiveness and efficiency of TFABC algorithm for solving multi-objective FJSSP with multiple constraints.

Chapter 10 presents the conclusions and future work.

Chapter 2

Literature Review

This Chapter reviews the literature for solving the flexible job shop scheduling problem. Harmony search algorithm and artificial bee colony algorithm are introduced and reviewed in detail.

2.1 Research approach

There are exact and approximate methods for solving FJSSP. Mathematical Programming and Branch and bound methods are two major exact methods for shop scheduling problems. There are many different mathematical programming methods, e.g. linear programming, dynamic programming, mixed-integer linear programming and so on. The principle of mathematical programming is to describe the scheduling problem as a mathematical programming model. The problem model is used to obtain the scheduling solutions.

Branch and bound (BB or B&B) is an algorithm design paradigm for discrete and combinatorial optimization problems, such as the job shop scheduling problems. Branch-and-bound consists of a systematic enumeration of candidate solutions by means of state space search: the set of candidate solutions is thought of as forming a rooted tree with the full set at the root. Branch-and-bound explores branches of the tree, which represent subsets of the solution set. Before enumerating the

candidate solutions of a branch, the branch is checked against upper and lower estimated bounds on the optimal solution, and is discarded if it cannot produce a better solution than the best one found so far by the algorithm.

The advantages of exact methods include that they can be employed to analyze most complex optimization problems in real-life, the exact methods can divide the solved problem to small sub-problems and make the solving process easier, the exact method can guarantee the optimal solutions. The exact methods also have some disadvantages. The combinatorial optimization problems and some real-life constraints cannot be described in the programming model. The performance of branch-and-bound is based on the efficient estimation of the lower and upper bounds of a region in the search space.

Exact methods do not scale up with problem size. The computational complexity of exact methods will rapidly growth with the scale of the problems. Exact methods are only suitable for small sized problems and cannot satisfy the real-life requirements for large scale problems.

Compared to exact methods, approximate methods can obtain feasible and acceptable quality solutions quickly. Approximate methods are therefore very suitable for large size problems. They can find satisfactory solutions and solve real industry problems in limited time.

This section mainly reviews approximate methods for solving FJSSP. Approximate methods consist of three classes: 1. Priority dispatch rule, 2. Heuristic and 3. Meta-heuristic. We review related literature based on these three classes, especially the meta-heuristics. For meta-heuristics, we review related literature based on the different algorithm types.

1. Priority dispatch rule

Priority dispatch rules are the earliest approximate method [18]. These rules are based on the predetermined operation priority. Priority dispatch rules are simple and easy to realize. First in first out (FIFO), shortest processing time first (SPT) and earliest due date first (EDD) rules are the usually employed priority dispatch rules. Some researchers have investigated composite dispatch rules (CDR) [19]. These CDR are the combination of single dispatching rules that aim to inherit the advantages of them.

2. Heuristic

Heuristics are simple and can obtain a feasible solution in very limited time. For FJSSP, there are some heuristics for machine assignment, some heuristics for operation sequences and some heuristics for both machine assignment and operation sequence.

1) Heuristics for machine assignment.

- Local minimum processing time heuristic [16]

This heuristic considers the processing time and starts from the operation with the minimum processing. For each operation, the machine with minimum processing time is fixed and assigned.

- Global minimum processing time heuristic [26]

This rule considers both the processing time and the workload of machines and starts from the operation with the global minimum processing time. The processing time is added to the machine workload. For every operation, the machine with the minimum workload is fixed and assigned. The machine's workload is updated. This rule considers the global workload among all machines and can potentially find better makespan.

- Two-step greedy heuristic [29]

The operations are sorted in ascending order based on the number of selectable machines with non-decreasing order of processing times to break ties when there is an equal number of selectable machines. The machines are sorted in their workload non-decreasing order. One operation is taken from the operations list and the first machine that belongs to the machine list is assigned to the operation. The workload of this machine is updated and the machine ranking is also updated. The process iterates until all operations have been assigned to machines.

- Minimum completion time heuristic [123]

This rule is based on the local minimum processing time heuristic. Two machines are selected for one operation that has more than one candidate machines. One machine has the minimum processing time and another machine has the earliest start time. The completion time on these two machines are calculated. The machine with the smaller completion time is fixed and assigned.

2) Heuristics for operation sequences

- Most work remaining [26,30]

This method firstly orders the jobs in a descending order based on the remaining processing time. The job with the most remaining processing time will be selected first and put into the operation sequence. The iteration is repeated until all operations of all jobs are sequenced.

- Most operation remaining [26,30]

This heuristic selects jobs based on the number of remaining operations. The job with the most remaining operations unprocessed is selected first.

- Shortest processing time (SPT) [30]

In this approach, the operation with the shortest processing time is selected from the next executable operations.

3) Heuristic for both machine assignment and operation sequences

- Newton-based hierarchical heuristic [31]

In this heuristic, an adaptation is made to consider discrete variables and also to redefine the stopping criterion. Each new solution improves simultaneously all the objectives till the stopping criterion is met.

- Greedy heuristic [32]

This heuristic considers jobs one after another according to a fixed job sequence. The operation sequence is fixed by applying the polynomial algorithm. The machine assignment is based on priority rule.

- Filtered-beam-search based heuristic [33,34]

Filtered beam search is an extension of beam search. The filtered beam search procedure uses both crude and accurate evaluations in two phases: filtering phase and beam selection phase. A successful filtered beam search based heuristic for FJSSP should solve four major elements:

  - Search tree representation to define a solution space.
  - Determination of beam width and filter width.
  - Branching scheme.
  - Local and global evaluation functions selection.

- Constructive procedure based heuristic [35,36]

This approach is motivated by the idea of developing a constructive heuristic that considers simultaneously many factors affecting the solution quality and intelligently balances their effects. This algorithm has a simple structure and great flexibility, is easy to implement, and requires very little computational time.

- Variant of dispatch heuristic [37]

The dispatch heuristic considered two variants: (i) an ordered variant, where the priority dispatching rules were applied in a predefined order, and (ii) a randomized variant, where the user could assign probabilities (weights) to the priority rules. Using the information of the number of units and due dates requested by the customer, the algorithm provided the sequence of operations that must be performed on each machine, as well as the start and completion times of operations. In order to reduce the impact of unexpected events on a generated schedule, several robust rules were considered. The obtained results substantially improved the former method used in the company in terms of minimizing average tardiness. Additionally, other important benefits were obtained, including significant saving in the time spent on scheduling, simplicity of use of the proposed procedure, robustness against unexpected events, reduction of idle times, improvement of decision-making information, and improvement of on-time delivery performance.

- Combination heuristic [38]

In the combination heuristic, a simulation based on earliest due date (EDD), the operations' lowest level code (LLC) of the bill of materials and the longest processing time (LPT) is proposed. A real-life weapons production factory was used as a case study to evaluate the performance of the proposed algorithm. Due to the high penalty of delays in military orders, the on-time delivery rate was the most important performance measure and makespan was the next most important measure.

- Discrepancy search heuristic [39]

In the heuristic, a variant of the climbing discrepancy search approach was developed for solving FJSSP. Various neighborhood structures related to assignment and sequencing problems were employed to enhance the results quality.

- Mixed integer goal programming model [40]

Two mixed integer goal programming models are formulated for FJSSP separable/non-separable sequence dependent setup times. In the first model, the sequence dependent setup times were non-separable. In the second one, they were separable. The second model was obtained from the first one with a minor modification. The formulation of the models was described on a small sized numerical example and the solutions were interpreted. Finally, computational results were obtained on test problems.

- Other heuristics

He W [41] proposed a multi-strategies based approach to schedule FJSSP with machine breakdown. In pre-scheduling, an idle time insertion strategy was put forward. In the policy, an idle time equal to repair time was inserted into an appropriate position of each machine according to the machine's breakdown nature. A route changing strategy combined with right-shift policy was proposed to keep the rescheduling as stable and robust as possible. In the policy, whether to right shift or route change dependent on the cost of archiving robustness and stability. Based on the two strategies, algorithms dealing with idle time insertion, right-shift scheduling, and route changing scheduling were designed. The computational results show the effectiveness of the strategies and algorithms compared with other strategies. Also to solve FJSSP with machine breakdown, Sun HD [42] proposed a game theory based approach. In the approach, pre-scheduling without considering machine failure was generated. The operations affected by machine failure were obtained according to time at which machine breakdown happened. The decision made by each side obtained better robust and stable performance. To achieve the optimal objects, Nash Equilibrium (NE) solution for each side was achieved. Considering NE solution might not exist or there were more than one NE solutions, concepts of ideal NE solution and near NE solution were proposed. To avoid huge decision searching space when the number of operations was more than two, an NE searching algorithm under multi-stage game based on the two concepts was also proposed to find NE or near NE solutions.

3. Meta-heuristics

Many meta-heuristics are employed and developed to solve FJSSP in recent years. This section will mainly review the literature based on different meta-heuristics.

- Genetic algorithm (GA)

GA is a widely used algorithm to solve FJSSP. Chan FTS [43] developed a GA based approach to solve FJSSP with resource constraints. Using the approach, a resource-constrained operation machines assignment problem and flexible operation sequencing problem could be solved iteratively. In the connection, the flexibility embedded in the flexible shop floor, which was important to today's manufacturers, could be quantified under different levels of resource availability. Ho NB [44] proposed a GA based architecture (LEGA) for learning and evolving FJSSP. Unlike the canonical evolution algorithm, random elitist selection and mutational genetics were assumed through LEGA. The knowledge extracted from the previous generation was used to influence the diversity and quality of offsprings. In addition, the architecture specified a population generator module that generated the initial population of schedules and trained the schemata-learning module. Experimental results indicated that an instantiation of LEGA outperformed the compared approaches in computational time and quality of schedules. Gao J. [30] proposed a GA algorithm with a local search based on criteria path to solve FJSSP with maintenances.

Pezzella F [16] studied on a genetic algorithm with different initial strategies to solve FJSSP. The algorithm integrated different strategies for generating the initial population, selecting the individuals for reproduction and reproducing new individuals. De Giovanni L [45] proposed an improved GA to solve the distributed and flexible job shop scheduling problem. A greedy decoding procedure exploited the flexibility and determined the job routings. Besides traditional crossover and mutation operators, a new local search based operator was used to improve available solutions by refining the most promising individuals of each generation. Mati Y [46] proposed a GA for practical FJSSP with blocking constraints. The model and algorithm were used to a company to design a new production workshop. Numerical experiments using real data were presented and analyzed. The authors showed how these results were used to support choices in the design of the workshop.

Ai-Hinai N [47] addressed a hybrid GA to solve robust and stable FJSSP with random machine breakdown. The first stage optimized the primary objective, minimizing makespan in the literature, where all the data was considered deterministic with no expected disruptions. The second stage optimized the bi-objective function and integrated machines assignments and operations sequencing with the expected machine breakdown in the decoding space. An experimental study and analysis of variance was conducted to study the effect of different proposed measures on the performance of the

obtained results. Results indicated that different measures had different significant effects on the relative performance of the proposed method. Chen James C. [48] proposed a GA and grouping GA for FJSSP with parallel machines. Gholami M. [49] researched on an integrating simulation GA for scheduling dynamic FJSSP. To solve multi-objective FJSSP, Rahmati SHA [50] studied a multi-objective genetic algorithm. In the study, non-dominated ranking are adapted. Chiang TC [51] addressed a simple and effective GA for scheduling FJSSP which utilized effective genetic operators and maintains population diversity carefully. A main feature of the algorithm was its simplicity. It needs only two parameters.

For FJSSP with overlapping in operations, Demir Y [52] presented an effective two-steps GA. Zhang GH [53] proposed a GA with high quality initial population to solve FJSSP. Global selection and local selection were designed to improve the quality of the initial population. Wang XJ [54] presented a multi-objective genetic algorithm (MOGA) for FJSSP. The algorithm was based on immune and entropy principle. MOGA applied Pareto-optimality to fitness scheme. To solve FJSSP with machine disruption, Wang YM [21] created a novel genetic algorithm. In the algorithm, a very special chromosome encoding was designed to adapt to disruption. The algorithm was compared to a right-shifting re-scheduler and a pre-scheduler algorithm. For FJSSP with fuzzy processing time, Lei DM [55] proposed a co-evolutionary genetic algorithm (CGA). A new crossover operator and a modified tournament selection are proposed in the study. To solve FJSSP with sequence dependent setups, Defersha FM [56] presented a parallel genetic algorithm. The algorithm runs on a parallel computing platform. Numerical examples showed that parallel computing could greatly improve the computational performance of the algorithm.

- Ant colony optimization (ACO)

Many researchers employed and developed ACO to solve FJSSP. Rossi A [57] proposed an ACO algorithm for FJSSP with routing flexibility and separable setup times. The optimization problem for a real environment, including parallel machines and operation lag times, was approached by means of an effective pheromone trail coding and tailored ant colony operators for improving solution quality. The method used to tune the system parameters was also described. The algorithm was tested by using standard benchmark problems, properly designed for a typical FMS layout. The effectiveness of the proposed system was verified in comparison with alternative approaches. Xing LN [22] proposed a knowledge-based ACO (KBACO) algorithm for FJSSP. The algorithm integrated ACO and a knowledge model. The knowledge-based system learned available knowledge from ACO and applied the existing knowledge to guide the searching. The authors also designed and evaluated an ACO based simulation model to solve multi-objective FJSSP. Yao BZ [58] presented an improved ACO algorithm for FJSSP. Three improved strategies, transition rule with adaptive parameters, crossover operation

and pheromone updating strategy with adaptive parameters were introduced to enhance the performance of ACO. For multi-objective FJSSP, Luo DL [59] proposed an ACO algorithm which combined SPT rule and local search method to improve performance. Rossi A [60] researched an ACO algorithm with reinforced pheromone relationships to solve FJSSP with sequence-dependent setup and transportation time. Huang RH [61] developed a two pheromone ACO (2PH-ACO) to solve FJSSP. Computational results indicated that 2PH-ACO performed better than ACO in terms of sum of earliness and tardiness time.

- Particle swarm optimization (PSO)

Boukef H [62] proposed an algorithm inspired from PSO for FJSSP with makespan criterion. The experiments and discussions showed that the algorithm was better than the compared GA algorithm. Liu HB [63] used a multi-swarm PSO algorithm to solve multi-objective FJSSP. The authors theoretically proved that the multi-swarm synergetic optimization algorithm converged with a probability of one towards the global optima. To solve Pareto-based multi-objective FJSSP, Moslehi G. [64] proposed a particle swarm optimization with local search. The results indicated that the algorithm satisfactorily captured the multi-objective flexible job-shop problem and competed well with the compared approaches. Nourali S [65] employed a PSO based algorithm for flexible assembly job shop scheduling problem with sequence dependent setup times. The literature focused on real industry application. Practice industry instances were solved to demonstrate the effectiveness of the algorithm.

- Differential evolution (DE)

To solve FJSSP, Yuan Y [66] proposed a hybrid differential evolution (HDE). In the algorithm, a conversion mechanism was developed to make the DE that worked on the continuous domain adaptive to explore the discrete problem space of the FJSSP. A local search algorithm based on the critical path was embedded in the HDE framework to balance the exploration and exploitation by enhancing the local searching ability. In addition, a speed-up method to find an acceptable schedule within the neighborhood structure was presented to improve the efficiency of whole algorithms. In addition, Xu XL [67] proposed a hybrid discrete differential evolution (HDDE) algorithm for lot splitting with capacity constraints in flexible job scheduling. The HDDE algorithm also employed criteria path based local search to improve performance. The simulation results showed that the HDDE algorithm could effectively solve the practical lot splitting problems with equipment capacity constraints.

- Simulated annealing (SA)

A few papers use just SA to solve FJSSP. Fattahi P [68] proposed a SA based hierarchical approach to solve large-scale FJSSP instances. To evaluate the validity of the proposed SA algorithm, the results

were compared with the optimal solution obtained by the Branch and Bound method. The authors also prove that the SA based algorithm could be applied easily to real factory conditions. SA is used in many hybrid algorithms for FJSSP. The literature for these hybrid algorithms will be reviewed in hybrid algorithms.

- Tabu search (TS)

TS is also a widely used meta-algorithm for FJSSP. Logendran R [69] proposed six TS-based meta-heuristics (TSH1-TSH6). The results showed that, as the problem size increases, TSH3 with fixed tabu-list size and long-term memory was preferred over the other heuristics. Further, the branch-and-bound technique, by failing to identify as goad a solution as that determined by the heuristics (TSH I-TSH 6), let alone an optimal solution, for a small problem reinforced the need for developing efficient heuristics for solving real problems encountered in industry practice. Saidi-Mehrabad M. [70] presented a TS algorithm that composed of two parts: a procedure that searched for the best sequence of job operations, and a procedure that found the best choice of machine alternatives. These two parts were used to model FJSSP using meta-heuristics. Computational results indicated that the proposed algorithm could produce optimal solutions in a short computational time for small and medium sized problems. Moreover, it could be applied easily in real factory conditions and for large size problems. For multi-objective FJSSP, Li JQ [71,72] proposed two TS algorithms, HTS and TSPCB, for solving FJSSP. HTS combined with two adaptive rules. A left-shift decoding method was employed to transform a solution to an active schedule. TSPCB combined with critical block neighborhood structure. Insert and swap neighbor structures were used on a critical path to perform local search in HTS and TSPCB algorithms. The two algorithms were tested on sets of benchmark instances and compared to several existing algorithms. Vilcot G [29] proposed two TS algorithms for multi-objective FJSSP. The first was based on the minimization of one criterion subject to a bound on the second criterion and the second was based on the minimization of a linear combination of criteria. Jia S [73] proposed a path-relinking tabu search for the multi-objective flexible job shop scheduling problem. A path-relinking heuristic was designed to generate diverse solutions in the most promising areas. An effective dimension-oriented intensification search was employed to find solutions that were located near extreme solutions.

Eshlaghy AT [74] proposed TS based algorithm for flexible job shop manufacturing systems. The algorithm was compared to a hierarchical method. Performance criteria for comparison are "response quality" and "calculation time" in which results of numerical test approved the priority of TS search algorithm in comparison to the hierarchical method. TS is used in some hybrid algorithms for FJSSP. The literature for these hybrid algorithms will be reviewed in hybrid algorithms.

- Estimation of distribution algorithm (EDA)

To solve FJSSP, Wang L and Wang SY [75,76,77] proposed three EDA based algorithms, bi-population based EDA algorithm (BEDA), Pareto-based EDA algorithm (PEDA) and an EDA for fuzzy processing time constraints. In BEDA algorithm, two sub-populations were used to adjust the machine assignment and operation sequencing respectively with a splitting criterion and a combination criterion. In PEDA algorithm, the fitness evaluation based on Pareto optimality was employed and a probability model was built with the Pareto superior individuals for estimating the probability distribution of the solution space. Both BEDA and PEDA algorithms employed critical path based local search operations to enhance performance. Triangle fuzzy number and three fuzzy number operations were used to calculate scheduling objective value in the EDA for FJSSP with fuzzy processing time. In addition, Taguchi method was employed to investigate the influence of parameter setting in all the three EDA based algorithms. Three algorithms were tested though comparing to existing algorithms.

- Variable neighborhood search (VNS)

Various neighborhood structures related to machine assignment and operation sequencing problems were used for generating neighboring solutions. Amiri M. [78] proposed a VNS algorithm for FJSSP. To compare the algorithm with previous ones, an extensive computational study on 181 benchmark problems had been conducted. The results obtained from the presented algorithm were quite comparable to those obtained by the compared algorithms. Yazadani M. [79] proposed a parallel variable neighborhood search algorithm (PVNS). Parallelization in this algorithm was based on the application of multiple independent searches increasing the exploration in the search space. The proposed PVNS used various neighborhood structures that carry the responsibility of making changes in machine assignment and operation sequencing for generating neighboring solutions.

To solve FJSSP with sequence-dependent setup times, Bagheri A. [80] proposed an integrated approach based VNS algorithm. In the algorithm, the external loop controlled the stop condition of the algorithm and the internal loop executed the search process. To evaluate the performance of the proposed algorithm, 20 test problems in different sizes were randomly generated. Consequently, computational results and comparisons validated the quality of the proposed approach. Among the above VNS algorithms, systematic changes of neighborhood structure were generated for evading local optimum. The search processes for finding a local or global optimum solution by VNS are very random. This is one of the weaknesses of this algorithm. To remedy this weakness of VNS, Karimi H. [81] combined VNS algorithm with a knowledge module and proposed a knowledge-based VNS (KBVNS). In KBVNS, the VNS part searched the solution space to find good solutions and knowledge module extracted the knowledge of good solution and feed it back to the algorithm. This would make

the search process more efficient. Computational results of the paper on different size test problems proved the efficiency of KBVNS algorithm for FJS problem.

In addition, Lei DM [82] presented a VNS algorithm for FJSSP with a dual-resource constraint. In the algorithm, the solution to the problem was indicated as a quadruple string of the ordered operations and their resources. Two neighborhood search procedures were sequentially executed to produce new solutions for two sub-problems of the problem, respectively. The search of VNS was restarted from a slightly perturbed version of the current solution by VNS when the determined number of iterations was reached. This VNS was tested on some instances and was compared with methods from existing literature. Computational results showed the significant advantage of VNS on the problem.

- Hybrid algorithm

To improve algorithm performance, many hybrid algorithms based on more than one meta-algorithm were proposed to solve FJSSP. Tanev IT [83] researched a real-world FJSSP, customers' order in factories of plastic injection machines. A hybrid evolutionary algorithm (HEA) was developed in the literature. HEA combined priority-dispatching rules and GA. The results obtained for evolving a schedule of 400 customers' orders on experiment indicated that the business delays in the order of half-an-hour could be achieved. Fattahi P [84] proposed two types of approaches, hierarchical approaches and integrated approaches to solve FJSSP. Two heuristics involving integrated and hierarchical approaches were developed to solve the real size problems. Six different hybrid-searching structures depending on the used searching approach and heuristics were presented in the paper. Frutos M [85] proposed an algorithm based on non-dominated sorting genetic algorithm II (NSGAII) and SA to solve FJSSP.

Roshanaei N [86] proposed a hybrid of artificial immune and SA (AISA) algorithm for larger instances of FJSSP. The AISA was compared to seven existing algorithms and a developed integer linear programming model. An industrial problem in a mould- and die-making shop was used to verify the AISA algorithm. A hybrid genetic algorithm and SA (GASA) was proposed by Shahsavari-Pour N. [87] for multi-objective FJSSP. Contrary to the other methods that assign weights to all objective functions to reduce them to one objective function, in the GASA and during all steps, problems are solved by three objectives. Gao J [88] researched a hybrid genetic algorithm and bottleneck shifting for multi-objective flexible job shop scheduling problems. Gao J [28] also proposed a hybrid genetic and variable neighbor search algorithm to solve FJSSP. The variable neighbor search algorithms were based on criteria path. Xing LN [89] proposed a multi-population interactive co-evolutionary algorithm for FJSSP. In the study, both ant colony optimization and genetic algorithm with different configuration were applied to evolve each population independently. For FJSSP with transportation constraints and bound processing times, Zhang Q [90] proposed a hybrid genetic algorithm with tabu

search procedure. A hybrid algorithm of gravitational search and colored petri-net (GSPN) was addressed by Barzegar B [91] to solve FJSSP. In GSPN, the flexible job-shop scheduling problem systems were modeled by color Petri net and color Petri net tool. A scheduled job was programmed by GSPN algorithm. The experimental results showed that the proposed method has reasonable performance in comparison with other algorithms. Liouane N [92] described an ACO and tabu search based local search algorithm to solve FJSSP with makespan objective.

To solve multi-objective FJSSP, Xia WJ [93] researched a hybrid optimization based on PSO and SA. The results obtained from the computational study had shown that the proposed algorithm was a viable and effective approach for the multi-objective FJSSP, especially for problems on a large scale. Shao XY [94] proposed a hybrid discrete PSO (DPSO) and SA algorithm for multi-objective FJSSP. Pareto ranking and crowding distance method were incorporated to identify the fitness of particles in the algorithm. SA was used as a local search algorithm in the above two hybrid algorithms. In addition, to solve multi-objective FJSSP, Zhang GH [53] proposed a hybrid PSO and tabu search algorithm. Three above PSO based hybrid algorithm were proved for multi-objective FJSSP, especially for the problems on a large scale.

Araghi MET [95] proposed an incorporating learning hybrid meta-heuristic based on GA algorithm and VNS method for FJSSP. An affinity function was used to enhance algorithm performance. Taguchi's robust design method was employed to define the best parameter values. Some experiments were designed where the proposed method was compared against some new and successful GA and VNS algorithms. Sadrzadeh A [96] developed an artificial immune system (AIS) and PSO algorithm for scheduling FJSSP. The algorithms were compared to the existing GA algorithm and PVNS. The obtained results showed that the proposed PSO outperforms the GA and the PVNS approaches. It was found that the average best-so-far solutions obtained from the proposed AIS were better than those produced by the GA, the PVNS, the VNS, and the PSO algorithms for all the examples. To solve FJSSP with parallel machine and maintenance constraints, Dalfard VM [97] developed two meta-heuristics based on GA and SA. The two meta-heuristics were compared to software LINGO scheduling system. The results showed that the applied hybrid GA and SA algorithms were much more effective than the solutions obtained using LINGO. Finally, solutions using the simulated annealing approach were compared with solutions of the hybrid genetic algorithm.

- Others algorithms

There are some other algorithms for FJSSP. To solve FJSSP, Wu ZB [98] proposed a multiagent scheduling method with earliness and tardiness objectives. A job-routing and sequencing mechanism was proposed. In the mechanism, two kinds of jobs were defined to distinguish jobs with one operation left from jobs with more than one operation left. Different criteria were proposed to route these two

kinds of jobs. Job sequencing enables to hold a job that might be completed. Two heuristic algorithms for job sequencing were developed to deal with these two kinds of jobs. The computational experiments showed that the proposed multi-agent scheduling method significantly outperforms the existing scheduling methods in the literature.

Yu XF [99] addressed a bio-inspired multi-agent model for FJSSP with sequence dependent setups. The study set the model parameters using two steps, termed mapping and tuning. Mapping established a set of coefficients to link the model parameters with the scheduling problem characteristics. Tuning determined good values of these coefficients were used to compute the model parameters. The performance of the proposed multi-agent model was compared with another scheduling method that was based on dispatching rules. It was concluded that the proposed parameter setting method was effective and worth considering when applying bio-inspired division of labor to dynamic manufacturing scheduling.

To solve FJSSP with alternative process plans, Rajabinasab A [100] proposed an agent based scheduling system. A pheromone-based approach was proposed for coordination among agents. The approach was compared with five dispatching rules from literature via simulation experiments to statistical analysis. The simulation experiments were performed under various experimental settings such as shop utilization level, due date tightness, breakdown level, and mean time to repair. To solve FJSSP with machine availability constraints, Moradi E [101] established an architecture. The literature applied an enforceable and easily extendable preventive maintenance policy on FJSSP. In order to figure the problem out, all operators and parameters were calibrated by means of the Taguchi experimental design.

Baghei A [102] addressed an artificial immune algorithm (AIA) for FJSSP. AIA used several strategies for generating the initial population and selecting the individuals for reproduction. Different mutation operators were utilized for reproducing new individuals. To show the effectiveness of AIA, numerical experiments by using benchmark problems were conducted. Consequently, the computational results validated the quality of AIA.

Karthikeyan S. [103] proposed a discrete firefly algorithm for multi-objective FJSSP. The machine assignment and operation sequence were processed by constructing a suitable conversion of the continuous functions as attractiveness, distance and movement, into new discrete functions. Meanwhile, local search method with neighborhood structures is hybridized to enhance the exploitation capability.

A hybrid SFLA algorithm was proposed by Li JQ [104] to solve multi-objective FJSSP. In the algorithm, several approaches were presented to construct the initial population with a high level of quality. Two crossover operators were presented to share information among the best frogs and the

worst frog. Several local search methods were embedded in the algorithm to enhance the exploitation capability. Li JQ [105] also proposed a discrete chemical-reaction optimization (DCRO) for FJSSP with maintenance activity. In the DCRO, each solution was represented by a chemical molecule. Four improved elementary reactions, on-wall ineffective collision, inter-molecular ineffective collision, decomposition, and synthesis, were developed. A well-designed crossover function was introduced in the inter-molecular collision, synthesis, and decomposition operators. In addition, the decoding mechanism considering the maintenance activity was presented. Several neighboring approaches were developed to improve the local search ability of the DCRO. Through the analysis of experimental results, the highly effective performance of the proposed DCRO algorithm was shown.

Rajkumar M [106,107] proposed two greedy randomized adaptive search procedure (GRASP) to solve FJSSP with maintenance constraints and limited resource constraints. The two GRASP algorithms were meta-heuristic algorithms that were characterized by multiple initializations. It operated in the following manner: first a feasible solution was obtained, which was then further improved by a local search technique. Representative benchmark problems were solved in order to test the effectiveness and efficiency of the two GRASP algorithms.

Farughi H [108] proposed a memetic algorithm (MA) for FJSSP with overlapping operations. A heuristic used the critical path method (CPM) in order to improve the results of MA and reduce the objective function. The experimental results showed that MA was capable of achieving the optimal solution for small size problems and near-optimal solutions for medium and large size problems in a reasonable time.

In addition, Nie L [109] studied a heuristic to implement reactive scheduling for the dynamic scheduling problem. A gene expression programming (GEP) algorithm was developed for scheduling FJSSP with dynamic job releasing date constraint. The GEP automatically constructed reactive scheduling policies for the dynamic scheduling. A variety of processing conditions three factors, such as the shop utilization, due date tightness, problem flexibility, were considered in the simulation experiments. The results showed that GEP-based approach could construct more efficient reactive scheduling policies for DFJSSP with job release dates under a big range of processing conditions.

Rahmati SHA [110] proposed a Biogeography-based optimization (BBO) algorithm for FJSSP. The migration operators of BBO were developed for searching a solution area of FJSSP and finding the optimum or near-optimum solution to this problem. To assess the performance of BBO, it was compared with a genetic algorithm that had the most similarity with the proposed BBO. This similarity caused the impact of different neighborhood structures being minimized and the differences among the algorithms being just due to their search quality. To evaluate the distinctions of the two algorithms

much more elaborately, they were implemented on three different objectives. BBO was also compared with some famous algorithms in the literature.

2.2 Harmony search algorithm

Harmony search (HS) is one of the recent meta-heuristic methods presented by Geem et al [111] for solving optimization problems. It imitates the improvisation process of musicians. The harmony in music is analogous to the solution vector in the corresponding optimization problem, and the musician's improvisations are analogous to local and global search in optimization techniques. The harmony search algorithm generates a new candidate solution from all existing solutions. Compared to the earlier meta-heuristics, the HS algorithm imposes fewer mathematical requirements. In addition, numerical comparisons show that the process of evolution in harmony algorithm is faster than that of the GA [112]. The authors of the literature improved the HS performance. They changed the parameters pitch adjusting rate (PAR) and bandwidth (bw) for each generation from fix values to dynamic values with different generations. The results for benchmark instance and standard engineering optimization problems show that harmony search can find better solutions when compared to other heuristics or deterministic methods. Harmony search algorithm is also a powerful search algorithm for various engineering optimization problems.

The computational procedure of the basic single objective HS algorithm can be summarized as follows:

Step 1: Initialization

The harmony vector or solution is generally represented as an *n*-dimensional real-valued vector $X = \{x(1), x(2), \ldots, x(n)\}$, where each dimension $x(k)$ denotes a decision variable of the optimization problem. In this step, the aims are to set parameters and fill the harmony memory (HM) with a population of initial harmonies (solutions). Parameters include the harmony memory size (HMS), harmony memory consideration rate ($HMCR$), pitch adjustment rate (PAR) and distance bandwidth (BW). The HM is filled by randomly generated harmony vectors.

Let $X_i = \{x_i(1), x_i(2), .., x_i(n)\}$ represent the i^{th} harmony vector in the HM, which is generated as follows:

$$x_i(j) = LB(j) + (UB(j) - LB(j)) \times rand() \tag{2-1}$$

for $j = 1,2,..,n$ and $i = 1,2,\cdots HMS$

where $rand()$ is a uniform random number in the range of [0,1], $LB(j)$ and $UB(j)$ are the lower and upper bounds for the decision variable $x(j)$, respectively.

Step 2: Improvising a new harmony

The process to produce a harmony vector X_{new} is called improvisation. X_{new} is produced by applying three rules, i.e., a memory consideration, a pitch adjustment and a random selection. First of all, if a uniform random number $rand()$ in the range of [0, 1] is less than $HMCR$, the decision variable $x_{new}(j)$ is generated by the memory consideration, otherwise, $x_{new}(j)$ is obtained by a random selection. Secondly, each decision variable $x_{new}(j)$ will undergo a pitch adjustment with a probability of PAR if it is updated by the memory consideration. The memory consideration, pitch adjustment and random selection are given in Eqs (3-2), (3-3) and (3-4), respectively.

$$if(rand1() < HMCR)$$

$$x_{new}(j) = x_a(j) \tag{2-2}$$

$$if(rand2() < PAR)$$

$$x_{new}(j) = x_{new}(j) \pm rand() \times BW \tag{2-3}$$

$$else$$

$$x_{new}(j) = LB(j) + rand() \times (UB(j) - LB(j)) \tag{2-4}$$

where $a \in (1,2,..,HMS)$, $x_a(j)$ is randomly selected from HM.

Step 3: Updating the HM

The HM will be updated if X_{new} is better than the worst harmony vector X_w in the HM. In this case, X_{new} will replace X_w and become a new member of the HM.

Step 4: Stopping condition

If a termination criterion is met, return the best harmony vector X_B in the HM; otherwise go to Step 2.

Many researchers have developed HS algorithm and have applied HS variants to solve industry optimization problems [113]. The authors of the literature employed HS algorithm for solving discrete structural optimization problem. The numerical results showed HS could yield better solutions than those obtained by existing methods. Omran and Mahdavi [114] developed a global-best harmony search algorithm. In the paper, the authors modified the pitch-adjustment steps so that the new harmony can mimic the best harmony in harmony memory. The BW parameter was replaced altogether and added a social dimension. The global-best HS algorithm could work efficiently on both continuous and discrete problems. Das [115] analyzed and improved the exploration power of HS algorithm. These papers analyzed the evolution of the population-variance over successive generations in HS and draw some important conclusions regarding the explorative power of HS. Geem applied harmony search algorithm for scheduling multiple dam system [116]. HS algorithm tackled popular

benchmark system with four dams. The HS algorithm could find more global optimal solutions than the compared GA algorithm in the literature. Furthermore, HS model arrived at the global optima without any sensitivity analysis of algorithm parameters whereas the compared GA model required tedious sensitivity analysis.

In recent years, HS are also employed for solving shop scheduling problem. Wang [117] and Pan [118] proposed a hybrid harmony search algorithm and a local-best harmony search algorithm with dynamic sub-harmony memories for blocking flow shop scheduling problem and lot-streaming flow shop scheduling problem. Gao proposed a grouping harmony search algorithm [119] and a discrete harmony search (DHS) algorithm [120] for no-wait flow shop scheduling problem. In these shop scheduling problem papers, some improving strategies for HS were developed, for example, sub-harmony memory for convergence of HS, local-best search for exploration. Based on the shop scheduling problem features, some simple heuristics or dispatch rules were employed in HS algorithm to improve the quality of initial solutions, for example, NEH heuristic. To improve the local search performance, HS is mixed with several local search operators that are famous in shop scheduling problem, for example insert operator, exchange operator and reverse operator etc.

To solve FJSSP and large-scale FJSSP, Yuan Y [121,122] proposed two harmony search based algorithms. In the proposed HS algorithms, HS was converted to adapt to FJSSP. Local search based on common critical operation was developed to enhance performance. For the large-scale FJSSP, large neighborhood search was integrated to HS to improve results. The HS algorithm for large-scale FJSSP had even found some new upper bounds among the solved benchmark instances. The author of this thesis proposed a discrete harmony search algorithm (DHS) [25] and a Pareto-based grouping discrete harmony search (PGDHS) algorithm [123]. A proposed heuristic and several existing simple heuristics were employed to initialize harmony memory in the two algorithms. A new rule was developed for improvisation a new harmony. In PGDHS algorithm, dynamic grouping was used to improve population diversity. Three widely used performance measures, a number of non-dominated solutions, diversification metric and quality metric, were employed to test the performance of PGDHS algorithm.

2.3 Artificial bee colony

Artificial Bee Colony (ABC) is one of the most recently defined algorithms by Dervis Karaboga in 2005, motivated by the intelligent behavior of honey bees [124]. It is simple and uses only common control parameters such as colony size and maximum cycle number. ABC as an optimization tool provides a population-based search procedure in which individuals are called foods positions and are modified by the artificial bees with time. In ABC, the colony of artificial bees contains three groups

of bees: employed bees associated with specific food sources, onlooker bees watching the dance of employed bees within the hive to choose a food source, and scout bees searching for food sources randomly. The bee's aim is to discover the places of food sources with high nectar amount and to find the one with the highest nectar. In ABC system, artificial bees fly around in a multidimensional search space and some (employed and onlooker bees) choose food sources depending on the experience of themselves and their nest mates, and adjust their positions. Some (scouts) fly and choose the food sources randomly without using experience. If the nectar amount of a new source is higher than that of the previous one in their memory, they memorize the new position and forget the previous one. Thus, ABC system combines local search methods, carried out by employed and onlooker bees, with global search methods, managed by onlookers and scouts, attempting to balance exploration and exploitation process.

The main steps of the basic ABC algorithm are as followings.

1) Initialization of the parameters and population:

   The parameters of ABC are the number of food sources (SN), the number of trials after which a food source is to be abandoned (*limit*) and the termination criterion. The number of food sources is equal to the number of employed bees or onlooker bees. The initialization of population is to fill the population with SN number of randomly generated food sources, *n*-dimensional real-valued vectors. Let $X_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$ represent the *i*th food source in the population. The food sources are generated as follows:

$$x_{ij} = LB_j + (UB_j - LB_j) \times r \text{ for } j = 1,2,...,n \text{ and } i = 1,2,...,SN \tag{2-5}$$

where r is a uniform random number in the range [0, 1]; LB_j and UB_j are the lower and upper bounds for the dimension j, respectively. The food sources are randomly assigned to employed bees and the corresponding fitnesses are evaluated.

2) Employed bee phase:

   In this phase, each employed bee X_i generates a new food source X_{new} in the neighborhood of its present position as follows:

$$x_{new(j)} = x_{ij} + (x_{ij} - x_{kj}) \times r' \tag{2-6}$$

   where $k \in \{1,2,...,SN\} \wedge k \neq i$ and $j \in \{1,2,...,n\}$ are randomly chosen indexes. r' is a uniformly distributed real number in [-1, 1].

   X_{new} will be compared to X_i. If the fitness of X_{new} is equal or better than that of X_i, X_{new} will replace X_i as a new food source; otherwise X_i is retained.

3) Onlooker bee phase:

An onlooker bee evaluates all the employed bees and selects a food source X_i depending on its probability value p_i calculated by the following expression.

$$p_i = \frac{f_i}{\sum_{i=1}^{SN} f_i} \tag{2-7}$$

where f_i is the nectar amount or the fitness value of the *i*th food source X_i. The higher the f_i is, the more ability that the *i*th food source is selected.

Once the food source X_i is selected, the onlooker bee will execute the update X_i using equation (3). If the new food source has equal or better fitness value than X_i, the new food source will replace X_i as a new member in the population.

4) Scout bee phase:

If a food source X_i can not be improved through a predetermined number of trials *limit*, the food source is to be abandoned and the corresponding employed bee becomes a scout. The scout produces a new food source randomly as follows:

$$x_{ij} = LB_j + (UB_j - LB_j) \times r \text{ for } j = 1,2,...,n \tag{2-8}$$

where r is a uniform random number in the range [0, 1].

5) Repeat steps 2)-4) until the termination condition is satisfied.

In recent years, ABC algorithm has received intensive interest from researchers in a variety of fields. It was first proposed to solve the multi-variable and multi-modal continuous functions [124]. Many comparative studies have shown that the performance of the ABC algorithm is competitive to other population-based algorithms with the advantage of employing fewer control parameters in the continuous space [125,126, 127,128]. Banharnsakun, Achalakul and Sirinaovakul adopted the best-so-far selection in ABC to enhance the exploitation and exploration processes [129].

ABC is used to solve combinatorial optimization problems. Wong, Low, and Chong developed a bee colony optimization with a local search to solve the travelling salesman problem [130]. For shop scheduling problem, a discrete artificial bee colony algorithm based on a self-adaptive strategy was proposed to solve the lot-streaming flow-shop scheduling problem with the criterion of total weighted earliness and tardiness penalties [131]. Banharnsakun, Sirinaovakul, and Achalakul applied the best-so-far ABC (B-ABC) to solve the JSSP problem [132].

In recent years, ABC algorithm is developed for FJSSP by many researchers. Li JQ [26,133] proposed an ABC algorithm and a Pareto-based discrete ABC algorithm for multi-objective FJSSP. Pareto archive set was employed to record non-dominated solutions in the two algorithms. Several local

search approaches were designed to balance the exploration and exploitation capability of the algorithm in discrete ABC algorithm. Li JQ [134] also proposed an artificial bee colony algorithm for FJSSP with maintenance activities constraint. An efficient initialization scheme was introduced to construct the initial population with a certain level of quality and diversity. A self-adaptive strategy was adopted to enable the DABC algorithm with learning ability for producing neighboring solutions in different promising regions whereas an external Pareto archive set was designed to record the non-dominated solutions found so far.

Two effective ABC algorithms were presented by Wang L [135,136] to solve the mono-objective and the multi-objective FJSSP respectively. The first ABC algorithm was for makespan objective while the second one was for Pareto-based multi-objective. Multiple initializing strategies and critical path based local search are employed to improve algorithms' performance. In addition, Wang L [137] also proposed an ABC algorithm for FJSSP with fuzzy processing time. In the algorithm, initializing strategies, left-shift decoding scheme and local search based on variable neighbor search were employed to enhance performance. Taguchi method was used to investigate the influence of several key parameters in the three algorithms proposed.

2.4 Shortcomings and research trends

Most existing research focuses on developing meta-heuristics for general FJSSP and few papers address on the practical application in variety industry environments. There are some problems and new trends for FJSSP.

1. The practical constraints in variety environments are serious obstacles to apply the meta-heuristics to FJSSP in the real shop floor. How to solve FJSSP with constraints in variety industry environments is a new trend of FJSSP research.
2. Most existing scheduling methods and algorithms cannot satisfy the dynamic changes of resources, environments and other factors in the real shop floor. Dynamic rescheduling is needed to satisfy the changes in practical production environments.
3. Little existing literature applies meta-heuristics to remanufacturing related scheduling problems and deal with the scheduling related constraints in remanufacturing.

Based on the review in Sections 2.3 and 2.4, harmony search algorithm and artificial bee colony algorithm have been successfully applied to a variety of engineering optimization problems. The effectiveness and efficiency of these two algorithm are also been proved in many papers. In this thesis, we employ discrete harmony search algorithm and artificial bee colony algorithm to solve scheduling problem in the remanufacturing industry. Two scheduling related constraints in remanufacturing

environment are solved to satisfy the practical requirements in the real shop floor. The most probable processing time and fuzzy number are employed to represent the uncertain processing time of operations in remanufacturing industry. Dynamic rescheduling is developed in this thesis for new job insertion constraint in remanufacturing.

Chapter 3

Modelling FJSSP with Multiple Constraints

In this chapter, the FJSSP model is described in detail and two characteristics in the remanufacturing environment, new job insertion and uncertain or fuzzy processing time, are modeled as constraints of FJSSP.

3.1 Flexible job shop scheduling problem

In a flexible job shop, each job consists of a sequence of operations. An operation can be executed on only one machine out of a set of candidate machines. Each operation of a job must be processed only on one machine at a time, while each machine can process only one operation at a time.

The following notations and assumptions are used for the formulation of multi-objective FJSSP.

1. Let $J = \{J_i\}, 1 \le i \le n$, index i, be a set of n jobs to be scheduled. q_i denotes the total number of operations of a job J_i, d_i is the due date of J_i.
2. Let $M = \{M_k\}$, $1 \le k \le m$, index k, be a set of m machines.
3. Each job J_i consists of a predetermined sequence of operations. *Let* $O_{i,h}$ be operation h of J_i.
4. Each operation $O_{i,h}$ can be processed without interruption on one of the set of candidate machines $M(O_{i,h})$. Let $P_{i,h,k}$ be the processing time of $O_{i,h}$ on machine M_k.

5. Decision variables

$$x_{i,h,k} = \begin{cases} 1, \text{if machine } k \text{ is selected for the operation } O_{i,h} \\ 0, \text{otherwise} \end{cases} \quad (3\text{-}1)$$

$c_{i,h}$ denotes the completion time of the operation $O_{i,h}$

c_i denotes the completion time of the job J_i

6. The objectives considered in this thesis are as follows:

Makespan, denoted by C_M, is the maximal of completion time of machines.

$$\text{Min } C_M = \max_{1 \le i \le n} c_i \quad (3\text{-}2)$$

Mean of earliness and tardiness, denoted by E/T, is the average of all jobs' earliness or tardiness compared to their due dates.

$$\text{Min } E/T = \frac{\sum_{i=1}^{n} |c_i - d_i|}{n} \quad (3\text{-}3)$$

where C_i is the completion time of job J_i and d_i is the due date of job J_i.

Maximum machine workload, denoted by W_M, can be calculated by:

$$\text{Min } W_M = \max_{1 \le j \le m} w_j \quad (3\text{-}4)$$

Where w_j is the workload of machine M_j

The FJSP is a NP-complete problem when the mahine number is laerger than two [13]. The time completity of calculation for objective is about $\boldsymbol{O}(n^3 m^2 (\log O)^2)$, where n is the job number, m is the machine number and O is the total operation number of all jobs.

3.2 New job insertion constraint

In remanufacturing environment, the new job arrival is unpredictable. When a new job arrives and is inserted into the processing job sequence, conflicts may arise and the objective may be affected. Rescheduling is needed in this condition. Because the scheduling solution is executing in workshop, the machines are available for rescheduling when the current operations on them are completed. It means that machines and jobs may have different available times when rescheduling is performed. We model the new job arrival and insertion feature of remanufacturing as a constraint of FJSSP, New job Insertion.

To explain FJSSP with a new job insertion, an example is shown in the following content. Fig. 3 - 1(a) shows a Gantt chart for 3-jobs and 3-machines FJSSP. The numbers of operations in three jobs are J_1, 3, J_2, 2 and J_3, 2, respectively. The makespan value in this scheduling solution is 10. The completion time of three machines is M_1, 8, M_2, 7 and M_3, 10. Fig. 3 - 1(b) shows the new job, J_4, arrives and is

inserted into the current running schedule at Time 3. The job J_4 has three operations. Fig. 3 - 1(c) shows the result with no rescheduling. The existing schedule remains the same and the Job4 is scheduled after all the assigned operation on the machines. The earliest start time of the three machines for rescheduling is M_1, 8, M_2, 7 and M_3, 10. Fig. 3 - 1(d) shows the revised schedule after rescheduling. Both new J_4 and all yet-to-start operations of existing three jobs are rescheduled. The earliest available time of M_1 and M_3 is 3 while the available time of jobs J_2, J_3 and J_4 is also 3. M_2 is processing the first operation of Job1 when the Job4 inserts at Time 3. M_2 and J_1 are available when the first operation of J_1 is completed. Therefore, the available time of M_2 and J_1 is Time 4. In Fig. 3 - 1(d), the start time of $O_{1,2}$ and $O_{1,3}$ are delayed as compared to that of schedule in Fig. 3 - 1(b).

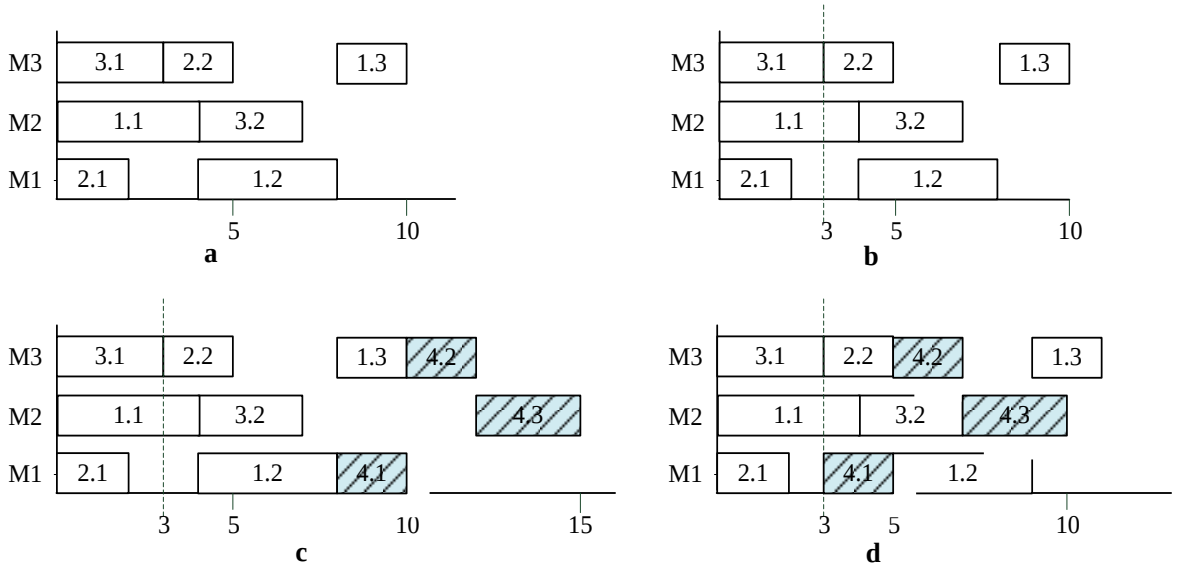

Fig. 3 - 1 An example of new job insertion

3.3 Uncertain processing time constraint

In remanufacturing environment, the operation processing time is not fixed and may vary in real shop floor. This section proposed two strategies to model the uncertainty of processing time. One model is based on the most probable processing time and another model is based on fuzzy number.

3.3.1 Modeling based on most probable processing time

In this model, the scheduling is according to operation's most probable processing time that is evaluated by engineers' experiences. Rescheduling is executed when the operation processing time increases larger than the most probable processing time. To explain the model detail, an example is shown in following contents.

Table 3 - 1 shows the operation most probable processing time of job re-processing in remanufacturing. There are three Volutes and two Impellers jobs. The first operation of Volute 2 can only be processed by milling machine 1. The processing time is 2. For the two Impeller jobs, there are different selectable machines and the processing times on different machines are different.

Table 3 - 1 An example of the model based on the most probable processing time

Sub-assembly	O 1		O 2	O 3	
	Milling 1	Milling 2	Metal building	Turning 1	turning 2
Volute 1	5	4	6	5	4
2	2		8	4	3
3	5	4	8	11	7
Impeller 1	5	4	6	4	2
2	3	6		3	4

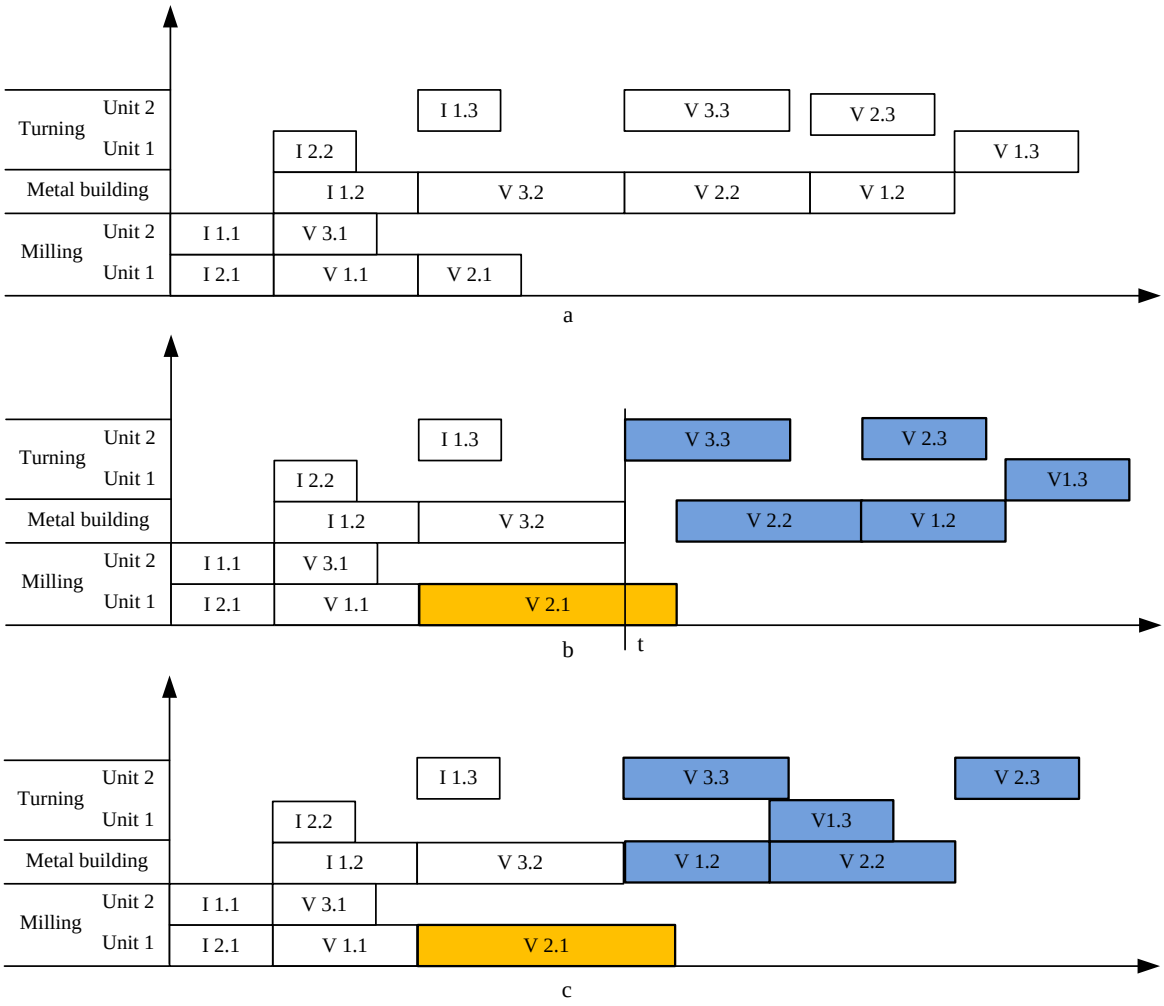

Fig. 3 - 2 An example of increasing processing time

Fig. 3 - 2 (a) shows one scheduling solution of the data in Table 3 - 1. If the processing time increases and affects the objective, rescheduling is executed on the operations that are not start yet in order to improve the results. For example, the processing time of operation V2.1 on machine milling 2 increases from the value shown in Fig. 3 - 2 (a) to that shown in Fig. 3 - 2 (b). The processing time change of job operation V2.1 affects the maximum completion time after time "t". The operations after V2.1 should be rescheduled at time "t". These operations are V2.2, V2.3, V1.2, V1.3 and V3.3. Fig. 3 - 2 (c) shows the Gantt chart after rescheduling. The operations V1.2 and V2.2 exchange their positions. The start time of operations V1.2 and V1.3 becomes earlier than that in Fig. 3 - 2 (b). It is clear that the maximum complete time in Fig. 3 - 2 (c) is less than that in Fig. 3 - 2 (b).

3.3.2 Modeling based on triangular fuzzy number

In this model, we model the uncertain processing time as fuzzy processing time. The uncertainty processing time is described in three values, the smallest processing time, the most probable processing time and the largest processing time. Triangular fuzzy number (TFN) is used to show the operation processing time in scheduling process. The TFN is shown in the following formula:

$$t_{i,j,k} = (t^1_{i,j,k}, t^2_{i,j,k}, t^3_{i,j,k}) \tag{3-6}$$

where $t^1_{i,j,k}$, $t^2_{i,j,k}$ and $t^2_{i,j,k}$ are three probable processing time of operation $O_{i,j}$ on machine M_k.

To compare and order TFN, addition operation, maximum operation and ranking operation are used. The addition operation is for computing the summary time. The maximum operation and the ranking operation are used to compare TFNs. The three operations are computed as follows:

Addition operation: two TFNs, $t = (t_1, t_2, t_3)$ and $t' = (t'_1, t'_2, t'_3)$, their addition is as follows.

$$t + t' = (t_1 + t'_1, t_2 + t'_2, t_3 + t'_3) \tag{3-7}$$

Ranking operation: three criteria are used to compare two TFNs $t = (t_1, t_2, t_3)$ and $t' = (t'_1, t'_2, t'_3)$.

1. If $(t_1 + 2t_2 + t_3)/4 > (<)(t'_1 + 2t'_2 + t'_3)/4$, then $t > (<)\ t'$.

2. If $(t_1 + 2t_2 + t_3)/4 = (t'_1 + 2t'_2 + t'_3)/4$, then t_2 and t'_2 is compared. If $t_2 > (<)\ t'_2$, then $t > (<)\ t'$.

3. If $t_2 = t'_2$, then the spreads of two TFNs are compared. If $t_3 - t_1 > (<)\ t'_3 - t'_1$, then $t > (<)\ t'$.

Maximum operation: The max of two TFNs is approximated with the criterion: If $t > t'$, then $t \vee t' = t$; otherwise $t \vee t' = t'$.

To explain the model in detail, an example is shown. Table 3 - 2 shows the fuzzy processing time of 3 jobs on 4 machines. In Table 3 - 2, each fuzzy processing time includes three values. For example, the processing time of operation $O_{2,1}$ on machine M_3 is (1, 2, 5). This means that the smallest

processing time is "1", the most probable processing time is "2" and the largest processing time is "5". To present the scheduling Gantt chart using TFN, Fig. 3 - 3 shows a Gantt chart of one solution for the data in Table 3 - 2.

Table 3 - 2 An example of triangle fuzzy number for modeling uncertain processing time

Job	Operation	Machine			
		M_1	M_2	M_3	M_4
1	O_{11}	-	(5,6,7)	-	(3,5,6))
	O_{12}	(1,2,3)	(6,8,10)	(5,9,11)	(9,10,14)
	O_{13}	(1,3,4)	-	(4,6,8)	-
2	O_{21}	-	(5,7,9)	(1,2,5)	(1,2,3)
	O_{22}	(5,7,9)	(4,5,7)	(3,5,6)	(1,2,3)
	O_{23}	(7,8,11)	-	-	(1,3,4)
	O_{24}	(7,9,10)	-	(5,6,9)	-
3	O_{31}	(2,4,8)	-	(4,5,7)	-
	O_{32}	(6,8,12)	(5,8,10)	(6,7,10)	(4,7,9)
	O_{33}	(8,9,12)	(3,4,5)	-	-

In Fig. 3 - 3, the triangle under machine line means the start time of one operation while the triangle above the machine line means the completion time of one operation. From the left side, the first vertex of one triangle means the smallest start or completion time. The second vertex means the most probable start or completion time and the last vertex presents the largest start or completion time. For example, the first triangle above the line of M_4 means the completion time of operation $O_{1,1}$ is (3, 5, 6). From the left, the first vertex value is "3", the second vertex value is "5" and the third vertex value is "6". The first triangle under the line of machine M_4 means the start time of operation $O_{2,3}$ is (7, 11, 17). For each job, if the first operation starts from time 0, the start time is presented using a rectangle under the line of processing machine.

To explain the Gantt chart in detail, job 2 is as an example. The first operation $O_{2,1}$ is processed on machine M_3. The start time of operation $O_{2,1}$ is 0. There is a rectangle under machine M_3. The completion time of operation $O_{2,1}$ is (1, 2, 5) and the first triangle above the line of machine M_3 shows the fuzzy number (1, 2, 5). From the left, the first vertex value is "1", the second vertex value is "2" and the third vertex value is "5". The operation $O_{2,2}$ is processed on machine M_1 and the start time is (2, 4, 8). The first triangle under the line of machine M_1 shows the fuzzy start time (2, 4, 8). The completion time of $O_{2,2}$ is (7, 11, 17) and the second triangle above machine M_1 line shows the fuzzy number (7, 11, 17). The operation $O_{2,3}$ is processed on machine M_4 and the start time is (7, 11, 17).

The first triangle under machine M_4 line shows the fuzzy start time. The completion time of $O_{2,3}$ is (8, 14, 21) and the second triangle above machine M_4 line shows the fuzzy completion time. The operation $O_{2,4}$ is processed on machine M_3 and the start time is (8, 14, 21). The second triangle under machine M_3 line shows the fuzzy start time. The completion time of $O_{2,4}$ is (13, 20, 30) and the third triangle above machine M_3 line shows the fuzzy completion time.

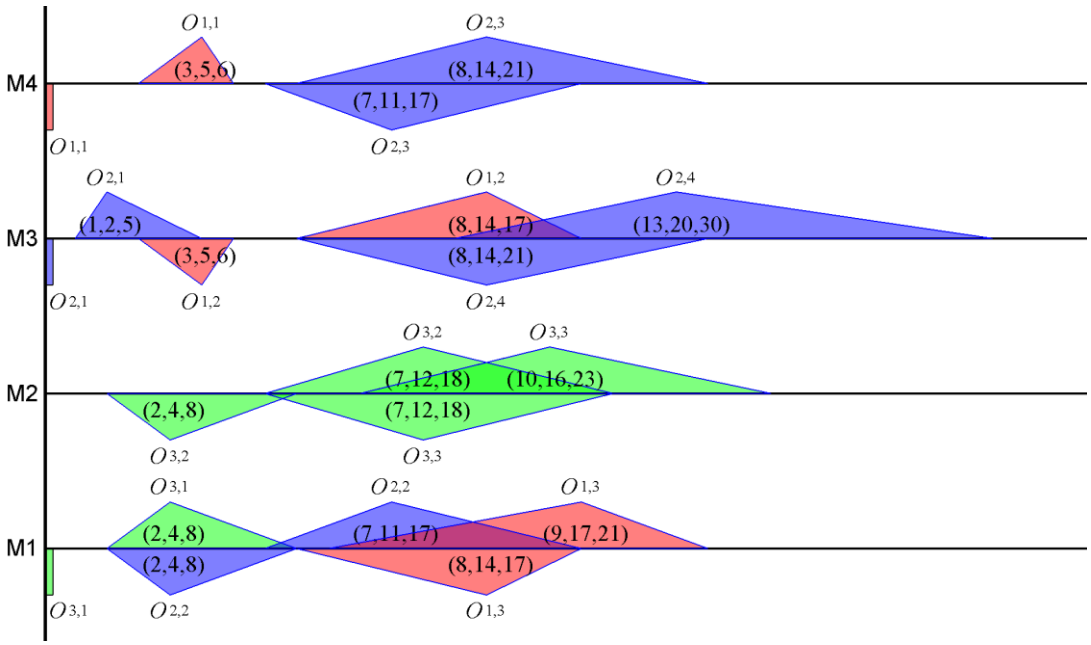

Fig. 3 - 3 Fuzzy Gantt chart

3.3.3 Comparisons of two models

Two models for uncertain processing time constraint are proposed in section 3.3.1 and section 3.3.2. The first model is based on the most probable processing time and the second model is based on fuzzy number. The first model can obtain a solution with an exact objective result while the second model obtains the result with a fuzzy number. The first model will execute rescheduling when new job comes and inserted into existing sequence or the processing time is larger than the most probable processing time and affect the objective. The second model executes rescheduling only for a new job insertion. Hence, the first model will execute more times rescheduling than the second model. The scheduling complexity of the first model is also higher than that of the second one. The second model uses a triangular fuzzy number to show the processing time and only give objective result in a range. However, the second model can reduce the rescheduling times. These two models have both advantage and disadvantage. Based on the two models, we will discuss FJSSP with new job insertion and fuzzy

processing time constraints in the following Chapters. Extensive experiments and discussions will be shown to evaluate the two models under different scheduling heuristics and algorithms.

3.4 Encoding and decoding methods for FJSSP

In harmony search algorithm, one solution is called a harmony. In artificial bee colony algorithm, one solution is called a food source. To employ these two algorithms for FJSSP, this section proposed two encoding and decoding methods to present scheduling solution.

1. MAOS encoding and decoding

In this encoding and decoding method, a solution consists of two vectors corresponding to the machine assignment and operation scheduling sub-problems. After encoding, a solution is therefore composed of two parts:

- Machine assignment vector (hereafter called MA)
- Operation sequence vector (hereafter called OS)

The MA vector is to assign machine for each operation while OS is to sequence the operations. Fig. 3 - 4 (a) illustrates a machine assignment vector while Fig. 3 - 4 (b) shows the corresponding operation sequence. In MA, each element represents the machine selected for the corresponding operation. In OS, the same elements represent the different operations of the same job. For example, the first "3" in Fig. 3 - 4 (a) means that machine 3 is selected for operation $O_{1,1}$. The second "2" in Fig. 3 - 4 (b) represents the second operation of job 2 while the third "4" is the third operation of job4.

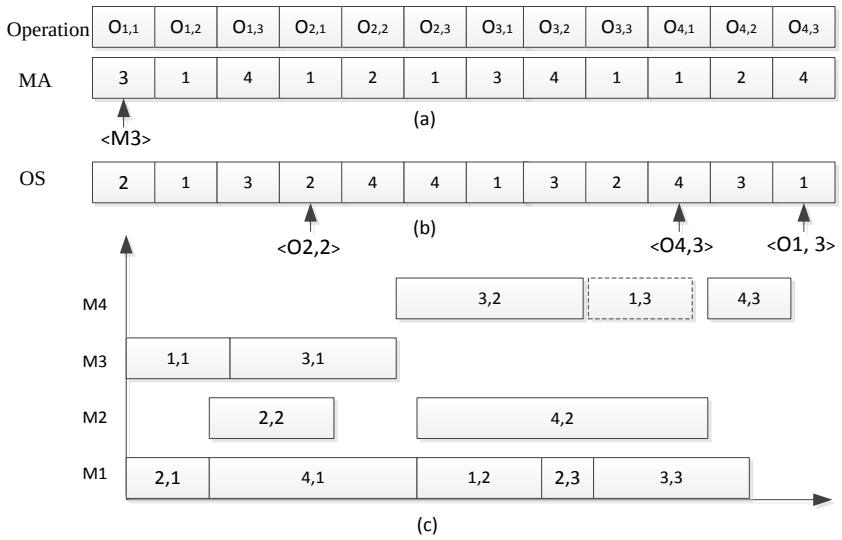

Fig. 3 - 4 Illustration of MA and OS

Pinedo [12] divided schedules into three classes: non-delay schedule, active schedule and semi-schedule. It has been verified in the above literature that an active schedule contains the optimal

schedule. In an active schedule, there is no operation to be processed earlier except putting off another operation's start time or changing the order of operations. Therefore, we decode the MAOS solution to an active schedule in order to reduce the search space. The criterion for obtaining the active schedule is to find the earliest idle time interval for each operation on the corresponding machine[53]. The process of finding the earliest idle time interval is shown in Fig. 3 - 5. For example, Fig. 3 - 4 (c) is the Gantt chart decoded from the encoding shown in Fig. 3 - 4 (a) and Fig. 3 - 4 (b). It can be seen from Fig. 3 - 4 (b) that the operation $O_{4,3}$ is decoded before operation $O_{1,3}$. In Fig. 3 - 4 (a), operation $O_{1,3}$ is processed on M_4, the same machine with operation $O_{4,3}$. If the earliest idle time is not considered, the operation $O_{1,3}$ will be assigned after operation $O_{4,3}$. It can be seen from Fig. 3 - 4 (c) that the end time of $O_{3,2}$ is larger than the end time of $O_{1,2}$ and the interval between operations $O_{3,2}$ and $O_{4,3}$ is larger than the processing time of operation $O_{1,3}$. Hence, operation $O_{1,3}$ can be inserted between operations $O_{3,2}$ and $O_{4,3}$. In this way, the release time of M_4 will be the completion time of operation $O_{4,3}$. Insertion operation $O_{1,3}$ on machine M_4 does not increase the release time of machine M_4.

Procedure: Find the earliest idle time interval for operation $O_{i,j}$

Step1: Set I=1.

Step2: Count the interval, S, between operations I and operation I+1 on current machine.

Step3: If S is larger or equal the processing time of operation O_{ij}, go to Step5, else go to Step4.

Step4: I=I+1, if I equals to the number of operations the machine has processed, go to Step6; else go to Step2.

Step5: Insert operation O_{ij} between operations I and I+1 and output.

Step6: Insert O_{ij} after the finial operation which has been scheduled and output.

Fig. 3 - 5 The process of finding the earliest idle time

2. Three-value element encoding and decoding

In three-value element encoding decoding method, the machine assignment and operation sequencing are done and describe in an element. In each element of a solution, there are three values, job number, operation number and processing machine number. For example, a solution of 4-job, 4-machine problem is shown in Fig. 3 - 6. The first element, (3, 1, 2), means that the operation $O_{3,1}$ is processed on machine M_2. The element number is the total operation number of all jobs. This encoding method consists of the order of operation sequence and machine assignment.

The decoding of this method is similar to MAOS method. Table 3 - 3 shows the fuzzy processing time of each job on corresponding machines for the solution shown in Fig. 3 - 6. The solution can be decoded

from left to right. The operations on the same machine are processed based on the order appearance in operation sequence. For example, the third element, (4, 1, 4) and the fifth element, (3, 2, 4) have the same processing machine M_4. The operation, (4, 1) will be processed first. The operation, (3, 2) can start when the operation (4, 1) is completed. In this way, the solution can be decoded to a schedule. The processing orders on all machines are $M_1 = O_{4,2}, O_{3,3}, O_{2,3}$, $M_2 = O_{3,1}, O_{2,1}, O_{1,3}$, $M_3 = O_{1,1}, O_{1,2}, O_{2,2}$ and $M_4 = O_{4,1}, O_{3,2}, O_{4,3}$. The fuzzy completion time can be computed and the fuzzy Gantt chart of this example is shown in Fig. 3 - 7.

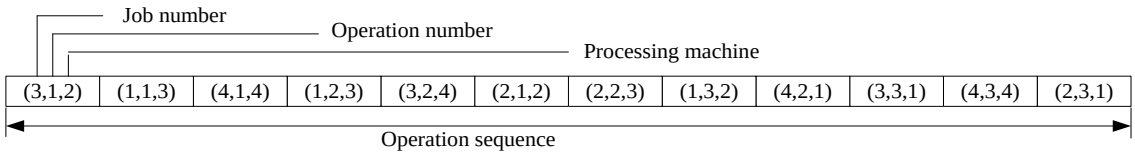

Fig. 3 - 6 A solution with three values element

Table 3 - 3 An example of fuzzy processing time for encoding and decoding

Job	Operation	Machine			
		M_1	M_2	M_3	M_4
1	O_{11}	(2,4,8)	-	(4,5,7)	-
	O_{12}	(6,8,12)	(5,8,10)	(6,7,10)	(4,7,9)
	O_{13}	(8,9,12)	(3,4,5)	-	-
2	O_{21}		(5,9,10)		(7,9,12)
	O_{22}	(2,4,5)		(4,7,8)	(1,2,4)
	O_{23}	(3,5,6)	(1,3,4)	(1,2,5)	
3	O_{31}	-	(5,6,7)	-	(3,5,6))
	O_{32}	(1,2,3)	(6,8,10)	(5,9,11)	(9,10,14)
	O_{33}	(1,3,4)	-	(4,6,8)	-
4	O_{41}	-	(5,7,9)	(1,2,5)	(1,2,3)
	O_{42}	(5,7,9)	(4,5,7)	(3,5,6)	(1,2,3)
	O_{43}	(7,8,11)	-	-	(1,3,4)

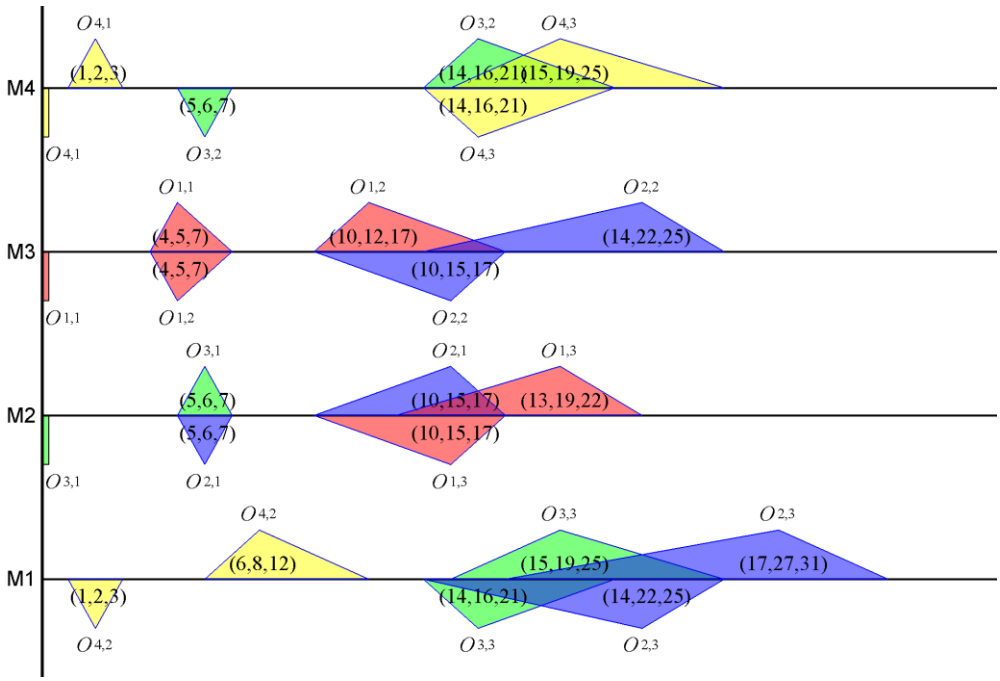

Fig. 3 - 7 Fuzzy Gantt chart of the three values element encoding and decoding

Chapter 4

DHS Algorithm for FJSSP

4.1 Introduction

This Chapter proposes a Pareto-based grouping discrete harmony search algorithm (PGDHS) to solve the single objective and multi-objective flexible job shop scheduling problem (FJSSP). Two objectives, namely the maximum completion time (makespan) and the mean of earliness and tardiness, are considered simultaneously. First, two novel heuristics and several existing heuristics are employed to initialize the harmony memory. Second, multiple new harmony improvising strategies are proposed to improve the performance of harmony search algorithm. The operation sequence in a new harmony is produced based on the encoding method and the characteristics of FJSSP. Thirdly, two local search methods based on critical path and due date are embedded to enhance the exploitation capability. Finally, extensive computational experiments are carried out using well-known benchmark instances. Three widely used performance measures, number of non-dominated solutions, diversification metric and quality metric, are employed to test the performance of PGDHS algorithm. Computational results and comparisons show the efficiency and effectiveness of the proposed PGDHS algorithm for solving the multi-objective flexible job-shop scheduling problem.

The rest of this Chapter is organized as follows: In Sections 4.2-4.6, we present the PGDHS algorithm in detail. Section 4.7 presents the experiments and results compared with other algorithms in existing

literature to demonstrate the superior performance of the proposed PGDHS algorithm. Finally, we conclude Chapter in Section 4.8.

4.2 Initialization

In DHS algorithm, the quality of initial HM affects the convergence speed to an optimal solution. Therefore, it is critical to generate a better quality initial HM. In this DHS algorithm, MAOS encoding and decoding method is employed to represent solution. The initialization process includes machine assignment phase and operation scheduling phase. The adoption of a mix of following rules is used to produce the machine assignment and operation scheduling for FJSSP. For example, the percentages of each initializing rule are equal.

1) Machine assignment component

Except Random rule, three heuristic rules are employed for initializing machine assignment:

- Global minimum-processing time rule [16]

This rule considers both the processing time and the workload of machines and start from the operation with the global minimum processing. The processing time is added to the machine workload. For other operations, the machine with minimum processing time (workload) are fixed and assigned. The machine's workload update is also performed. This rule considers the global workload among all machines and can help find better makespan. The disadvantage is the lack of diversity.

- Two-step greedy rule [29]

The operations are sorted in ascending order based on the number of selectable machines with ascending order of processing times to break ties when there is equal number of selectable machines. The machines are sorted in their workload non-decreasing order. The operation is taken from the operations list and the first machine that belongs to the machine list is assigned to the operation. The workload of this machine is updated and the machine sorting is also updated. The process iterates until all operations have been assigned to machines. The two-step greedy rule is proposed for minimization of the maximum tardiness.

- Proposed minimum-completion time rule

We develop this rule based on the operation minimum processing time rule [28]. The proposed rule is as follows:

Step1: For each operation $O_{i,j}$, select machine M1 with the minimum processing time and machine M2 with the earliest feasible time from selectable machine set.

Step2: Calculate the completion time T1 and T2 of operation $O_{i,j}$ on machines M1 and M2.

Step3: If T1<T2, M1 is selected for processing operation $O_{i,j}$; otherwise, M2 is selected for processing operation $O_{i,j}$.

This rule considers the completion of each operation and is conducive to optimize the makespan. It can be seen from the steps of the minimum-completion time rule that the computation complexity is $O(2\sum_{i=1}^{n} q_i)$

2) Operation scheduling component

Once the machine assignment is fixed, the operations on each machine should be sequenced. Except for random rule, the operation scheduling component is initialized by the following three different methods[26]:

- Most work remaining rule

This method firstly orders the jobs in a descending order based on the remaining processing time. The job with the most remaining processing time will be selected first and put into the operation sequence. The iteration is repeated until all operations of all jobs have been sequenced.

- Most operations remaining rule

This rule selects jobs based on the number of remaining operations. The job with the most remaining number of operations will be selected first.

- Shortest processing time rule

In this approach, the operation with the shortest processing time will be selected from the next executable operations.

3) Early end time rule

The initialization methods in Sections 4.2.1 and 4.2.2 do machine assignment component first and perform the operation scheduling after machine assignment. In this section, we propose an initialization rule in the reverse process. The process is as follows:

Step1: The operation sequence is obtained by randomly shuffling the order of all operations of all jobs.

Step2: For each operation $O_{i,j}$ in operation sequence, calculate the end time on all selectable machines.

Step3: The machine with the minimum end time is selected for processing operation $O_{i,j}$.

4.3 Dynamic grouping

The small-sized HM works better than a large one for the HS algorithm and dynamic subpopulation can effectively balance the fast convergence and large diversity[138]. Hence, the DHS algorithm employs multiple small-sized dynamic sub-HMs. More specifically, the whole HM of the DHS algorithm is grouped into small-sized sub-HMs randomly in first iteration. Then, each sub-HM uses its own members to search for better solutions in the search space. After this iteration, the sub-HMs form a whole HM that is regrouped again into small-sized sub-HMs randomly, and the sub-HMs restart

their search independently. This process, shown in Fig. 4 - 1, is continued until a termination criterion is met.

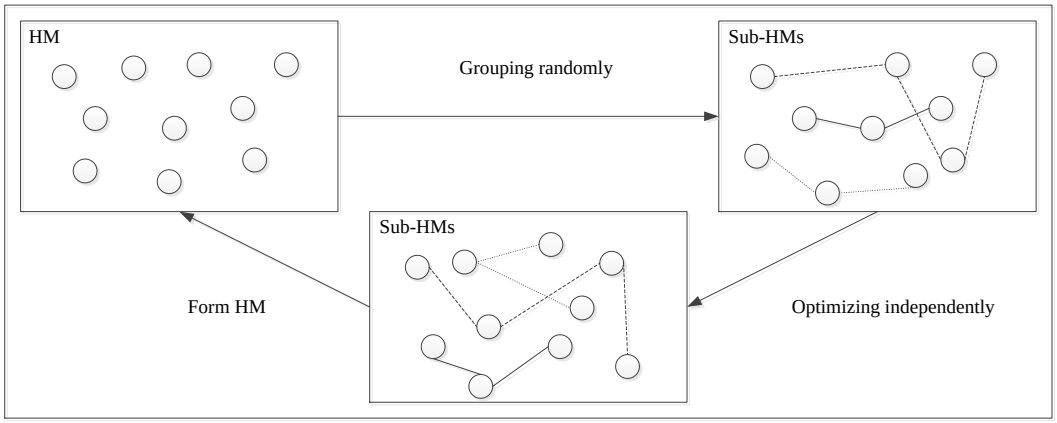

Fig. 4 - 1 Dynamic grouping optimization

4.4 Improvising new harmony

In DHS algorithm, each harmony includes machine assignment and operation sequence. In MA part, each element represents a machine selected for processing corresponding operation. From different harmonies, the elements at the same position are for the same operation in MA part while the elements at the same position may mean the operation of different jobs in OS part. Therefore, we improvise a new harmony for the machine assignment part and operation sequence part respectively. The process of improvising a new MA solution is shown in Fig. 4 - 2.

Procedure: Improvising the new MA part for new harmony
Step1: Generate two random numbers *R1* and *R2*.
Step2: If *R1*<*HMCR* and *R2*<*PAR*, go to Step3; else, go to Step6;
Step3: Select one machine *k* for current operation. If *k* is the last machine in the set of candidate machine, go to Step4; else, go to Step5.
Step4: Select the first machine *k'* in the set of candidate machines for current operation.
Step5: Select the next machine *k''* of machine *k* in the set of candidate machines for current operation.
Step6: Randomly select one machine *k'''* from the set of candidate machines for current operation.

Fig. 4 - 2 Process of improvising the MA part of a new harmony

According to the characteristic of FJSSP and the encoding strategies, we use crossover operators for the OS part of a new harmony. There are several crossover operations proposed during the past decades, such as partial-mapped crossover, order crossover, cycle crossover and so on [16]. In this study, we employ a new crossover operator based on order crossover. We obtain two new harmonies from the

current two harmonies. The employed crossover operator works for the OS part as illustrated in Fig. 4 - 3 and summarized below:

Step1: Generate a random number *R* from 1 to number of jobs;

Step2: Copy the values from the OS part of Harmony 1 to the corresponding positions in New Harmony 1 where the values are less than or equal to *R*.

Step3: Copy the values from the OS part of Harmony 2 to the corresponding positions in New Harmony 2 where the values are larger than *R*.

Step4: From the OS part of Harmony 2, copy the values which do not appear in New Harmony 1 to the vacant positions in New Harmony 1 from left to right according to the order of the sequence in Harmony 2.

Step5: From the OS part of Harmony 1, copy the values which do not appear in New Harmony 2 to the vacant positions in New Harmony 2 from left to right according to the order of the sequence in Harmony 1.

We proposed an improvisation to generate multiple new harmonies. For each sub-HM, the number of new harmonies is the same as the number of harmonies in this sub-HM. For the OS part, there will be four operation sequences after applying crossover operation in Fig. 4 - 3. Meanwhile, we also generate four MA solutions correspondingly. We select two harmonies from the four harmonies based on domination. Two harmonies will be selected randomly if the four harmonies are non-dominated.

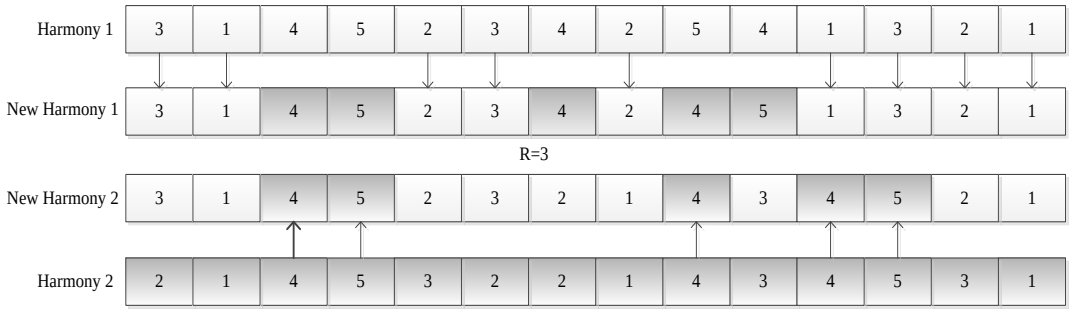

Fig. 4 - 3 Illustration of the crossover operator for OS part of a new harmony

4.5 Local search

In this section, we proposed two local search algorithms for FJSSP, one for makespan criterion based on critical path and another for E/T criterion based on due date of jobs.

1) Local search algorithm for makespan criterion

The critical path concept has been employed in job shop scheduling problem for improving the convergence speed[139]. In this study, we proposed a local search algorithm based on critical path for makespan criterion of FJSSP. However, we also consider the neighborhood from the different

selectable machines of public critical operations. In the local search method, if the operation has two or more machines available for selection, we will assign a machine that is different from the current one. The first operation and the last operation in the public critical block will swap with the adjacent public critical operations.

2) Local search algorithm for E/T criterion

For earliness-tardiness criterion, the jobs are divided into two groups in the proposed local search algorithm. Group one includes the jobs which are completed before the due date while group two has the jobs which are completed after the due date. For all operations on all machines, the operations which belong to the job in group one move to right adjacent position while the operations which belong to the job in group two move to left adjacent position. More specifically, the operations on the same machine are considered as pair from the first two operations. If the jobs including one pair of operations are included in different groups, the position of the pair of operations on machine will be interchanged.

4.6 Algorithm framework

Due to the complexity of flexible job shop scheduling problem, the proposed DHS algorithm employs multiple initialization methods, novel improvisation of new harmonies, two local search algorithms for solving the FJSSP effectively. The diversity of populations and the balance of global exploration and local exploitation are all considered. The proposed DHS algorithm is illustrated in Fig. 4 - 4. First, harmony memory and archived set are initialized. Then, harmony memory is divided into multiple Sub-HMs. Each Sub-HM executes improvising new harmonies and local search independently. Then, Sub-HMs and AS are updated. If the maximum iteration is not reached, the harmony memory is grouped again. The grouping operator and local search balance the global exploration and local exploitation. The grouping operator and improvising a new harmony stress the diversity of population during the search process. Hence, the DHS is expected to yield good performance. In each generation, the computational complexity of improvising new harmony includes two parts, Machine assignment and operations sequencing. For machine assignment, the complexity is $O(\sum_{i=1}^{n} q_i)$. The complexity for operation sequencing is $O(log_2 \sum_{i=1}^{n} q_i * log_2 \sum_{i=1}^{n} q_i)$. The complexity to update one harmony is $O(1)$. Hence, the computational complexity to generate new harmonies and update existing harmonies is $O((\sum_{i=1}^{n} q_i + log_2 \sum_{i=1}^{n} q_i * log_2 \sum_{i=1}^{n} q_i + 1) * HMS)$.

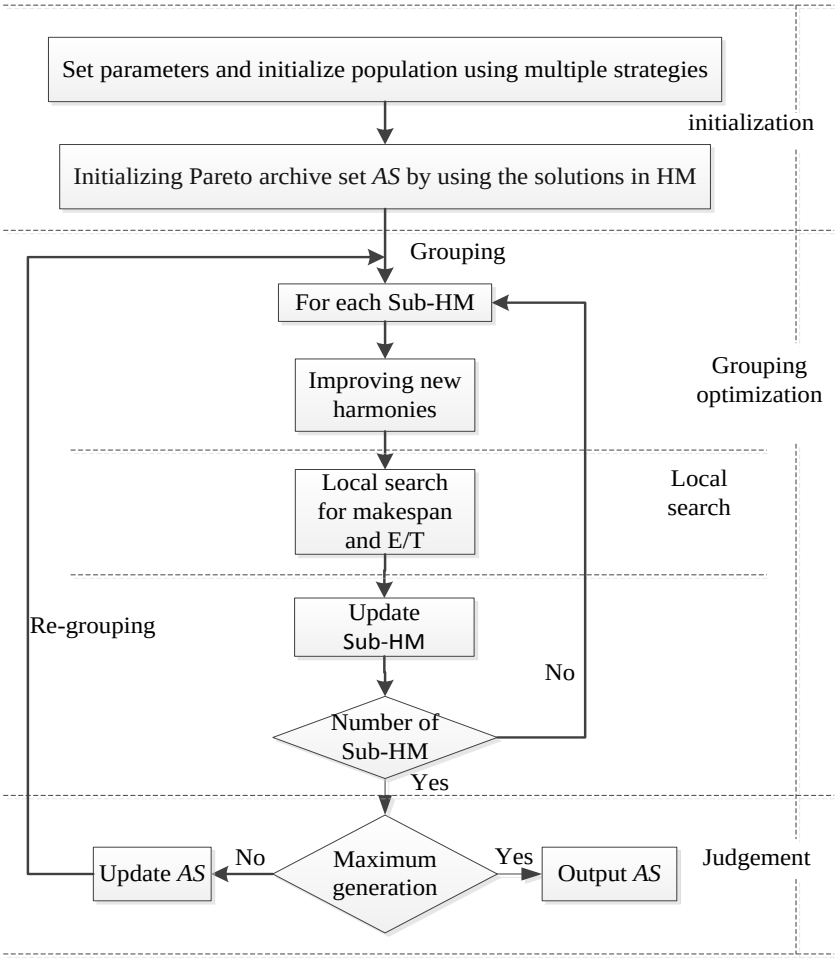

Fig. 4 - 4 The proposed DHS algorithm

4.7 Results and comparisons

4.7.1 Experiment setup

To test the performance of the proposed DHS algorithm, extensive experimental evaluation and comparisons with other methods are presented using well-known FJSSP benchmark sets. Two sets of problem instances are considered in this paper:

- The first data set are five Kacem instances[140].
- The second data set, called BRdata, is a set of 10 problems by Brandimare[141].

The Kacem benchmark set is composed of 5 instances with the size ranging from 4 jobs, 5 machines and 11 operations to 15 jobs, 10 machines and 60 operations. The Brandimare benchmark set includes 10 problems with the size ranging from 10 jobs, 6 machines and 55 operations to 20 jobs, 16 machines and 240 operations. In order to introduce due date in the data sets, the method inspired by Gholami[49] is employed. In this study, the due date of job J_i can be defined by

$$D_i = \left(1 + \frac{T \times n}{m}\right) \times \sum_{j=1}^{n_i} p_{i,j} \tag{4-1}$$

where T is a parameter that has been fixed as $T = 0.3$, n is the number of jobs, m is the number of machines, $p_{i,j}$ is the average processing time of operation $O_{i,j}$ on all selected machines.

Algorithms were coded in C++.net and run on Intel 2.8 GHz PC with 1 GB memory. The parameters HMS, HMCR and PAR affect the performance of HS algorithm. Omran and Mahdavi [114] suggested that it was generally better to use a large value of HMCR (i.e., >=0.9). A large PAR value enhances the local exploitation ability of the algorithm while a small PAR value enlarges the search area and diversity of the HS algorithm. Based on our previous research experiences, the parameters are fixed as follows: HMS=1000, HMCR=0.95 and PAR=0.5. Each instance is run for the 20 independent replications with 10×*n*×*m* generation. In this paper, grouping operator is employed for improving algorithm convergence.

4.7.2 Single objective-Makespan

To test the performance of the proposed DHS algorithm for a single objective, we considered the makespan of the 10 instances from BRdata, which range from 10 jobs, 6 machines to 20 jobs, 15 machines. We compared the DHS with four recent algorithms, BR [141], LEGA [16], Xing [142]and HTSA[71]. The experimental results and comparisons are shown in Table 4 - 1. It can be seen from Table 4 - 1 that: 1) Our DHS algorithm achieves the optimal solutions for 7 out of 10 instances. 2) For instances MK04, MK09 and MK10, DHS algorithm outperforms the four compared algorithms. 3) For instances MK01 and MK05, DHS and HTSA obtain the same results and the results are optimal. 4) Xing, HTSA and DHS produce the optimal solution for instance MK03 while all compared algorithms can find the optimal solution for instance MK08. 5) For instances MK02 and MK06, the DHS obtains worse results than HTSA. It should be noted that the objectives of BR, LEGA, Xing and HTSA are makespan and workload while the DHS algorithm in this study considers makespan, earliness and tardiness. We also show the low bound solutions of the 10 instances. DHS algorithm can find the low bound results for MK03 and MK08. For MK01 and MK02, DHS algorithm can find results that are very near to the low bound results. For other instances, there are some distance between the DHS results and the low bound results. However, DHS algorithm has obvious advantage in computation time. For 10 instances, the computation times of DHS are from several seconds to tens of seconds that are much less than those for getting the low bound. We also count the relative error (DHS_RE) between the results by DHS and the low bound. The values of minimum, average and the maximum relative errors are 0.00, 0.11 and 0.28, respectively. Thus, the comparisons of makespan results verify

the efficiency of the DHS algorithm. Fig. 4 - 5 shows the Gantt chart of the best solution by DHS algorithm for instance MK01, and the value of makespan is 40 in this solution.

Table 4 - 1 The makespan results of BRdata by five algorithms

Instance	Size	LB	*CM*					
			BR	LEGA	Xing	HTSA	DHS	DHS_RE
MK01	10,6	36	42	40 [a]	42	40 [a]	40 [a]	0.11
MK02	16,6	24	32	29	28	26 [a]	28	0.17
MK03	15,8	204	211	N/A	204 [a]	204 [a]	204 [a]	0.00
MK04	15,8	48	81	67	68	61	60 [a]	0.25
MK05	15,4	168	186	176	177	172 [a]	172 [a]	0.02
MK06	10,15	33	86	67	75	65 [a]	67	0.16
MK07	20,5	133	157	147	150	140 [a]	143	0.08
MK08	20,10	523	523 [a]	523 [a]	523 [a]	523 [a]	523 [a]	0.00
MK09	20,10	299	369	320	311	310	309 [a]	0.03
MK10	20,15	165	296	229	227	214	212 [a]	0.28

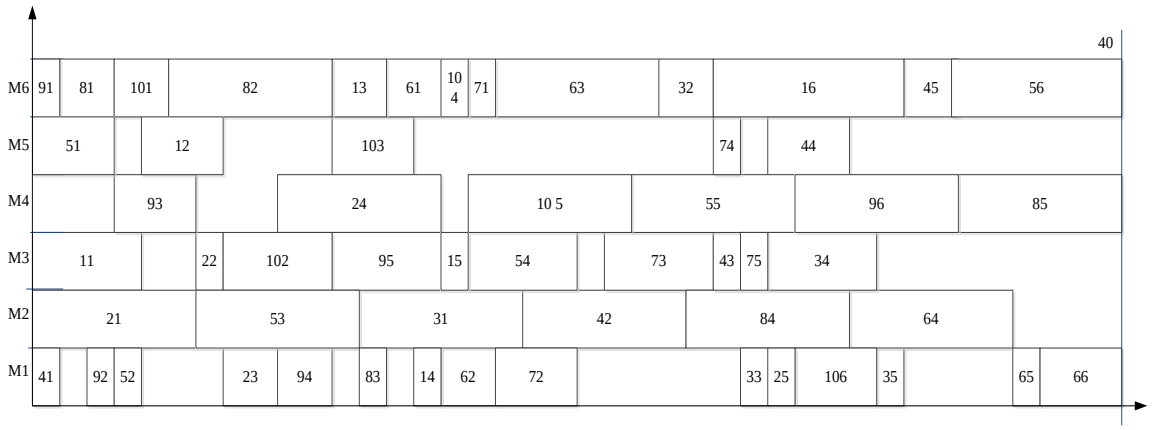

Fig. 4 - 5 Gantt chart of instance MK01 with makespan 40

4.7.3 Pareto-based multi-objective objectives

A multi-objective optimization problem can be described as follows:

$$\text{Min } f(x) = (f_1(x), f_2(x), \cdots, f_n(x)), \quad x \in \Omega, \quad f(x) \in R^n \tag{4-2}$$

where x is the decision vector in space Ω and $f(x)$ is the objective vector.

Pareto domination states that solution A dominates solution B if and only if $\forall i \in \{1,2,\cdots,n\}, f_i(A) \le f_i(B)$ and $\exists i \in \{1,2,\cdots,n\}, f_i(A) < f_i(B)$. Solution A is an optimal in the Pareto sense if there is not any solution B which dominates A. Pareto optimal set is the collection of all Pareto optimal solutions and the corresponding image in the objective space is the Pareto front. In this paper, an archive set (AS) is used to record the non-dominated solutions during the iterations. During the

search process, if a new solution dominates one or more solutions in AS, the new solution will replace the dominated solutions. For multi-objective optimization, the algorithm should obtain more non-dominated solutions with good proximity and diversity [143] with respect to the true Pareto front. In other words, it requires the algorithm to obtain more non-dominated solutions, on or closer to the optimal Pareto front, and distributed evenly over the whole Pareto front.[76]

Pareto-based multi-objective optimization algorithms are committed to find an approximate of non-dominated solutions. Hence, the performance metrics are different from the single objective methods. The following widely used performance measures are used in the paper[144] with larger values representing better performances:

- Number of non-dominated solutions (NNdS)

  This performance measure counts the total number of non-dominated solutions generated by the compared algorithms.

- Diversification metric (DM)

  DM determines the diversity of non-dominated solutions by each compared algorithm. Inverted generational distance (IGD) is used by many researchers for assessing the DM performance[145]. Let P^* be a set of uniformly distributed points in the objective space along the Pareto front (PF). Let P be the set of non-dominated solutions by compared algorithms. The inverted generational distance from P^* to P is defined as

$$IGD(P*, P) = \frac{\sum_{v \in P^*} d(v, P)}{|P^*|} \tag{4-3}$$

  where $d(v,P)$ is the minimum Euclidean distance in the objective v and the points in P. If $|P^*|$ is large enough to represent the PF well, $IGD(P*,P)$ could measure both the diversity and convergence of P in a sense. To have a low value of $IGD(P*,P)$, P must be very close to the PF and should not miss any part of the whole PF. In this paper, we set P^* as the set of Pareto solutions in PF and P is the set of non-dominated solutions obtained by each algorithm.

- Quality metric (QM)

  In this metric, a combined overall Pareto front is constructed from all non-dominated solutions obtained by all compared algorithms. Then, the number of Pareto solutions contributed by each comparing algorithm to the overall Pareto front is counted and the percentage of the Pareto solutions belonging to each algorithm is also calculated. The algorithm with higher value is better.

To test performance, the proposed PGDHS algorithm is compared with several recently published algorithms. These algorithms are AL+CGA presented by Kacem et al. [140], Multiagent method (Multi-Agent) proposed by Wu and Weng [98], the multi-objective genetic algorithm (MOGA) designed by Wang et al.[146], the particle swarm optimization and local search algorithm (MoPSO+LS) developed by Moslehi and Mahnam [64], the hybrid shuffled frog-leaping algorithm (HFSLA) proposed by Li et al. [104] and the enhanced Pareto-based artificial bee colony algorithm (EPABC) designed by Wang [136]. The two objectives are considered simultaneously, i.e. minimization of makespan (denoted by C_M) and the mean of earliness and tardiness (denoted by E/T).

Table 4 - 2 shows the results of five Kacem instances. All Pareto solutions in Table 4 - 2 are shown in boldface. It can be easily seen that: 1) Except AL+CGA, all algorithms can obtain the three Pareto solutions, (11, 7), (12, 6) and (16, 2), for instance 4×5. 2) For instance 8×8, MOPSO+LS, HSFLA and EPABC identify two Pareto solutions, (14, 10) and (15, 8) while MOGA obtains three, (14, 10), (15, 8) and (20. 5). However, PGDHS obtains one more Pareto solution, (24, 3). 3) For instance 10×7, PGDHS obtains three Pareto solutions, (11, 13), (12, 9) and (16, 4). MOGA, MOPSO+LS, HSFLA and EPABC obtain just two. 4) AL+CGA algorithm obtains three of seven Pareto solutions for instance 10×10 while MOGA, MOPSO+LS, HSFLA and EPABC obtain five. PGDHS discovers all seven Pareto solutions. 5) For instance 15×10, PGDHS obtains four Pareto solutions, (11, 18), (12, 15), (18, 8), (20, 6), more than other compared algorithms. Table 4 - 2 also shows the central processing unit (CPU) time of PGDHS algorithm. It also shows that proposed algorithm is efficient since the CPU time cost is low.

Table 4 - 2 Pareto-based multi-objective results of five Kacem instances

Instance	AL+CGA		MOGA		MOPSO+LS		HSFLA		EPABC		PGDHS		CPU
(n×m)	C_M	E/T	C_M	E/T	C_M	E/T	C_M	E/T	C_M	E/T	C_M	E/T	(s)
4×5	16	6	11	7	11	7	11	7	11	7	11	7	0.32
	18	4	12	6	12	6	12	6	12	6	12	6	
			16	2	16	2	16	2	16	2	16	2	
8×8	15	11	14	10	14	10	14	10	14	10	14	10	2.12
	24	8	15	8	15	8	15	8	15	8	15	8	
			20	5							20	5	
											24	3	
10×7	12	13	11	13	11	13	11	13	11	13	11	13	2.50
	13	9	12	9	12	9	12	9	12	9	12	9	
			17	6	17	6	17	6	17	6	16	4	
10×10	7	13	7	12	7	12	7	12	7	12	7	12	3.49
	8	11	8	11	8	11	8	11	8	11	8	11	

	10	7	10	7	10	7	10	7	10	7	10	7		
	11	6	11	6	11	6	11	6	11	6	11	6		
			12	5	12	5	12	5	12	5	12	5		
											15	4		
											16	3		
15×10	23	15	11	18	11	18	11	18	11	18	11	18	9.06	
	25	10	12	15	12	16	12	15	12	15	12	15		
			18	8	18	11	18	10	18	10	18	8		
							20	9	20	9	20	6		

Table 4 - 3 shows the number of non-dominated solutions and quality metric of all compared algorithms for five Kacem instances. The second column shows the number of Pareto solutions in PF. For each compared algorithm, the total number of non-dominated solutions, the number of non-dominated solutions which belong to the overall combined PF and the average quality metric values for five instances are also calculated. For example, AL+CGA can find two non-dominated solutions for instance 5, but both of them are not in the overall combined PF. For the same instance, MOGA obtained three non-dominated solutions and all of them are in PF while PGDHS algorithm can find four non-dominated solutions and all of them are in the overall combined PF. It can be seen that the proposed PGDHS algorithm has 21 non-dominated solutions and all these non-dominated solutions are in the overall combined PF. The quality metric values obtained by PGDHS are all 100% for five instances. It means that PGDHS can obtain all Pareto solutions for five instances. For above mentioned performance measures, PGDHS has better results than the compared five algorithms.

In addition, the diversification metric values of each compared algorithms are computed for five instances. The minimum and standard deviation of IGD-metric values are shown in Table 4 - 4. It can be seen from Table 4 - 4 that five algorithms can obtain the best minimum values for five instances except AL+CGA. Among six compared algorithms, only PGDHS found the best mean and standard deviation values for five instances. Hence, the diversification metric of PGDHS algorithm is better than the five compared algorithms.

Hence, the PGDHS is better than compared five algorithms for five Kacem instances. As an example, Fig. 4 - 6 shows the Gantt chart of Pareto solution (24, 3) of Kacem instance 8×8 in which the due dates of all jobs are 22, 31, 22, 23, 33, 28, 26 and 31.

Table 4 - 3 Number of Non-dominate solutions (NNdS) and quality metric (QM) for 5 Kacem instances

Instance		AL+CGA		MOGA		MOPSO+LS		HSFLA		EPABC		PGDHS	
(n×m)	PF	NS/PF	QM%	NS/PF	QM%	NS/PF	QM%	NS/PF	QM%	NS/PF	QM%	NS/PF	QM%

4×5	3	2/0	0.0	3/3	100.0	3/3	100.0	3/3	100.0	3/3	100.0	3/3	100
8×8	4	2/0	0.0	3/3	75.0	2/2	50.0	2/2	50.0	2/2	50.0	4/4	100
10×7	3	2/0	0.0	3/2	66.7	3/2	66.7	3/2	66.7	3/2	66.7	3/3	100
10×10	7	4/3	42.9	5/5	71.4	5/5	71.4	5/5	71.4	5/5	71.4	7/7	100
15×10	4	2/0	0.0	3/3	75.0	3/1	25.0	4/2	50.0	4/2	50.0	4/4	100
Sum	21	12	8.6	17	77.6	16	62.6	17	67.6	17	67.6	21	100
Ave		3		16		13		14		14		21	

Table 4 - 4 Diversification metric (DM) for 5 Kacem instances

Instance	AL+CGA		MOGA		MOPSO+LS		HSFLA		EPABC		PGDHS		
(n×m)	min	sd	min	sd	min	sd	min	sd	min	sd	min	mean	sd
4×5	0.94	0.38	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8×8	0.35	0.43	0.00	0.56	0.00	1.25	0.00	1.25	0.00	1.25	0.00	0.00	0.00
10×7	0.33	0.93	0.00	0.43	0.00	0.43	0.00	0.43	0.00	0.43	0.00	0.00	0.00
10×10	0.00	0.34	0.00	0.27	0.00	0.27	0.00	0.27	0.00	0.27	0.00	0.00	0.00
15×10	1.60	0.72	0.00	0.35	0.00	0.59	0.00	0.38	0.00	0.38	0.00	0.00	0.00

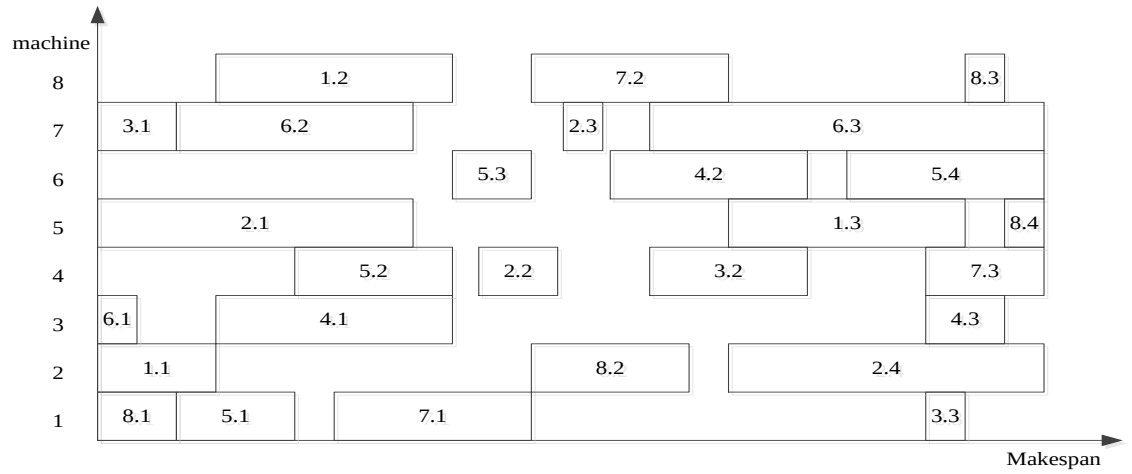

Fig. 4 - 6 Gantt chart of the solution (24, 3) for Kacem instance 8×8

PGDHS algorithm is compared with four very recent algorithms, i.e. Multi-Agent [98], MOGA [144], HSFLA [104] and EPABC [136] using BRdata set. The results obtained by compared algorithms are given in Table 4 - 5. All the Pareto solutions in Table 4 - 5 are also in boldface.

As shown in Table 4 - 5, the proposed PGDHS algorithm is better than or competitive to four compared algorithms. 1) For instance MK01, all five algorithms can locate Pareto solution (42, 3). Except Multi-Agent algorithm, other four algorithms can also obtain Pareto solution (40, 4). For instance MK 03, all compared algorithms obtain Pareto solution (204, 21). But, only PGDHS algorithm obtains Pareto solutions (220, 20) and (225, 18). 2) Multi-Agent can locate only one non-

dominated solution (42, 3) for instance MK 01 and (204, 21) for instance MK03. 3) MOGA, HSFLA and EPABC obtain two Pareto solutions, (26, 5) and (27, 3) for instance MK 02 and (173, 59) and (174, 58) for instance MK 05. However, PGDHS obtains one more Pareto solution (31, 2) for instance MK 02 and obtains four more Pareto solutions (178, 41), (180, 38), (181, 37) and (190, 36) for instance MK 05. 4) MOGA discovers a Pareto solution (60, 19) for instance MK 04 and a Pareto solution (139, 26) for instance MK 07. These two Pareto solutions are not found by PGDHS and other compared algorithms. However, PGDHS obtains four Pareto solutions for instance MK 04 and three of them are not found by MOGA. At the same time, PGDHS discovers five Pareto solutions for instance MK 07 and two of them are not found by MOGA. 5) PGDHS obtains one Pareto solution (62, 2) for instance MK 06. Meanwhile, PGDHS identifies two Pareto solutions (523, 171) and (525, 165) for instance MK 08. But, other compared algorithms could not find any non-dominated solution for these two instances. 6) For instance MK 09 and MK 10, PGDHS obtains five and four Pareto solutions respectively while other compared algorithms could not find any Pareto solutions for these two instances. Table 4 - 5 also shows the CPU times for each instance.

Table 4 - 6 shows the number of non-dominated solutions and quality metric values of all compared algorithm for 10 BRdata instances. For each compared algorithm, the number of Pareto solutions, the total number of non-dominated solutions, the number of non-dominated solutions which belong to the overall combined PF and the average quality metric values are also calculated. It can be seen from Table 4 - 6 that the proposed PGDHS algorithm has 35 non-dominated solutions. This total count is more than the number of solutions obtained by other compared algorithms (Multi-Agent 2, MOGA 13, HSFLA 10 and EPABC 10). The PGDHS algorithm also obtained maximum Pareto solutions compared to other algorithms. For 10 instances, PGDHS algorithm found 35 out of 38 Pareto solutions in the overall combined PF. The quality metric values obtained by PGDHS are 100 for seven out of ten instances. The corresponding values for MK04, MK05 and MK07 are 80.00, 85.71 and 83.33, respectively. The average quality metric value of PGDHS is the best one among the compared algorithms.

Table 4 - 7 shows the diversification metric values of each compared algorithms for 10 BRdata instances. The mean and standard deviation of IGD diversification metric values of each algorithm are also calculated. For 10 instances, the minimum values of PGDHS are all zero. At the same time, the mean and standard deviation values of PGDHS are also zero for 7 instances except MK04, MK05 and MK07 instances. For instances MK05 and MK07, the mean and standard deviation values are (0.06, 0.15) and (0.09, 0.22), respectively. These results are better than the four compared algorithms even though the values are larger than zero. For instance MK04, PGDHS can also obtain the best mean value, 0.18, while the standard deviation value, 0.40, is worse than four compared algorithms. MOGA,

HSFLA and EPABC can also obtain the best mean and standard deviation values for instance MK01. However, PGDHS algorithm has the best competitive values for mean and standard deviation of 10 instances because most results generated by PGDHS are better than those by the four compared algorithms are.

Once again, it verifies that the PGDHS possesses superior performance than the four compared algorithms in terms of several widely used performance measures. As an example, Fig. 4 - 7 shows the Gantt chart of Pareto solution (40, 4) of MK 01 instance 10×6 in which the due dates of job1-10 are 30, 28, 31, 25, 42, 33, 19, 36, 33 and 30.

Table 4 - 5 Pareto-based multi-objective results for BRdata

Instance	Multi-Agent		MOGA		HSFLA		EPABC		PGDHS		CPU
	C_M	E/T	C_M	E/T	C_M	E/T	C_M	E/T	C_M	E/T	(s)
MK01			40	4	40	4	40	4	40	4	5.2
	42	3	42	3	42	3	42	3	42	3	
MK02	28	3	26	5	26	5	26	5	26	5	5.3
			27	3	27	3	27	3	27	3	
									31	2	
MK03	204	21	204	21	204	21	204	21	204	21	24.1
									220	20	
									225	18	
MK04	67	17	60	19	62	15	62	15	62	15	14.7
	70	16	62	15	68	12	68	12	65	12	
			68	12	71	11	73	11	68	10	
									69	8	
MK05	174	61	173	59	173	59	173	59	173	59	9.0
	178	58	174	58	174	58	174	58	174	58	
			176	56	176	56	176	56	178	41	
			184	49	182	52	179	48	180	38	
									181	37	
									190	36	
MK06	67	7	62	9	64	5	64	5	62	4	25.7
	77	6	64	5	66	4	66	4			
			66	4							
MK07	144	23	139	26	141	21	141	21	140	23	13.9
	146	20	140	23	144	20	144	20	141	21	
			141	21					144	20	
			144	20					150	16	
									152	14	
MK08	523	177	523	177	523	177	523	177	523	171	53.6
	525	171	525	171	525	171	525	171	525	165	
	526	170	526	170	526	170	526	170			

	527	169	527	169	527	169	527	169			
MK09	311	110	310	112	311	110	309	114	307	96	62.4
			311	110			310	112	316	80	
							311	110	320	77	
									333	68	
									336	65	
MK10	215	51	214	53	215	51	215	48	211	48	78.9
	217	45	216	45	217	45	217	45	212	40	
	218	41	218	41	218	41	218	41	213	36	
									215	31	

Table 4 - 6 Number Non-dominate solutions (NNdS) and quality metric (QM) for BRdata

Instance	PF	Multi-Agent		MOGA		HSFLA		EPABC		PGDHS	
		NS/PF	QM%	NS/PF	QM%	NS/PF	QM%	NS/PF	QM%	NS/PF	QM%
MK01	2	1/1	50.00	2/2	100.00	2/2	100.00	2/2	100.00	2/2	100.00
MK02	3	1/0	0.00	2/2	66.7	2/2	66.7	2/2	66.7	3/3	100.00
MK03	3	1/1	33.33	1/1	33.33	1/1	33.33	1/1	33.33	3/3	100.00
MK04	5	2/0	0.00	3/2	40.00	3/1	20.00	3/1	20.00	4/4	80.00
MK05	7	2/0	0.00	4/3	42.86	4/3	42.86	4/3	42.86	6/6	85.71
MK06	1	2/0	0.00	3/0	0.00	2/0	0.00	2/0	0.00	1/1	100.00
MK07	6	2/0	0.00	4/4	66.67	2/2	33.33	2/2	33.33	5/5	83.33
MK08	2	4/0	0.00	4/0	0.00	4/0	0.00	4/0	0.00	2/2	100.00
MK09	5	1/0	0.00	2/0	0.00	1/0	0.00	3/0	0.00	5/5	100.00
MK10	4	3/0	0.00	3/0	0.00	3/0	0.00	3/0	0.00	4/4	100.00
Sum/Ave	38	19/2	8.33	28/14	34.96	24/11	29.62	26/11	29.62	35/35	94.90

Table 4 - 7 Diversification metric (DM) for 10 BRdata instances

Instance	Multi-Agent		MOGA		HSFLA		EPABC		PGDHS		
	mean	sd	mean	sd	mean	sd	mean	sd	min	mean	sd
MK01	0.56	0.79	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MK02	0.78	0.39	0.46	0.79	0.46	0.79	0.46	0.79	0.00	0.00	0.00
MK03	4.14	3.69	4.14	3.69	4.14	3.69	4.14	3.69	0.00	0.00	0.00
MK04	1.30	0.24	0.36	0.37	0.52	0.34	0.54	0.36	0.00	0.18	0.40
MK05	1.87	1.43	0.99	0.94	1.20	1.15	0.91	0.94	0.00	0.06	0.15
MK06	5.83	0.00	2.24	0.00	2.24	0.00	2.24	0.00	0.00	0.00	0.00
MK07	0.82	0.37	0.48	0.76	0.69	0.68	0.69	0.68	0.00	0.09	0.22
MK08	1.62	0.87	1.62	0.87	1.62	0.87	1.62	0.87	0.00	0.00	0.00
MK09	7.12	2.94	7.12	2.94	7.12	2.94	7.12	2.94	0.00	0.00	0.00
MK10	1.79	0.59	1.84	0.53	1.79	0.59	1.73	0.67	0.00	0.00	0.00

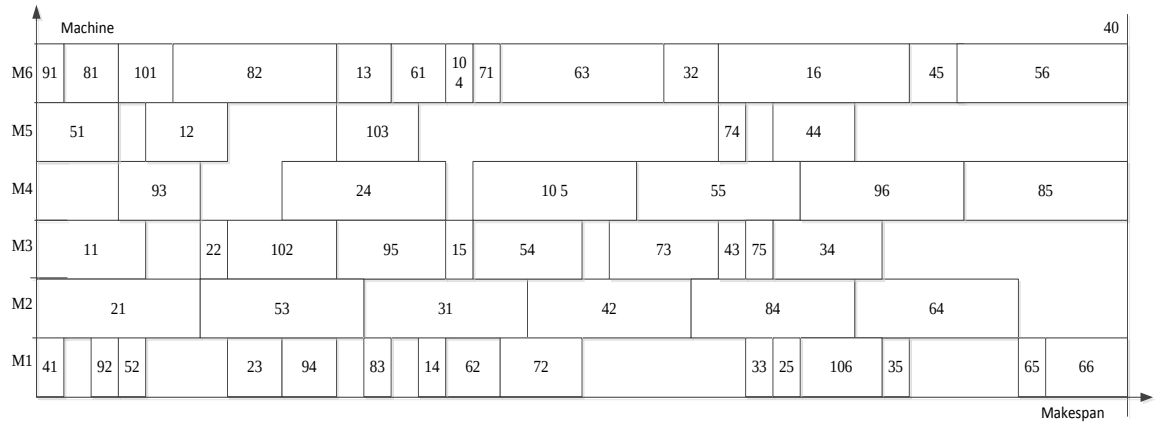

Fig. 4 - 7 Gantt chart of solution (40, 4) for MK01 instance 10×6

4.8 Conclusions

This Chapter proposed a grouping discrete harmony search algorithm for flexible job shop scheduling with makespan, and earliness-tardiness criteria. Two heuristics were developed for initializing the harmony memory. Sub-Harmony memory was employed to improve the diversity of harmony for a large population in harmony memory. Based on the problem characteristics and solution representation, a method was developed to improvise new harmonies. In order to improve the search ability and reduce the search space, two local search algorithms were proposed based on critical path and due date for makespan and E/T criterion, respectively. The large population and local search algorithms balanced the capability of exploration and exploitation. It has been demonstrated by simulation results based on the widely studied benchmarks and comparisons with the state-of-the-art algorithms that the proposed grouping discrete harmony search algorithm is better than the existing algorithms in terms of quality and the number of obtained overall non-dominated solutions.

Chapter 5

Two-stage ABC Algorithm for FJSSP with New Job Insertion

5.1 Introduction

This Chapter addresses flexible job-shop scheduling problem (FJSSP) with new job insertion. The new job insertion is one of seven important characteristics in remanufacturing. Scheduling problem in remanufacturing can be divided into two stages: scheduling and re-scheduling when a new job arrives. A two-stage artificial bee colony (TABC) algorithm is proposed for scheduling and re-scheduling with new job(s) insertion. The objective is to minimize maximum complete time (makespan). An ensemble local search is proposed to improve the convergence of TABC. Three re-scheduling strategies are proposed for new job insertion. Extensive computational experiments are carried out using fifteen well-known benchmark instances with eight instances from remanufacturing. Three re-scheduling strategies are compared and discussed. For scheduling performance, TABC is compared to several existing algorithms. For re-scheduling performance, TABC is compared to several simple heuristics and proposed hybrid heuristics. The results and comparisons show that the TABC algorithm is

effectiveness and efficiency for solving FJSSP with new job insertion. In the TABC algorithm, MAOS encoding and decoding method is employed to represent a solution.

The remainder of this Chapter is organized as follows. Section 5.2 describes the three rescheduling strategies. In Section 5.3, an ensemble local search algorithm is proposed. Section 5.4 presents the two-stage ABC algorithm framework. Experimental design, comparison and discussion are shown in Sections 5.5-5.7. We conclude this Chapter in Section 5.8.

5.2 Rescheduling strategies

In this section, we proposed three rescheduling strategies for the new job insertion. The three different rescheduling strategies are shown as follows:

1. Strategy I is just reschedule the new inserted job(s). Existing scheduling scheme remains. In rescheduling strategy I, the machines are available when all the assigned operations are completed. It means that machines may have different start time when rescheduling is implemented. Hence, machine start time must be considered for rescheduling in strategy I. The rescheduling problem becomes FJSSP with different machine starting times.
2. Strategy II is rescheduling both new insertion job(s) and the existing jobs' operations that are not started at the new job insertion time. The scheduling solution for existing jobs' operations may be changed. In strategy II, some machines may be performing some operations of existing jobs at the new job insertion time. The machines are available when the processing operations are completed. The existing jobs are also available for rescheduling when the corresponding operations are completed. It means that both machines and jobs have different start times when rescheduling is implemented. Hence, both machine start times and job start times must be considered for rescheduling in strategy II. The rescheduling problem becomes FJSSP with different machine starting times and different job starting times.
3. Strategy III is free-time-gap rescheduling based on existing scheduling scheme. Similar to Strategy I, the existing scheduling schema is remained. The new job(s) is (are) scheduled on all gaps of all machines after arrival time. It means that one machine is available if it has free-time-gap on the existing scheduling schema. In this condition, the machine can be employed for scheduling new job(s). After considered all free gaps on all machine, the same scheduling strategy in Strategy I is employed for scheduling remains operations. Compared to Strategy I, Strategy III has higher time complexity.

5.3 Ensemble local search

Critical path theory is employed for solving the FJSSP by many researchers. In one solution, there may be one or more critical paths. Each operation on one critical path is called a critical operation. This Chapter proposed an ensemble local search on a machine. The machines are sorted based on the number of critical operations processed on them. The first ensemble local search is executed on the machine with the maximum critical operations. There are five local search operators, one Insert, one Swap, two Inserts, two Swaps, one Insert and one Swap. These five operators are stored in an operator pool. In the beginning, each operator has the same ratio to be selected for generating a neighbor solution. The operator improving the solution will be remained in operator pool while the operator not improving the solution will be deleted from the operator pool. The ensemble local search will be terminated when all the operators have been deleted from operator pool. The next machine will be executed ensemble local search until the maximum local search iterations are met. The detail steps of ensemble local search can be described as follows:

1. Set $l = 1$.
2. Sort all machines non-increasing based on the number of critical operations.
3. Set $m = 1$.
4. Ensemble local search
   4.1 Select a local search operator from the operator pool.
   4.2 Generate neighbor solution.
   4.3 If the neighbor solution is better than the current one, remain the operator; otherwise, delete the operator from operator pool.
5. Set $l = l + 1$, if $l \le L$ (L is the maximum iteration), repeat (4) until operator pool is empty; otherwise, stop and return solution.
6. If the solution is improved, update the solution and go to (2).
7. Set $m = m + 1$, if $m \le M$ (M is machine number), go to (4); otherwise, go to (3).

The computational complexity to get a non-increasing machine sequence is $O(M\, log_2 M)$ and the complexity to search local space is $O(M * L)$. Hence, the computational complexity of the ensemble local search is $O(M(log_2 M + L))$.

5.4 Two-stage ABC algorithm framework

The TABC algorithm employs multiple initialization strategies, novel machine assignment and operation sequencing operator for generating new solutions, ensemble local search method to improve algorithm performance. The exploitation and exploration are balanced and stressed in this algorithm.

The proposed TABC algorithm includes two stages: scheduling stage and re-scheduling stage. At scheduling stage, the start time of all jobs and machines is at time 0. After scheduling stage, the solution with the best objective value will be output, and all operations will be processed on the corresponding machines based on the best solution. On new job(s) coming and inserting into the shop floor, the re-scheduling stage will be activated and reschedule the new job(s) and the operations that are not yet started. All not yet started operations and new job(s) will proceed based on the re-scheduling best solution. The computational procedure of TABC algorithm for scheduling and re-scheduling can be described as follows:

Scheduling stage:

1. Set parameters, including the number of employed bees, the number of onlooker bees and the number of scout bees. In TABC algorithm, the number of employed bees equal to the number of food sources.
2. Initialize population with multiple strategies shown in Section 4.2, evaluate each solution and determine the best food source.
3. Employed bee phase. For all populations, repeat the following sub-steps:
   3.1 For each pair of current solutions, generate two new solutions by using the strategy presented in Section 4.3.
   3.2 Improve the two new solutions based on the ensemble local search method in Section 4.4.
   3.3) If the new solutions are better than or equal to the current solutions, update the population.
4. Onlooker bee phase. For all populations, repeat the following sub-steps:
   4.1 Select two food sources for two onlooker bees by using tournament selection.
   4.2 Generate two new solutions for the two onlookers by using the strategy in Section 4.3.
   4.3 Improve the two new solutions based on the ensemble local search method in Section 4.4.
   4.4 If the new solutions are better than or equal to the current solutions, update the population.
5. Scout bee phase. If a solution has not been improved during the last limit number of trials, abandon it, generate a new solution randomly and execute the ensemble local search on it.
6. Update the best solution. If the termination criterion is reached, return the best solution; otherwise, go to step (3).

Re-scheduling stage:

7. Insert the new job(s) in the scheduled and executing sequence.
8. Calculate and record the start time of each job and each machine based on the inserting time.
9. For existing jobs, Re-calculate the number of operations that are not yet started. Add the new job(s) to the re-scheduling job set.
10. Execute re-scheduling.

11. Link up the re-scheduling results to executing scheduling results based on the inserting time.

5.5 Experiment setup

To test the performance of the proposed TABC algorithm, extensive experimental evaluation and comparisons with existing algorithms are provided using well-known FJSSP benchmark sets. Two sets of instances are considered in this Chapter: (1) the first data set are five Kacem instances [140], (2) the second data set, called BRdata, is a set of 10 problems by Brandimare[141]. The Kacem benchmark set is composed of five instances with the size ranging from 4 jobs, 5 machines and 11 operations to 15 jobs, 10 machines and 60 operations. The Brandimare benchmark set includes 10 problems with the size ranging from 10 jobs, 6 machines and 55 operations to 20 jobs, 16 machines and 240 operations.

Table 5 - 1 The data of eight remanufacturing instances

Instance	Inserting Order	Inserting time	Job Number	Machine Number	Operation number
1	0	0	5	4	23
	1	10	1		4
	2	15	1		5
2	0	0	8	8	64
	1	12	1		8
	2	24	1		8
3	0	0	10	6	81
	1	15	1		8
	2	25	1		7
	3	35	1		6
4	0	0	10	10	100
	1	15	2		20
	2	30	2		20
5	0	0	15	8	171
	1	21	2		18
	2	47	2		17
6	0	0	15	10	185
	1	14	2		20
	2	31	2		19
	3	44	2		19
7	0	0	20	10	308
	1	21	3		38
	2	52	3		32
8	0	0	20	15	355
	1	18	3		42
	2	34	3		42
	3	49	2		25

To test the performance of TABC for FJSSP with new job inserting and the three re-scheduling strategies provided in Section 5. 2, an instance set from remanufacturing enterprise is considered. This instance set includes eight instances with the size ranging from 5 jobs, 4 machines and 23 operations to 20 jobs, 15 machines and 355 operations. There are 35 new jobs inserted to existing scheduling

sequence of eight instances. Each inserting has different inserting time, job number and operation number. The detail information is shown in Table 5 - 1. The first two columns are instance number and new job insertion times. The third column is the inserting time of corresponding inserting order. The last three columns are the corresponding job number, machine number and operation number. The first row of each instance shows the job number, machine number and operation number at initial scheduling.

The TABC algorithm is coded in C++ and implemented on an Intel ® Core ™2 Duo CPU P8600 @ 2.40GHZ PC with 4 GB RAM. The population size is fixed at 50. The maximum generation is 3000. The probability of crossover operator is 0.4. The maximum iteration of ensemble local search is set to 5 times machine number. Each instance is carried out 30 replications.

5.6 Discussion and comparison of three re-scheduling strategies

For the eight instances from remanufacturing enterprise, the re-scheduling is based on the initiating scheduling or last time re-scheduling for new job(s) inserting. In this section, we just focus on the minimal makespan over 30 runs. To compare the three re-scheduling strategies, we also calculated the relative percentage increase (RPI) as follows:

$$RPI(C_M^i) = \frac{C_M^i - C_M^*}{C_M^*} \times 100 \tag{5-1}$$

Where C_M^i is the makespan value obtained in the ith replication, C_M^* is the best maksespan value found by three re-scheduling strategies. Obviously, the smaller the RPI value, the better result the re-scheduling strategy produces. To show the algorithm performance, we also calculated the average relative percentage increase (ARPI) of each re-scheduling strategy. For each instance, the average run time over 30 replications also recorded for the performance of TABC algorithm and three re-scheduling strategies.

To compare three re-scheduling strategies detailed, the eight remanufacturing instances are divided into small-scale set and large-scale set. The small-scale set include instance 1 to instance 4, and the larger set include instance 5 to instance 8. For each instance, the minimal makespan value (CM), the average run time over 30 times and the average relative percentage increase (ARPI) are shown in Table 5 - 2 and Table 5 - 3. The first row of each instance is the corresponding results of initiating scheduling, and the result values of three re-scheduling strategies are the same. The insert order of initiating scheduling is set as "0" while the each new job(s) inserting order is increased.

It can be easily seen from Table 5 - 2 and Table 5 - 3 that the re-scheduling strategy II was found to give the best makespan and ARPI values and for each new job(s) inserting. The re-scheduling strategy

II also obtained average APRI for small-scale, 0.45, and large-scale, 0.75, instance sets. The re-scheduling strategy I got about the same average APRI for small-scale, 31.39, and large-scale, 30.14, instance sets. The re-scheduling III obtained different average APRI for small-scale, 21.62 and large-scale, 18.85, instance sets. The re-scheduling strategy III has smaller average APRI value for large-scale instance set. Compared to re-scheduling I and III, the re-scheduling strategy II costs maximum summation-running time for all new job(s) inserting re-scheduling. The summation re-scheduling time for small-scale is 21.483 seconds, large-scale 279.157. The summation-running time of re-scheduling strategy I and III are no significant difference. Hence, the re-scheduling II found the best solutions with the maximum running time in three re-scheduling strategies. The re-scheduling strategy III obtained better solutions than re-scheduling strategy I with about the same time.

Table 5 - 2 Scheduling and re-scheduling results for small-scale instances

Instance	Inserting Order	Re-sche I			Re-sche II			Re-sche III		
		C_M	Time(s)	ARPI (%)	C_M	Time(s)	ARPI (%)	C_M	Time(s)	ARPI (%)
1	0	25	1.032	0.00	25	1.032	0.00	25	1.032	0.00
	1	36	0.156	33.33	27	0.469	0.37	36	0.312	33.33
	2	36	0.219	12.50	32	0.531	0.00	36	0.341	12.50
2	0	36	4.203	0.00	36	4.203	0.00	36	4.203	0.00
	1	63	0.235	38.64	44	2.343	0.00	53	0.419	20.45
	2	77	0.312	57.14	49	1.531	1.84	59	0.487	20.41
3	0	55	5.641	0.00	55	5.641	0.00	55	5.641	0.00
	1	77	0.281	32.76	58	3.640	0.17	77	0.646	32.76
	2	111	0.261	68.18	66	2.687	1.97	111	0.583	68.18
	3	111	0.247	58.57	70	2.172	1.14	111	0.602	58.57
4	0	44	7.938	0.00	44	7.938	0.00	44	7.938	0.00
	1	69	1.020	35.29	51	5.344	0.00	56	2.754	9.80
	2	103	1.111	71.67	60	2.766	0.33	75	1.673	25.00
Avg				31.39			0.45			21.62
Sum			3.842			21.483			7.817	

To illustrate the three re-scheduling strategies more clearly, the first time new job inserting re-scheduling of instance 2 is used as an example. Fig. 5 - 1 shows the initiating scheduling Gantt chart. The makespan value is 36. It means that all existing jobs will be completed and all machines are available at time "36" if there is no new job(s) inserting. The new job, Job 9, inserts at time "12". At this time, the machine release time, job release time and corresponding operation number are shown in Table 5 - 4. For example, the fourth operation of job 1, O1.4, is processing on machine M3 on time "12". This operation will be completed on time "13" on machine M3. Hence, the release time of job 1 is on time "13" and the start operation is O1.5. At the same time, the release time of machine M3 is

also on time "13". In the same way, all job release time and machine can be obtained. For the new job J9, the release time is on time "12" and the first start operation is operation O9.1.

The three re-scheduling strategies' Gantt charts are shown in Fig. 5 - 2 to Fig. 5 - 4. Fig. 5 - 2 is the Gantt chart of re-scheduling strategy I. The new inserting job, Job 9 is re-scheduling from time "36" and the finial complete time is 63. Fig. 5 - 3 shows the Gantt chart of re-scheduling strategy II. The re-scheduled operations include Job9's operations and the existing jobs' operations that have not started at the inserting time "12". Compared to Fig. 5 - 1, the existing jobs' operations have different and new operation sequence. The Job9's operations are also included in the new operation sequences. The final complete time is 44 by re-scheduling strategy II. The Gantt chart by re-scheduling strategy III is shown in Fig. 5 - 4. It can be seen that the existing jobs' operations are the same with initial scheduling, shown in Fig. 5 - 1. The first three operations of Job 9 are inserted into the free-time-gap of existing operation sequence. The fourth operation's start time is time "35" which is also earlier than the makespan value, unit time "36", in initial scheduling. The final complete time in Fig. 5 - 4 is 53.

Table 5 - 3 Scheduling and re-scheduling results for large-scale instances

Instance	Inserting Order	Re-sche I			Re-sche II			Re-sche III		
		C_M	Time(s)	ARPI (%)	C_M	Time(s)	ARPI (%)	C_M	Time(s)	ARPI (%)
5	0	79	19.813	0.38	79	19.813	0.38	79	19.813	0.38
	1	106	0.904	29.27	82	10.891	0.98	101	1.657	23.17
	2	132	0.817	48.31	89	5.391	1.12	114	1.490	28.09
6	0	73	25.157	0.00	73	25.157	0.00	73	25.157	0.00
	1	97	1.008	32.88	73	18.875	1.37	97	1.369	32.88
	2	115	0.939	49.35	77	11.250	1.04	110	1.425	42.86
	3	118	0.931	51.28	78	8.594	1.28	110	2.247	41.03
7	0	96	53.594	0.31	96	53.594	0.31	96	53.594	0.31
	1	140	2.423	26.13	111	41.391	0.45	140	3.163	26.13
	2	168	1.807	37.70	122	25.391	0.82	140	6.592	14.75
8	0	91	112.078	0.66	91	112.078	0.66	91	112.078	0.66
	1	116	2.865	24.73	93	71.578	0.00	104	4.036	11.83
	2	147	2.800	51.55	97	53.156	1.03	109	4.317	12.37
	3	161	2.237	69.47	95	32.640	1.05	123	3.941	29.47
Avg				30.14			0.75			18.85
Sum			16.731			279.157			30.237	

Table 5 - 4 Machine release time, job release time and corresponding operations

Order	1	2	3	4	5	6	7	8	9
Job release time	13	12	17	13	19	18	13	16	12
Job start operation	1.5	2.4	3.3	4.4	5.5	6.6	7.6	8.4	9.1
Machine release time	16	17	13	18	19	12	13	13	

It is clear that the re-scheduling strategy II is more appropriate than re-scheduling strategy III if the new Job 9 has the same priority with the existing jobs. The re-scheduling strategy III should be selected if the existing jobs are priority completed. In practical remanufacturing environment, re-scheduling strategy III may also be selected if the operations of existing jobs cannot be moved from assigned machine. Although three re-scheduling obtain different performance, each of them may be selected in practical shop floor.

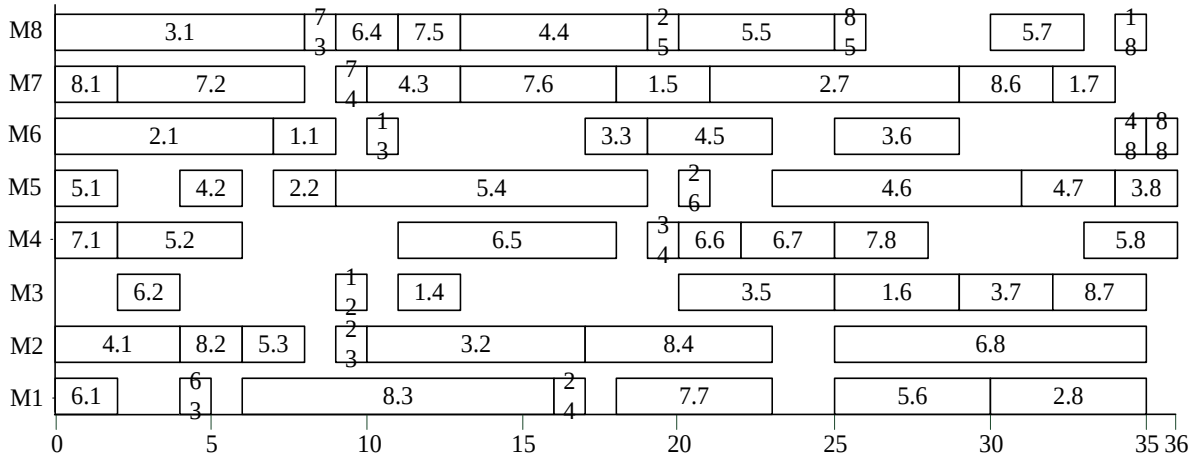

Fig. 5 - 1 The Gantt chart of initial scheduling

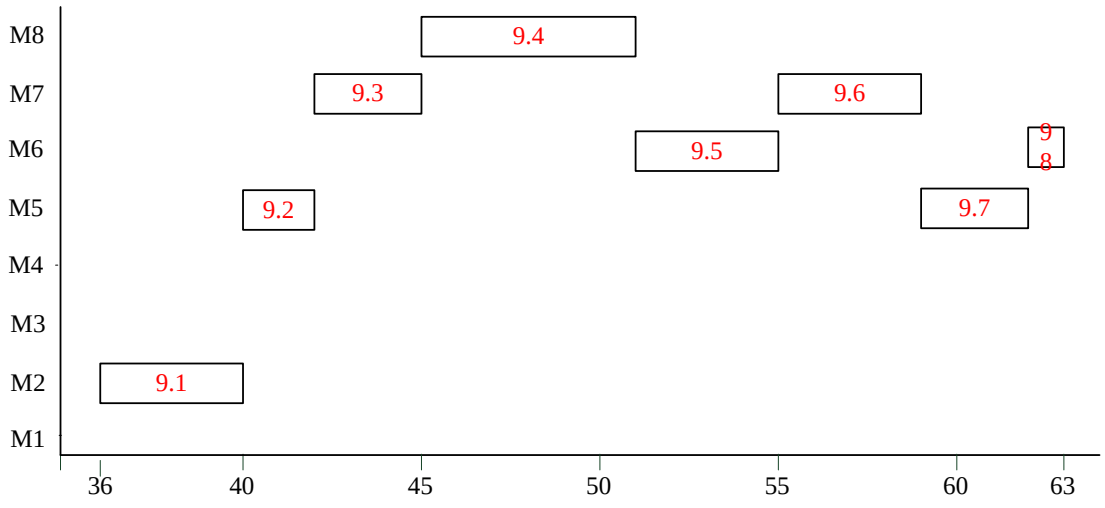

Fig. 5 - 2 The Gantt chart by re-scheduling strategy I

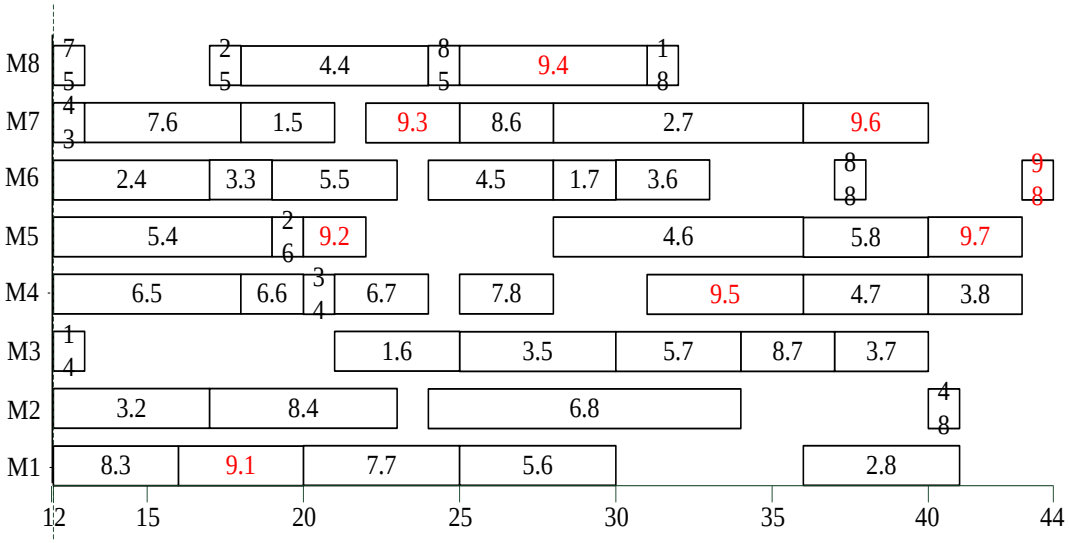

Fig. 5 - 3 The Gantt chart by re-scheduling strategy II

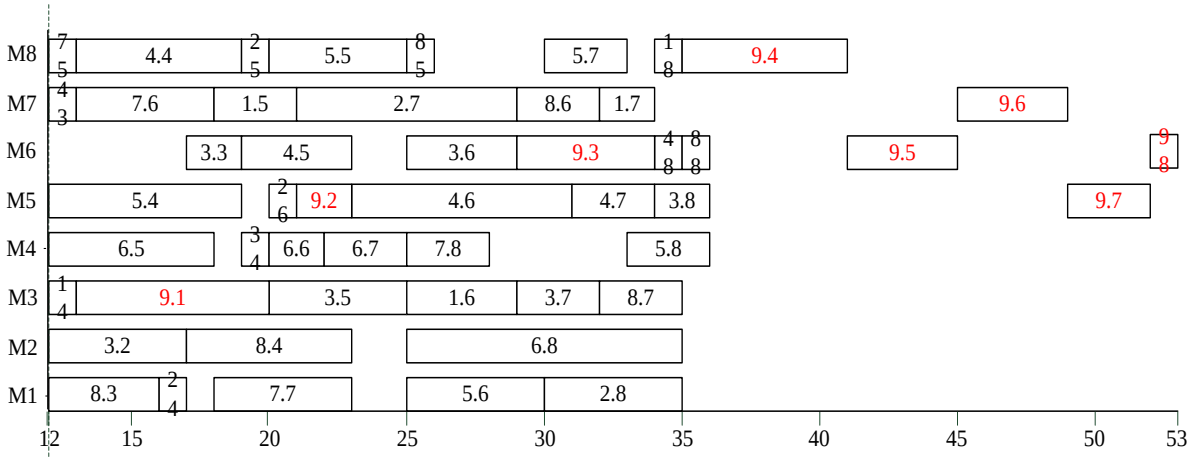

Fig. 5 - 4 The Gantt chart by re-scheduling strategy III

5.7 Comparison with existing algorithms

In scheduling phase, the job start time and machine release time are the same, zero. Hence, the problem is a general FJSSP. In re-scheduling phase, the job start time may be different and the machine release time may be different. The problem becomes FJSSP with different job start time and different machine release time. In addition, the job number is also different in different re-scheduling stages. In this section, we compare the TABC algorithm to several existing algorithms for FJSSP in scheduling phase. For the re-scheduling phase, we compare TABC algorithm with several simple and combination heuristics. The results and discussions are shown in sections 5.7.1 and 5.7.2.

5.7.1 Scheduling stage

There are many existing meta-heuristics for solving flexible job shop scheduling problem with makespan criterion. To test the performance of proposed algorithm, we compare TABC algorithm to five competitive meta-heuristics, parallel variable neighborhood search (PVNS) , knowledge-based ant colony optimization (KBACO) [22], tabu search algorithm with efficient neighborhood structure (TSPCB) [71], effective artificial bee colony algorithm (EABC) [136] and a simple and effective evolutionary algorithm (SEA) [51]. TABC algorithm is also compared to the DHS algorithm proposed in Chapter 4. The makespan values for all the algorithms are shown in Table 5 - 5. For five Kacem instances, all compared algorithms can obtain the best known makespan as long as the instances were solved. The only exception is that PVNS did not reach the best-known makespan for Kacme 5 with 15 jobs, 10 machines and 60 operations. For 10 BRdata instance, TABC and EABC are the two best algorithms. TABC obtained minimal makespan values for eight instances except MK01 and MK05 while EABC also reached eight minimal makespan values except instance MK01 and MK10. PVNS found minimal makspan for six instances, and TSPCB did it for four instances. Both KBACO and SEA found 3 minimal makespn values in 10 instances. Compare to DHS algorithm, TABC algorithm improves the results of instances 6, 7, 9 and 10. The result of instance 5 by TABC is larger than that by DHS algorithm. TABC algorithm is more competitive for large-scale instances. For each instance, the standard deviation of makespan value over 30 runs is shown in the second last column of Table 5 - 5. All the compared algorithms are run on different PC with different parameters and running time. In this Chapter, we just provide the average running time over 30 times of the proposed TABC algorithm to find the minimal makespan for corresponding instances. The detail information of running time for all benchmark instances is also shown in the last column of Table 5 - 5. Based on the makespan results of TABC and comparison to existing algorithms, the TABC algorithm is effective and efficiency for solving the flexible job shop scheduling problem. Hence, the TABC algorithm is also employed for re-scheduling when new job(s) come(s) and insert(s) into scheduled operation sequence.

Table 5 - 5 Results of two benchmark sets by all compared algorithms

Instance	PVNS	KBACO	TSPCB	EABC	SEA	DHS	TABC		
							C_M	SD	Avg_t (s)
Kacme1	N/A	11	11	11	11	11	11	0.00	0.47
Kacme 2	14	14	14	14	14	14	14	0.00	1.19
Kacme 3	N/A	11	11	11	11	11	11	0.00	1.2
Kacme 4	7	7	7	7	7	7	7	0.00	1.4
Kacme 5	12	11	11	11	11	11	11	0.00	2.97
MK01	40	39	40	40	40	40	40	0.53	3.36
MK02	26	29	26	26	26	28	26	0.81	3.72

MK03	204	204	204	204	204	204	204	0.00	1.56
MK04	60	65	62	60	61	60	60	0.83	66.58
MK05	173	173	172	172	173	172	173	0.00	78.45
MK06	60	67	65	60	65	67	60	2.02	173.98
MK07	141	144	140	139	140	143	139	0.91	66.19
MK08	523	523	523	523	523	523	523	0.00	2.15
MK09	307	311	310	307	311	309	307	1.81	304.43
MK10	208	229	214	208	225	212	202	4.98	418.19
No. C_M	2+6	5+3	5+4	5+8	5+3	5+4	5+8		

5.7.2 Re-scheduling for new job inserting

Several simple heuristics are proposed for machine assignment and operation sequencing in FJSSP. For example, machine with minimum workload heuristic (MSC) is for machine assignment while job with maximum remaining work (MSB) and job with maximum remain operations (MSC1) are two heuristics for operation sequencing. In this section, we improve some simple heuristics and develop combined heuristics based on simple heuristics. For example, random job order in machine with minimum workload heuristic (RMSC), MSC+MSB heuristic, RMSC+MSB heuristic, MSC+MSC1 heuristic, RMSC+MSC1 heuristic etc al. we also combine MSC1 heuristic with RH heuristic proposed in Section 4.2. MSC1 heuristic is employed to obtain the operation sequence in RH heuristic. Based on the initial scheduling results of simple heuristics and the combination heuristics, we compare TABC algorithm to MSC, RMSC, RMSC+MSB, RMSC+MSC1, RH and MSC1+RH heuristics in re-scheduling phase. The initializing scheduling and all re-scheduling results are shown in Table 5 - 6. MSC and MSC1 are simple heuristics and the results are the fixed. MSC1+RH heuristic also have fixed results. RMSC, RMSC+MSB, RMSC+MSC1 and RH heuristics run 30 times. We count minimum values (min), average values (ave) and standard deviation results by these four heuristics. TABC algorithm is also run 30 times and the results are fixed for the same instance. For each instance, the best solution is selected for next rescheduling operator.

It can be seen from Table 5 - 6 that MSC has the worst average (Ave) result for eight instances. RMSC and RMSC+MSB obtain quite minimum (min) values, average (ave) values and standard deviation (SD) values for initializing scheduling and all re-scheduling operators. RMSC+MSC1, MSC1+RH and RH have similar average values of min, ave and SD, RMSC+MSC1 (85.8, 93.6, 4.5), MSC1+RH (84.3) and RH (81.3, 90.3, 5.0). In these three heuristics, RH has the best Ave value of min (81.3) and the worst average value of SD (5.0). For all initializing scheduling and all re-scheduling operators, TABC obtain best results and best Ave value. Among all compared heuristics, only RH can find the best result of instance 1 at initializing phase.

For all compared algorithms, there are several special results. For example, MSC1+RH heuristic obtains a smaller first re-scheduling minimum result (71) than the initial result (77) for the second instance. It means that the re-scheduling results have much improvement after inserting the first new job into the second instance. MSC1+RH show the same situation for the fourth instance and the eighth instance. In addition, this heuristic has the same initial scheduling and the first re-scheduling results (100) for the eighth instance. TABC algorithm has the same initial scheduling and the first re-scheduling results (73) for the sixth instance. In the third re-scheduling operator for the eighth instance, TABC obtains a smaller result (95) than the second re-scheduling result (97). These situations mean that, after inserting new job, the re-scheduling operator may remain or even reduce makespan value in some special situation. To compare different heuristics and TABC algorithm clearly, Fig. 5 - 5 and Fig. 5 - 6 show the initial scheduling results and all re-scheduling results for the sixth instance and eighth instance. It is clear that TABC has better results than other heuristics in initial scheduling and re-scheduling operator phase.

In summary, TABC algorithm is effective and efficiency for solving FJSSP. For scheduling and re-scheduling FJSSP with new job inserting, combination heuristics and RH has better performance than simple heuristics. TABC algorithm obtains better results than simple and combination heuristics in both initial scheduling and re-scheduling phase.

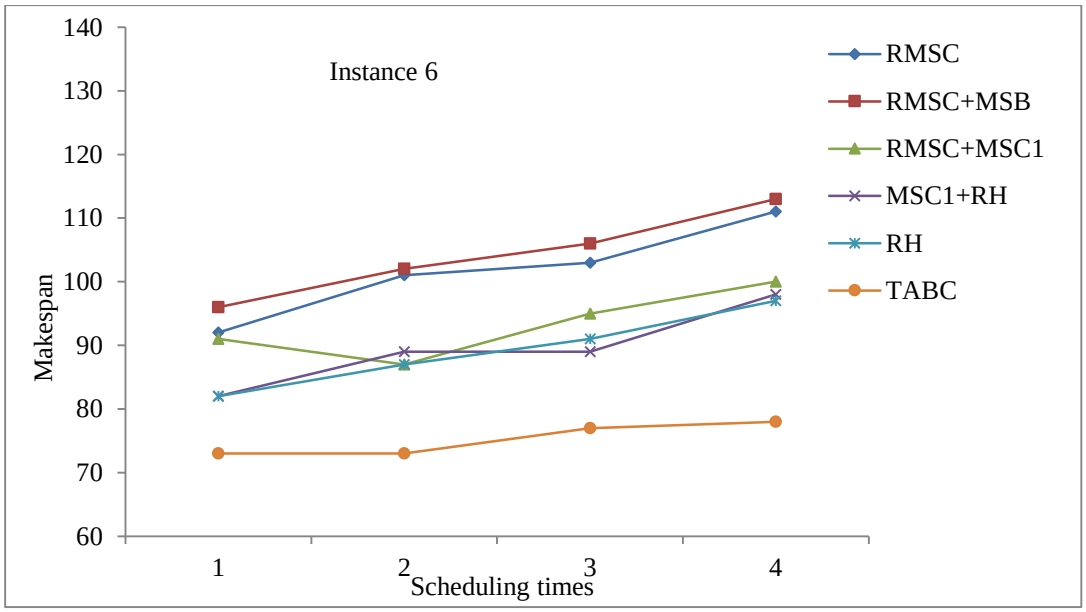

Fig. 5 - 5 Scheduling and re-scheduling results for instance 6

Table 5 - 6 Scheduling and rescheduling results

Instance	Insertion Order	MSC	RMSC			RMSC+MSB			RMSC+MSC1			MSC1 +RH	RH			TABC
			min	ave	SD	min	ave	SD	min	ave	SD		min	ave	SD	
1	0	39	27	34.3	4.03	28	35.6	3.81	27	29.4	2.77	28	25	33	4.39	25
	1	43	31	34.3	1.75	31	33.4	1.77	34	35.4	1.61	36	28	30.6	2.42	27
	2	56	34	39.7	3.41	35	40.2	3.45	35	38.1	2.07	44	33	37.2	3.59	32
2	0	90	48	57.3	5.66	50	58.9	6.04	49	58.7	5.06	51	44	51.6	3.12	36
	1	107	55	64.1	6.32	56	64.7	3.91	57	64.2	3.63	55	52	58.4	3.82	44
	2	102	58	63.2	2.62	60	67.7	4.79	63	68.4	4.15	62	56	63	5.37	49
3	0	106	70	81.8	7.01	71	82.1	4.78	69	78.1	4.97	77	67	80.9	7.34	55
	1	123	73	80.3	3.52	74	80.8	4.27	71	80.1	5.75	71	71	81.6	6.04	58
	2	131	84	94.6	4.87	88	97.1	4.77	78	86.8	4.71	80	80	91.6	5.61	66
	3	145	89	97.8	3.87	91	100.6	4.69	86	90.7	4.27	81	86	94.6	4.49	70
4	0	111	61	70.1	5.63	63	69.8	4.2	58	69.7	5.92	67	56	67.4	5.94	44
	1	116	69	75.8	5.04	70	79.7	4.77	67	76.7	5.57	64	61	71.7	4.59	51
	2	140	84	94.4	7.73	83	93.8	4.61	81	88.4	4.91	84	74	84.7	5.97	60
5	0	195	103	115.7	6.09	102	115.2	7.07	97	106	3.95	92	93	102.4	5.31	79
	1	199	115	125.5	5.95	112	122.3	5.56	110	115.9	4.15	101	101	111.3	5.63	82
	2	216	124	131.5	5.43	124	131.8	4.87	113	125	6.03	117	110	119.2	3.67	89
6	0	174	92	107.1	7.17	96	110.5	5.96	91	96.3	4.55	82	82	91.3	5.98	73
	1	184	101	111.5	7.18	102	110.9	5.85	87	99.1	5.78	89	87	95.6	5.13	73
	2	192	103	112.8	5.43	106	116.3	4.29	95	101.6	4.28	89	91	102.9	5.04	77
	3	220	111	117.2	4.29	113	123.4	6.64	100	106.9	4.48	98	97	105.1	6.53	78
7	0	263	122	137.3	6.81	119	134.7	8.41	120	130.9	3.64	117	117	124.4	4.77	96
	1	267	142	152.6	4.97	143	153.9	6.24	137	145.6	5.86	126	131	140.4	5.56	111
	2	303	152	165.9	6.62	153	163.9	5.26	144	152.6	5.16	136	140	147.8	4.63	122
8	0	223	110	121.2	5.16	111	121.1	7.12	99	110.7	5.29	100	94	106.2	7.24	91
	1	258	118	126.5	4.7	119	126.8	5.69	108	117.1	4.86	100	101	106.7	3.83	93
	2	275	134	141.1	4.05	130	139.4	5.45	118	126.6	4.47	116	108	117.5	4.93	97
	3	268	134	140.9	5.04	137	143	4.88	123	130.1	4.05	114	110	121.3	4.97	95
Ave		168.3	90.5	99.8	5.2	91.3	100.6	5.1	85.8	93.6	4.5	84.3	81.3	90.3	5.0	69.3

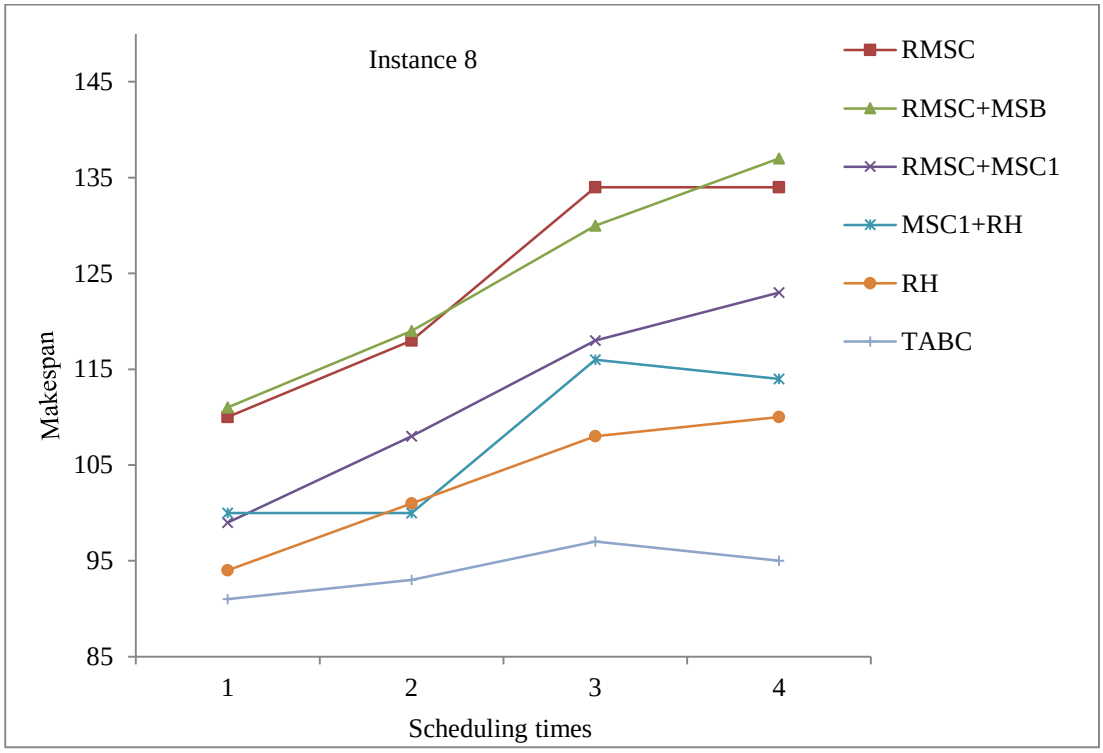

Fig. 5 - 6 Scheduling and re-scheduling results for instance 8

5.8 Conclusions

This Chapter proposed a two-stage hybrid artificial bee colony algorithm to solve flexible job shop re-scheduling problem with the arrival of new jobs. The objective of this Chapter was to minimize makespan (maximum completion time). Three re-scheduling strategies were proposed for rescheduling new jobs. The three re-scheduling strategies were compared in terms of the re-scheduling performance. The proposed algorithm compared against several existing algorithms on scheduling performance. The experimental results showed the effectiveness and efficiency of the proposed algorithm for solving the flexible job shop scheduling problem. For the re-scheduling performance, TABC algorithm compared against several simple heuristics and the proposed combined heuristics. The results and comparisons demonstrated improved the performance of TABC algorithm.

Chapter 6

Two-stage DHS Algorithm for FJSSP with Multiple Constraints

6.1 Introduction

Remanufacturing is a form of product recovery process and very important in waste management, material recovery and sustainable manufacturing. This Chapter researches a case study for the scheduling and rescheduling problem with processing time increasing and new job insertion constraints. The scheduling and rescheduling problems for re-processing are modeled as FJSSP with two constraints. This Chapter uses MAOS encoding and decoding method to represent a solution. An experience-based strategy is used to tackle the unpredictability of job processing time. Rescheduling strategy is considered for (1) processing time increasing and (2) new job insertion. A two-stage discrete harmony search (TDHS) algorithm based on problem characteristics is proposed for scheduling and rescheduling FJSSP. A local search method is employed for speeding up the convergence of TDHS. Two instances from a pump remanufacturing enterprise are solved. The objectives are to minimize the maximum completion time (makespan) and the mean of earliness and tardiness (E/T). These objectives are considered individually as well as together as a multi-objective

problem. As a case study, we select two instances from remanufacturing engineering and describe the solving in detail. The specific information of the two instances are provided. Computational results show that the proposed TDHS algorithm can solve the real remanufacturing problems effectively. Scheduling and rescheduling results are satisfactory and can be used in practice.

The rest of this Chapter is organized as follows: Section 6.2 introduces the processing time increasing and new job insertion constraints in remanufacturing. The rescheduling strategy is proposed in Section 6.3. Section 6.4 gives the detailed computational procedure of the two-stage DHS algorithm for scheduling and rescheduling. Section 6.5 sets up experiments. Section 6.6 is devoted to computational results obtained from real data and comparisons to existing approaches. We conclude this Chapter in Section 6.7.

6.2 Processing time increasing and new job insertion

Fig. 6 - 1 shows one scheduling solution with processing time increasing. The dotted rectangle after operation V2.1 is the increasing processing time of job V2.1 in practice. Because this change of job V2.1 affects the makespan criterion in this instance, the operations after V2.1 should be rescheduled.

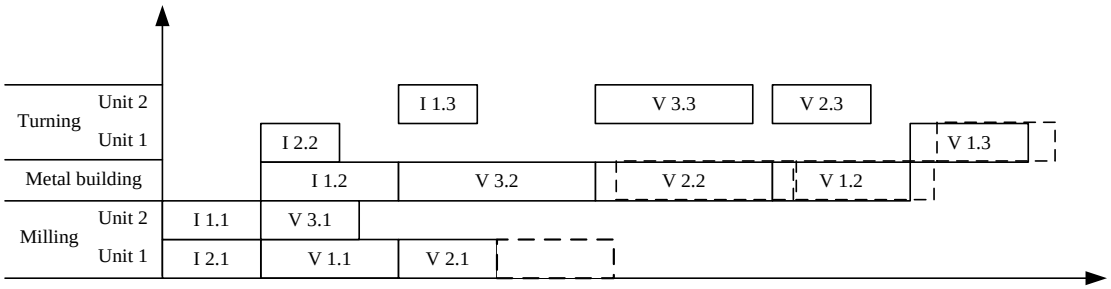

Fig. 6 - 1 A Gantt chart for increasing processing time

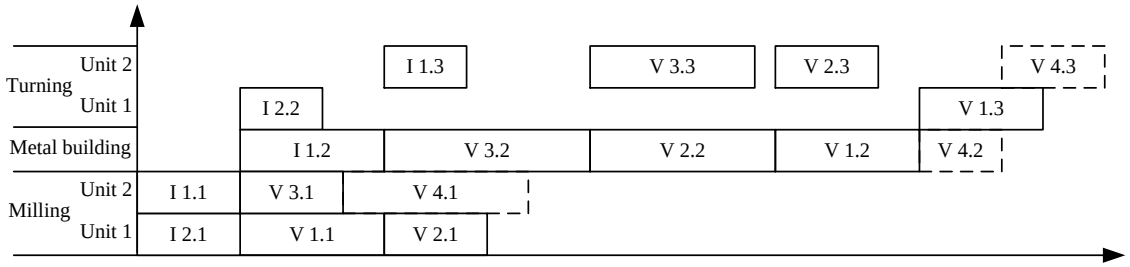

Fig. 6 - 2 A Gantt chart for new job insertion

The arrival of new products is also unpredictable in remanufacturing. When a new job inserts into the executing job sequence, the objective value and level satisfaction may also be affected. In this condition, we reschedule the operations that have not yet started at the insertion time. Fig. 6 - 2 shows

an example of inserting a new incoming job into the schedule and executing sequence. The job Volute 4 is inserted into the scheduled sequence. Obviously, the insertion of Volute 4 affects the value of makespan. The operations that have not started at the insertion time should be rescheduled.

6.3 Rescheduling strategy

Rescheduling is for processing time increasing or new job insertion when the schedule is being executed. There are two strategies for these two stochastic unexpected situations: (1) The processing time increasing or new job insertion does not affect the value of objective nor satisfaction. There is no rescheduling for the executing sequence. (2) The processing time increasing or new job insertion affects the value of objective or satisfaction level. The remaining operations that have not yet started will be rescheduled for improving results. As shown in Fig. 6 - 3, the value of makespan becomes larger after inserting job Volute 4 into the scheduled sequence. To obtain a better value of makespan, we should reschedule the operations which have not yet started at insertion time. The corresponding operations are filled with lines. Rescheduling results have to link up to the executing scheduling results. Therefore, the available time of each machine and start time of each job in rescheduling must be recorded previously.

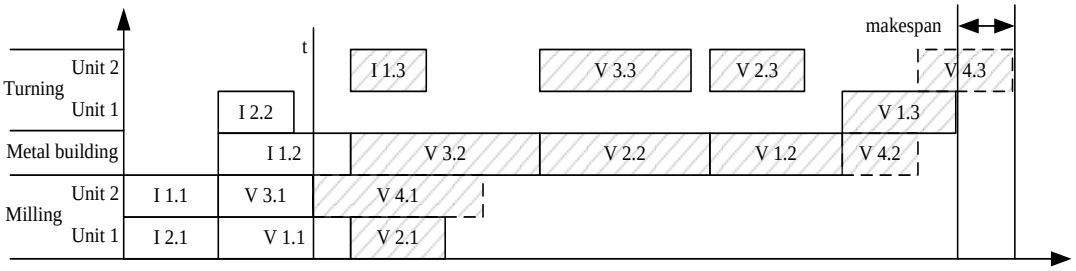

Fig. 6 - 3 Gantt chart for rescheduling

6.4 Computational procedure

The two-stage DHS algorithm employed dynamic grouping and local search operators in Chapter 4 to improve performance. The computational procedure of the two-stage DHS algorithm for scheduling and rescheduling can be described as follows:

Scheduling stage:

1) Initializing algorithm parameters and harmony memory.

2) Grouping harmony memory into many small Sub-HMs.

3) Sub-HM optimization

4) Improvising a new harmony.

5) Local search for a new harmony.

6) Update Sub-HM.

7) Repeat step 4) to step 6) till all Sub-HMs are executed.

8) If the termination condition is not met, go to step 9); otherwise, go to step 11).

9) All Sub-HMs form HM and corresponding results are recorded.

10) Repeat step 2) to step 9).

11) Output the final scheduling results.

Rescheduling stage:

12) The processing time inceasing or new job insertion in the executing sequence.

13) If the objective values are affected, go to step 14); otherwise, continue processing in practice.

14) Reinitializing the avaliable time of all machines and recording the operations that have not yet started.

15) Rescheduling the operations which have not yet started at the insertion time.

16) Link up the rescheduling results to executing scheduling results.

For the two-stage DHS algorithm, the computational complexity of the first stage is the same to the complexity of DHS in Chapter 4. When the processing time increases or new job is inserted into the executing sequence, the first stage will be executed. Hence, the computational complexity of the two-stage DHS algorithm is based on the number of times of processing time increasing or new job insertion.

6.5 Experimental setup

Table 6 - 1 and Table 6 - 2 show the jobs and the corresponding processing times in two Orders from a pump remanufacturing enterprise. In Table 6 - 1, there are two Volutes, two Impellers, one Shaft, two Shaft sleeves, three Case wear rings, and two Impeller keys. In Table 6 - 2, the jobs include two Volutes, three Impellers, two Shafts, three Shaft sleeves, four Case wear rings, three Impeller wear rings and three Impeller keys. Although the exact processing time of each job is unpredictable, experienced engineers can estimate it in advance. In Table 6 - 1 and Table 6 - 2, the processing times are the most probable processing time evaluated by engineers' experiences. In practice, if the processing time is larger than the most probable processing time, rescheduling will be executed.

In HS algorithm, the parameters HMS, HMCR and PAR affect the performance of HS algorithm. The HMS should be small. Omran [114] suggested that it was generally better to use a large value of HMCR (i.e., >=0.9). A large PAR value enhances the local exploitation ability of the algorithm while a small PAR value enlarges the search area and diversity of the HS algorithm. In this Chapter, the

parameters are fixed as follows: HMS=1000, HMCR=0.95 and PAR=0.5. The number of Sub-HM is 50. To show the performance of DHS algorithm, we compared the DHS algorithm to the GA algorithm [28] and EPABC algorithm [136]. All algorithms were coded in C++.net and run on Intel 2.8GHz PC with 1 GB memory. The two Orders were run for the 30 independent replications by three compared algorithms.

6.6 Scheduling results and comparisons

In this section, makespan, the mean of earliness and tardiness, and Pareto-based multi-objective criteria are considered. The size of re-processing problem in Order 1 is 12 jobs, 10 machines, and 44 operations while the corresponding values in Order 2 are 20, 12 and 91. Our purpose is to obtain high quality scheduling solutions for both criteria individually as well as together.

Table 6 - 1 The range of processing time of all jobs in Order 1

		Milling			Metal		Turning			Welding	
	Job	1	2	3	1	2	1	2	3	1	2
1	Volute 1	9	8	10	13	11	7	7		8	11
2	2	12	12	14	13	12		9	11		11
3	Impeller 1	8	9		10	9	10	9	8		
4	2		8	10		9	7	8	9		
5	Shaft 1	7	6			12		9	10	4	5
6	Shaft sleeve 1			8				8	9	4	
7	2	7		9			9	8		6	
8	Case wear ring 1		9	10	11	14	9		7		7
9	2		8	9	11	11		7	6	5	5
10	3		6	8	10			5	6	7	7
11	Impeller key 1	7	7		9	8	6	7		9	8
12	2		6	7	7	8		6	10	9	9

Table 6 - 2 The range of processing time of all jobs in Order 2

	Milling			Metal		Turning			Welding		Coating	
Job												
	1	2	3	1	2	1	2	3	1	2	1	2
Volute 1	8	6	8	10	8	6	5		6	10	5	6
2	11	10	12	12	11		8	9		9	6	4
3	4	3		9	6	7	6	7	10	8	4	
Impeller 1	7	8		8	7	9	8	6			7	6
2		6	9		7	5	7	6			5	6
3	7		6	6		12		11			3	5

Shaft 1	6		5			9		6	8	2		4	3	3
2			7	7		10	6	9		3		3	3	3
Shaft sleeve 1			6					7	7	3			4	5
2	6			7	5	6	7	5	7	5			4	5
Case wear ring 1			8	7	8	12	6		5			5	2	2
2			7	7	10	9		6	4	3		4	5	6
3			4	7	7			4	3	5		6	8	7
4			5	3		11	4			4		4	6	6
Impeller wear ring 1	4		2				8		7				4	5
2	4			3			6	6	9				3	5
3			4					7	6				4	
Impeller key 1	6		5		6	5	4	6		8		6	4	7
2			5	5	4	7		5	8	7		6		7
3			4	5	6	7	6	7		5		5		5

6.6.1 Makespan criterion

For Orders 1 and 2, the scheduling results by three algorithms are shown in Table 6 - 3. The first column of one algorithm is the minimum makespan value by this algorithm in 30 runs. The other three columns are the average value, maximum value and the value of standard deviation. It can be seen from Table 6 - 3 that DHS algorithm obtained the best min, avg, max and stdev values than GA and EPABC algorithms in 30 runs. The best makespan values of two Orders by DHS algorithm are 66 and 80 while the corresponding values by GA and EPABC algorithms are all 68 and 81.

For the jobs and corresponding processing times in Table 6 - 2, the values of makespan with iteration times by three algorithms are shown in Fig. 6 - 4. It can be seen from Fig. 6 - 4 that DHS algorithm can obtain the best result of makespan when the iteration count is between 109 and 127. The DHS algorithm has a better convergence than GA algorithm. The EPABC algorithm has better convergence than DHS in the beginning. However, the EPABC algorithm falls into a local optimum around 100th iteration. Compared to EPABC algorithm, DHS has a superior ability to escape from local optima to generate better solutions eventually.

To better guide the actual production in re-processing shop, Table 6 - 4 shows the completion time of all jobs in Order 1 and Order 2 when the values of makespan are 66 and 80. The average processing time of all jobs in Order 1 is 52 while the corresponding value of Order 2 is 59.

Table 6 - 3 The results of Makespan criteria by three algorithms

Order	EPABC				GA				DHS			
	min	ave	max	stdev	min	ave	max	stdev	min	ave	max	stdev
1	68	69	70	1.000	68	69.2	70	0.687	66	66.6	67	0.493
2	81	82.5	85	1.269	81	82.6	84	0.966	80	80.4	81	0.5163

6.6.2 E/T scheduling results

We discuss the E/T criterion for Order 1 and Order 2 in this section. The scheduling results of E/T by three algorithms are given in Table 6 - 5. The first column of one algorithm is the minimum E/T value by this algorithm in 30 runs. The other three columns are the average value, maximum value and the standard deviation value. It can be seen from Table 6 - 5 that DHS algorithm obtains the best min, avg, max and stdev values. The best E/T values of two Orders by DHS algorithm are 8 and 14 while the corresponding values by GA and EPABC algorithms are (10, 19) and (10, 18), respectively. The values of the best E/T of Order 1 and Order 2 are 8 and 14 and are not approximately zero (0). However, the results obtained by DHS algorithm are satisfactory in practice.

To better guide the actual production in re-processing shop, Fig. 6 - 5 shows the Gantt chart for Order 1 with the best E/T value (8). In Table 6 - 6, the second column and the fifth column show the due date of all jobs in Order 2 while the third column and the sixth column show the completion time of all jobs with the best value of E/T (14).

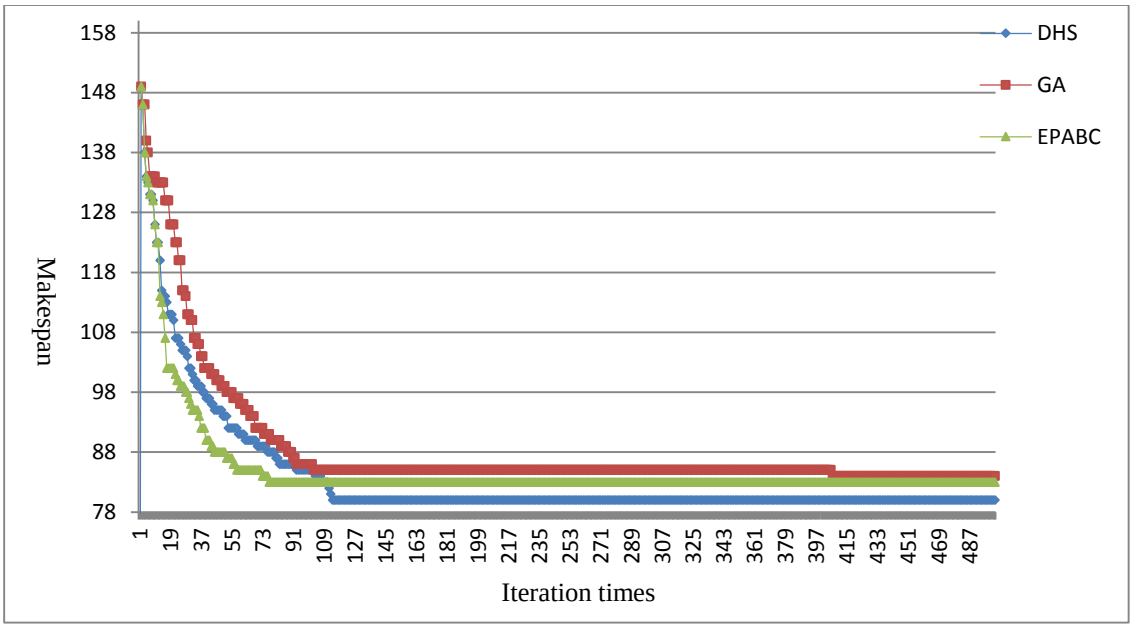

Fig. 6 - 4 The value of makespan with iteration

Table 6 - 4 The complete times of all jobs with best makespan in two Orders

job	complete time	job	complete time
Volute 1	34	Volute 1	37
2	47	2	76
Impeller 1	66	3	48
2	66	Impeller 1	67
Shaft 1	49	2	38
Shaft sleeve 1	53	3	42
2	41	Shaft 1	70
Case wear ring 1	54	2	80
2	58	Shaft sleeve 1	52
3	65	2	75
Impeller key 1	64	Case wear ring 1	80
2	29	2	62
avg	52	3	33
Makespan	66	4	54
		Impeller wear ring 1	67
		2	78
		3	56
		Impeller key 1	60
		2	30
		3	72
		avg	59
		Makespan	80

Table 6 - 5 The results of E/T criteria by three algorithms

Order	EPABC				GA				DHS			
	min	ave	max	stdev	min	ave	max	stdev	min	ave	max	stdev
1	10	11	12	0.666	10	10.7	12	0.674	8	8.8	9	0.421
2	18	20.3	22	1.251	19	20.2	21	0.788	14	14.4	15	0.516

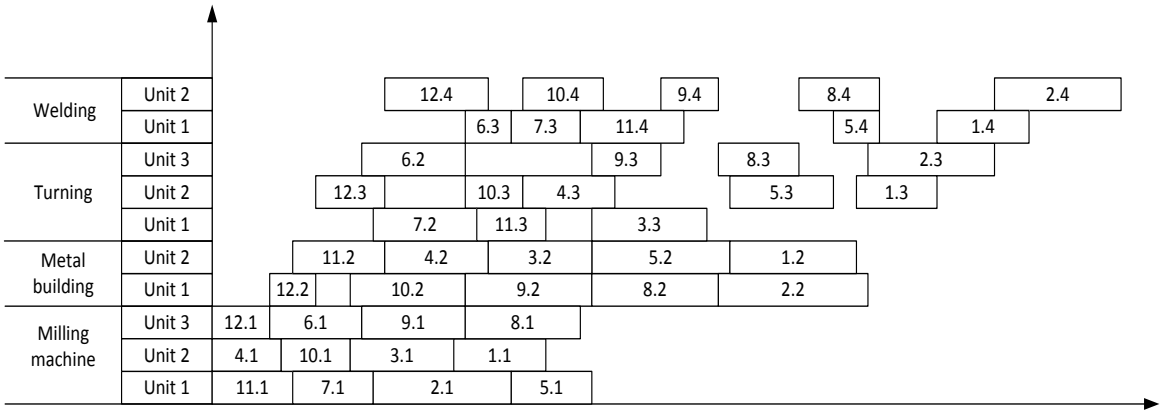

Fig. 6 - 5 The Gantt chart of Order 1 with E/T=8

Table 6 - 6 The completion times of all jobs with best E/T in Order 2

Job	Due date	Completion time	Job	Due date	Completion time
Volute 1	51	75	Case wear ring 1	43	43
2	66	90	2	43	63
3	43	49	3	40	57
Impeller 1	40	41	4	43	33
2	37	49	Impeller wear ring 1	21	26
3	40	56	2	21	22
Shaft 1	40	70	3	21	32
2	45	79	Impeller key 1	40	67
Shaft sleeve 1	30	25	2	43	63
2	40	40	3	39	73

6.6.3 Pareto-based multi-objective scheduling

For multi-objective optimization, the purposes are to obtain more non-dominated solutions that satisfy multiple objectives with good proximity and diversity with respect to the true Pareto front. Table 6 - 7 shows the Pareto-based non-dominated solutions obtained by three algorithms for Order 1 and Order 2. It can be seen from Table 6 - 7 that DHS algorithm obtains 6 solutions for Order 1 while the corresponding values by GA and EPABC are 4 and 6, respectively. There are 6 non-dominated solutions, (77, 8), (75, 9), (74, 10), (72, 11), (68, 18), and (66, 19) for Order 1. Three algorithms can

all obtain the non-dominated (68, 18). Only DHS algorithm can obtain 6 non-dominated solutions. For Order 2, the number of solutions by DHS algorithm is eight while the corresponding values by GA and EPABC are 7 and 6. There are 9 non-dominated solutions, (90, 14), (87, 15), (86, 16), (85, 17), (84, 18), (83, 19), (82, 32), (81, 40), and (80, 43) for Order 2. GA and EPABC algorithms have the same non-dominated solution (81, 40). DHS obtained 8 additional non-dominated solutions except the non-dominated solutions (81, 40). Hence, DHS algorithm finds more non-dominated solutions than GA and EPABC. For the merits of non-dominated solutions by three algorithms, Fig. 6 - 6 shows the non-dominated solutions by three algorithms for Order 2. It can be easily seen from Fig. 6 - 6 that the quality of non-dominated solutions obtained by DHS algorithm is better than those obtained by GA and EPABC algorithms. In addition, the best single objective solutions for two Orders are included in the non-dominated solutions obtained by DHS algorithm. For example, the best values of makespan for Order 1 and Order 2 are 66 and 80 while the corresponding values of E/T are 8 and 14. These four best single objective solutions correspond to the non-dominated solutions (66, 19), (80, 43), (77, 8) and (90, 14). Hence, the re-processing scheduling problem corresponding to Order 1 and Order 2 can be satisfactorily solved with respect to different single objective criterion and Pareto-based multi-objective criteria by DHS algorithm.

Table 6 - 7 Pareto-based non-dominated solutions by three algorithms

Order	EPABC		GA		DHS	
	Makespan	E/T	Makepsan	E/T	Makespan	E/T
1	80	10	79	10	77	8
	75	11	75	12	75	9
	74	12	72	13	74	10
	70	14	68	18	72	11
	69	17			68	18
	68	18			66	19
2	90	18	93	17	90	14
	89	19	92	19	87	15
	86	20	91	22	86	16
	85	22	85	24	85	17
	83	33	84	31	84	18
	81	40	83	37	83	19
			81	40	82	32
					80	43

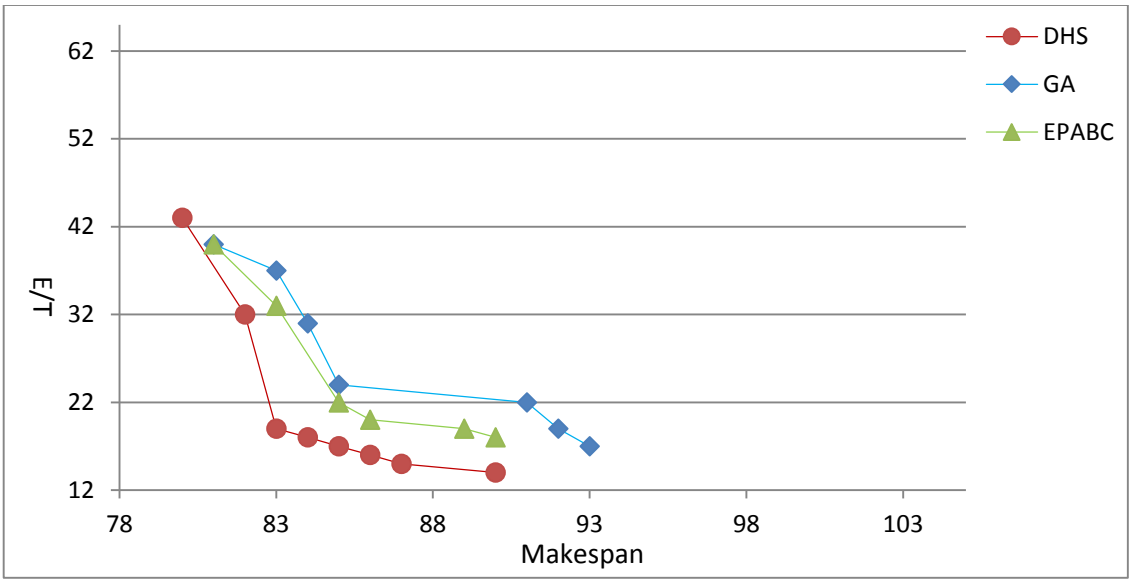

Fig. 6 - 6 Pareto-based non-dominate solutions

6.6.4 Re-scheduling for processing time increasing

In this section, we discuss the rescheduling problem for the processing time increasing in practice. According to scheduling results, the jobs will be processed in real re-processing shop floor. The processing time of one operation is estimated as a most probable processing time in advance. However, the processing time may be larger than the most probable processing time in some cases.

As shown in Fig. 6 - 8, the final scheduling result of makespan for Order 1 is 66 while the value of E/T in this schedule is 15. When the jobs in Order 1 are processed in shop floor, the processing time of operation 2.2 (the second operation of job Volute 2) increases from 13 time units to 16 time units. If there is no rescheduling, the final Gantt chart after this change is shown in Fig. 6 - 7. The value of makespan is 68 and the increasing of E/T criterion is 16.2. After completion of operation 2.2, the operations which can be re-scheduled are 1.4, 2.3, 2.4, 3.2, 3.3, 4.2, 4.3, 6.2, 6.3, 8.2, 8.3, 8.4, 9.3, 9.4, and 10.4. After rescheduling, the Gantt chart is shown in Fig. 6 - 10. The rescheduled operations are shown in dotted rectangles. The value of makespan after rescheduling is the same with that in Fig. 6 - 7. The increase of E/T criterion reduces from 16.2 to 15.2.

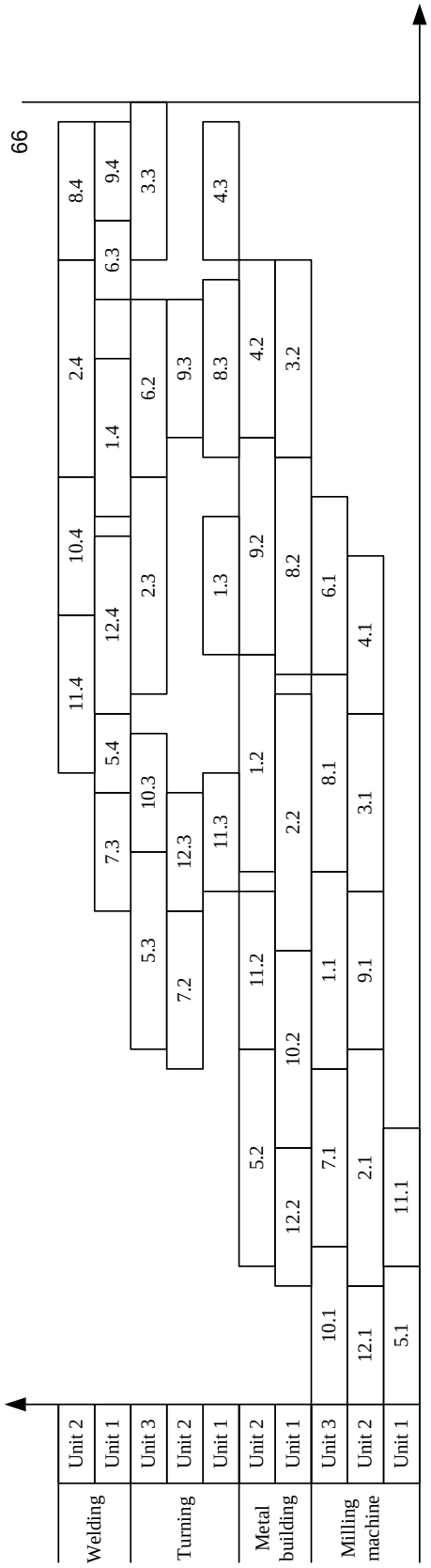

Fig. 6 - 8 The Gantt chart of Order with the makespan = 66 and E/T = 15

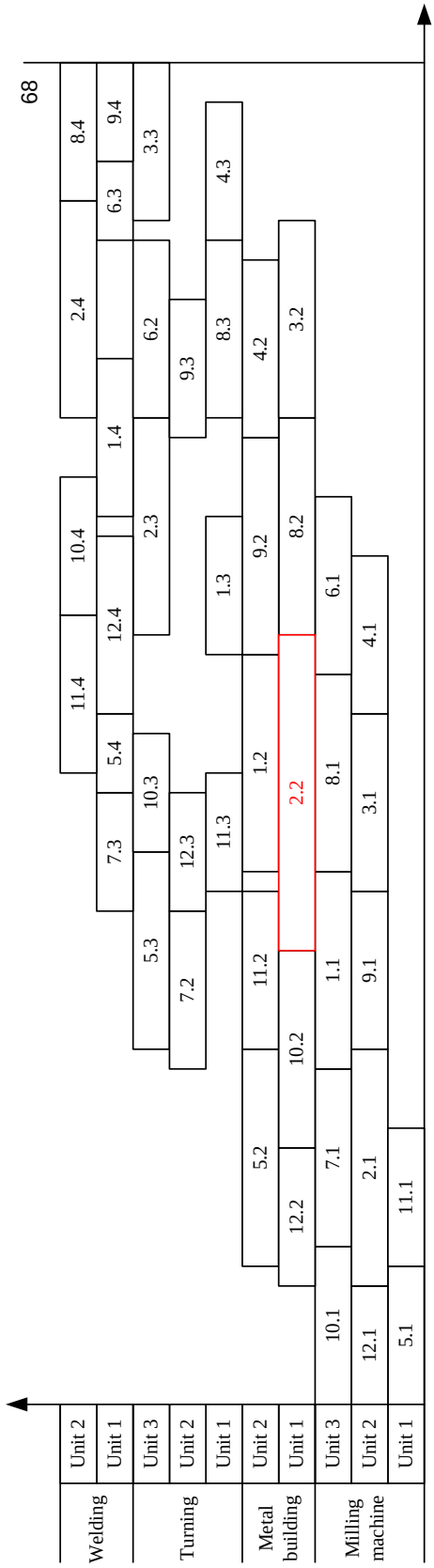

Fig. 6 - 7 The Gantt chart with no re-scheduling

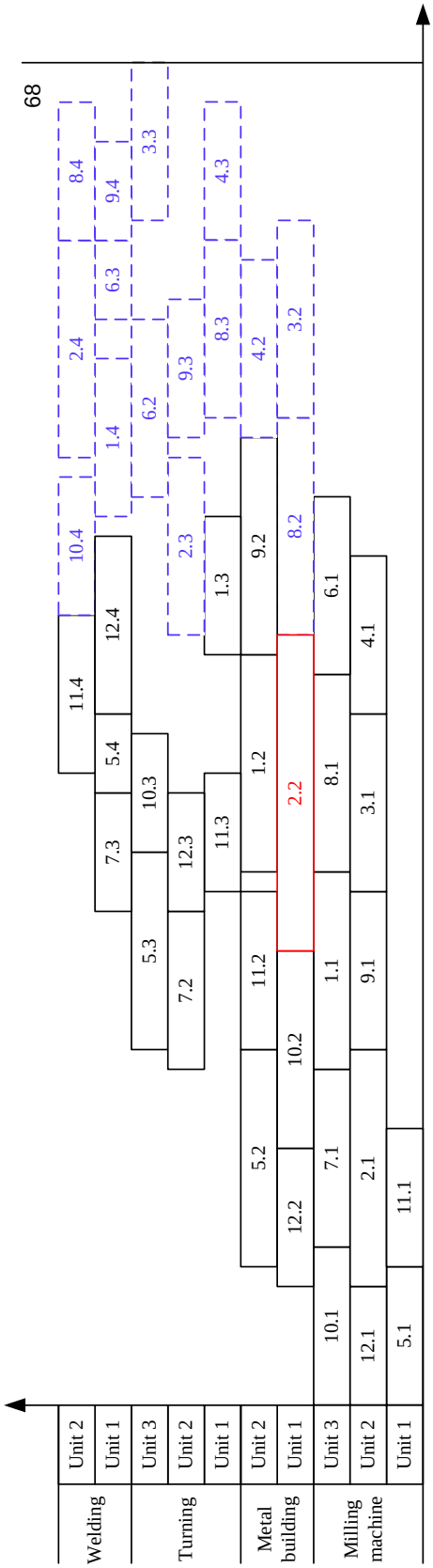

Fig. 6 - 10 The Gantt chart with re-scheduling

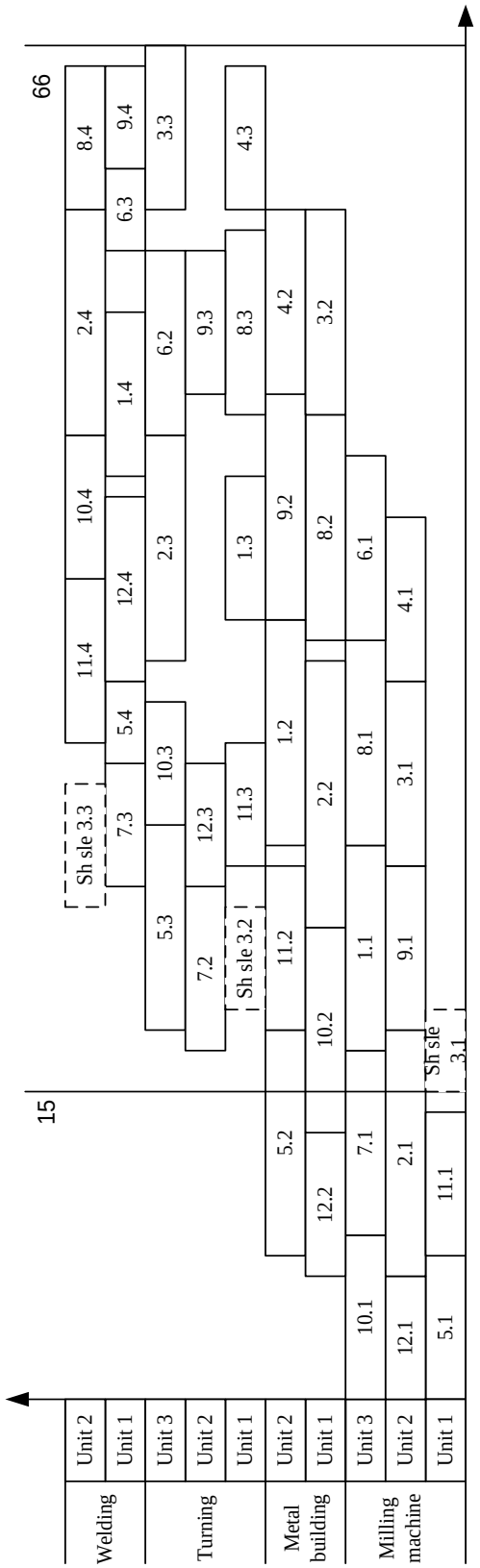

Fig. 6 - 9 The Gantt chart after inserting job Shaft sleeve 3

6.6.5 Re-scheduling for a new job insertion

In this section, we discuss the rescheduling in practice when a new job is inserted into the executing sequence. According to scheduling results for the Order 1, the value of makespan is 66 and E/T is 15. The Gantt chart is shown in Fig. 6 - 8. When the jobs in Order 1 are re-processed in the shop, two new incoming jobs Shaft sleeve 3 and Shaft 2 are inserted. The operations and corresponding processing times are shown in Table 6 - 8. The Due dates of Shaft sleeve 3 and Shaft 2 are 15 and 26.

The job Shaft sleeve 3 is inserted into the scheduled sequence at 15 time units. After insertion, the Gantt chart is shown in Fig. 6 - 9 and the operations of Shaft sleeve 3 are in dotted rectangles. The value of makespan is not affected. The start time and completion time of Shaft sleeve 3 are 15 and 31. Therefore, the tardiness time of Shaft sleeve 3 is 1. The E/T values of all scheduled jobs are also unchanged. Hence, there is no need for rescheduling.

The Shaft 2 is inserted into the sequence at 32 time units. After insertion, the Gantt chart with no rescheduling is shown in Fig. 6 - 11. The operations of the Shaft 2 are also shown in dotted rectangles. The first operation of Shaft 2 is re-processed on unit 1 of Milling machine and the processing time is 4 time units. The third operation is re-processed on unit 1 of Turning machine. Both of the two operations are re-processed on the machines which have the minimum processing times among the selectable machines. The second and fourth operations have only one selectable machine. The value of makespan increases from 66 to 77. The start time and completion time of Shaft 2 are 32 and 77. The tardiness time of Shaft sleeve 3 is 45. Hence, the operations which are not yet start at time 32 will be rescheduled. Fig. 6 - 12 shows the Gantt chart after rescheduling. There are nine operations changed their start times or processing machine, such as 2.2- 2.4, 8.2-8.4, 3.2-3.3 and 4.3. The completion times of job 3, 4, and 8 are increased from 66, 65, 65 to 72, 66, and 70, respectively. However, the completion time of Shaft 2 decreases from 77 to 63. Hence, the E/T criterion is not affected obviously. The value of makespan reduces from 77 to 72. The value of makespan is still larger than 66, but it has been reduced from 77 to 72 by rescheduling.

Table 6 - 8 The operators and corresponding range of processing times of Shaft 2

	Milling			Metal		Turning			Welding	
Job	1	2	3	1	2	1	2	3	1	2
Shaft sleeve 3	3		4			4	4	5	6	5
	4		5			5	6	6	7	6
Shaft 2	3	4	5	5		6	8			4
	4	5	6	7		7	10			5

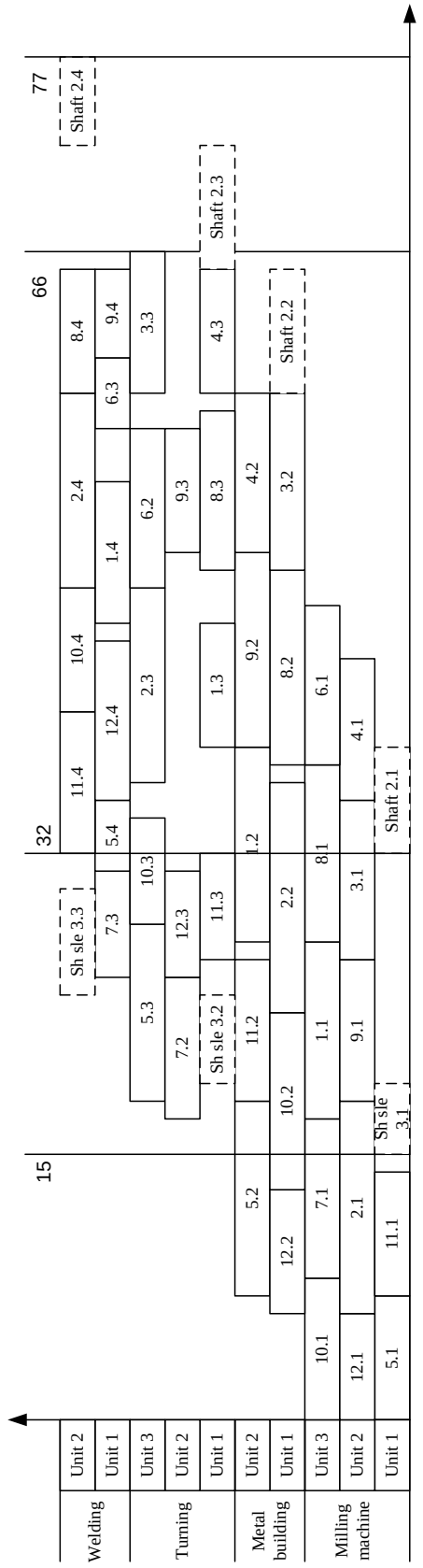

Fig. 6 - 11 Gantt chart after inserting job Shaft sleeve 3

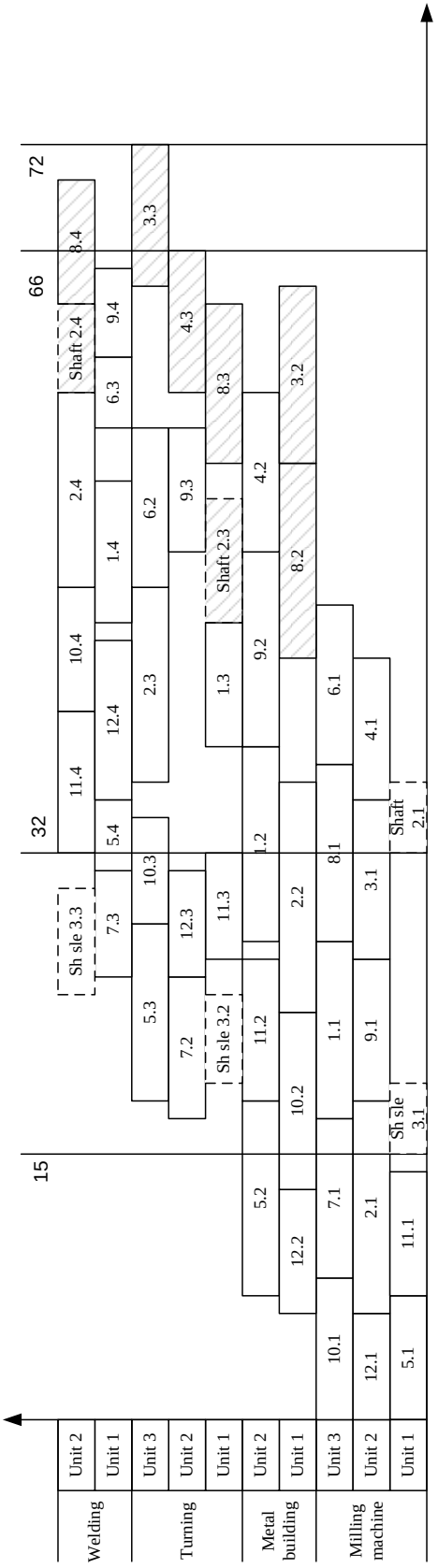

Fig. 6 - 12 Gantt chart after rescheduling for job Shaft sleeve 3

6.7 Conclusions

This Chapter investigated the re-processing scheduling and rescheduling problem in remanufacturing system. The scheduling and rescheduling problem of re-processing were modeled. Considering the processing time of each job is unpredictable, the processing time was estimated as a most probable value by experience. Discrete harmony search algorithm and local search method were employed for this problem. Rescheduling strategy was proposed for processing time increasing and new job insertion in practice. Makespan, mean of earliness and tardiness objectives were considered. As a case study, we solved two instances from real-life remanufacturing engineering and described the process in detail. The results demonstrate DHS algorithm can effectively solve scheduling and rescheduling problems in practice.

Chapter 7

FDHS Algorithm for FJSSP with Fuzzy Processing Time

7.1 Introduction

In this Chapter, we propose an effective fuzzy discrete harmony search (FDHS) algorithm to solve the FJSSP with fuzzy processing time. The objective is to minimize the maximum fuzzy completion time. A simple and effective heuristic, namely MinEnd, is proposed to initialize population. To test performance, MinEnd is compared to several existing heuristics. FDHS is tested using existing benchmark cases and real instances from remanufacturing. The experimental results and comparisons demonstrate the performance of FDHS.

The rest of this Chapter is organized as follows: Section 7.2 briefly describes the objective of FJSSP with fuzzy processing time constraint. The MinEnd heuristic is presented in Section 7.3. Section 7.4 presents the method for improvising new harmony in FDHS. The framework of the proposed FDHS algorithm is given in Section 7.5. Section 7.6 is for experimental results and comparisons. We conclude this Chapter in Section 7.7.

7.2 FJSSP with fuzzy processing time

The FJSSP with fuzzy processing time means that the operation processing time is not an exact value and the processing time is represented with triangular fuzzy number (TFN) as follows:

$$t_{i,j,k} = (t^1_{i,j,k}, t^2_{i,j,k}, t^3_{i,j,k}) \tag{7-1}$$

where $t^1_{i,j,k}$, $t^2_{i,j,k}$ and $t^3_{i,j,k}$ are three probable processing times of operation $O_{i,j}$ on machine M_k. The fuzzy completion time of operation $O_{i,j}$ is also a TFN as follows:

$$C_{i,j} = (C^1_{i,j}, C^2_{i,j}, C^3_{i,j}) \tag{7-2}$$

where $C^1_{i,j}$, $C^2_{i,j}$ and $C^3_{i,j}$ are three probable completion times of operation $O_{i,j}$.

The objective is to minimize maximum fuzzy completion time. The maximum fuzzy completion time denoted by C_M can be calculated by the following formula.

$$\text{Min } C_M = \max_{1 \le i \le n} \{C_i\} \tag{7-3}$$

where C_i is the fuzzy completion time of job i.

7.3 Proposed initializing rule

For meta-heuristics, the quality of initial population often affects the speed of convergence. Therefore, it is a critical step to generate a high quality initial population. For machine assignment, there are random rule, local minimal processing time (LS) rule and global minimal processing time rule (GS). For operation sequencing, random rule, most work remaining rule (MReW) and most number of operations remaining (MReO) rule were proposed in literature [16].

In this Chapter, we proposed a new heuristic, named MinEnd heuristic, for initializing population. In this heuristic, the operation sequence is decided randomly. The processing machine for each operation is assigned based on the operation sequence. The steps of this heuristic are shown as follows:

Step1: All operations of all jobs are shuffled randomly to obtain an operation sequence (OS).

Step2: Repair OS, make sure the operations of the same job can satisfy the processing precedence.

Step3: For each operation $O_{i,j}$, evaluate the fuzzy completion time on each selectable machine.

Step4: The machine with minimum fuzzy completion time is selected for processing operation $O_{i,j}$.

The minimum computational complexity to obtain a random operation sequence is $O(\sum_{i=1}^{n} q_i)$. The complexity to repair operation sequence is $O(\sum_{i=1}^{n} q_i * log_2 \sum_{i=1}^{n} q_i)$. The complexity to select machine for all operatons is $O(\sum_{i=1}^{n} q_i)$. Hence, the computational complexity of the MinEnd heuristic

is $O(\sum_{i=1}^{n} q_i * (2 + log_2 \sum_{i=1}^{n} q_i))$. To describe this rule clearly, we take the problem shown in Table 7 - 1 to represent the heuristic rule in detail. In Table 7 - 1, there are three jobs, three machines and ten operations. First, we generate an operation sequence randomly. The operation sequence is as $\mathrm{OS} = \{(3,2),(2,1),(1,3),(1,1),(2,2),(2,4),(3,3),(2,3),(3,1),(1,2)\}$. In step2, the operations sequence is repaired to satisfy the processing precedence of the same job's operations. The operation sequence is changed as $OS = \{(3,1),(2,1),(1,1),(1,2),(2,2),(2,3),(3,2),(2,4),(3,3),(1,3)\}$. Step3 is to select processing machine for each operation. For the first operation $(3,1)$, selectable machines are M_1 and M_2.The fuzzy completion time on machine M_1 is (2, 4, 8) while the fuzzy completion time on M_2 is (4, 5, 7). Based on the fuzzy TFNs ranking criteria, (4, 5, 7) is larger than (2, 4, 8). Hence, M_1 is selected for operation $(3,1)$. The available time of M_1 for next operation is (2, 4, 8). The second operation is $(2,1)$. The fuzzy completion on selectable machine M_2, M_3 and M_4 are (5, 7, 9), (1, 2, 5) and (1, 2, 3), respectively. Machine M_4 is selected and the available time becomes (1, 2, 3). For third operation $(1,1)$, the selectable machines are M_2 and M_4. The fuzzy completion time on M_2 is (5, 6, 7). Because the available time of M_4 is (1, 2, 3), the fuzzy completion time on M_4 is the addition of (1, 2, 3) and (3, 5, 6). Based on the TFNs addition operation, the fuzzy completion time is (4, 7, 9). Based on the TFNs ranking criteria, (4, 7, 9) is larger than (5, 6, 7). So, M_2 is selected for operation $(1,1)$ and the available time becomes (5, 6, 7). All remaining operations can be assigned to one processing machine in the same way. The final operation order on each machine are $M_1 = O_{3,1}, O_{1,2}, O_{2,4}$, $M_2 = O_{1,1}, O_{3,3}$, $M_3 = O_{3,2}, O_{1,3}$ and $M_4 = O_{2,1}, O_{2,2}, O_{2,3}$. The Gantt chart of fuzzy completion time is shown in Fig. 7 - 1.

Table 7 - 1 An example of FJSSP fuzzy processing time

Job	Operation	Machine			
		M_1	M_2	M_3	M_4
1	O_{11}	-	(5,6,7)	-	(3,5,6))
	O_{12}	(1,2,3)	(6,8,10)	(5,9,11)	(9,10,14)
	O_{13}	(1,3,4)	-	(4,6,8)	-
2	O_{21}	-	(5,7,9)	(1,2,5)	(1,2,3)
	O_{22}	(5,7,9)	(4,5,7)	(3,5,6)	(1,2,3)
	O_{23}	(7,8,11)	-	-	(1,3,4)
	O_{24}	(7,9,10)	-	(5,6,9)	-
3	O_{31}	(2,4,8)	-	(4,5,7)	-
	O_{32}	(6,8,12)	(5,8,10)	(6,7,10)	(4,7,9)
	O_{33}	(8,9,12)	(3,4,5)	-	-

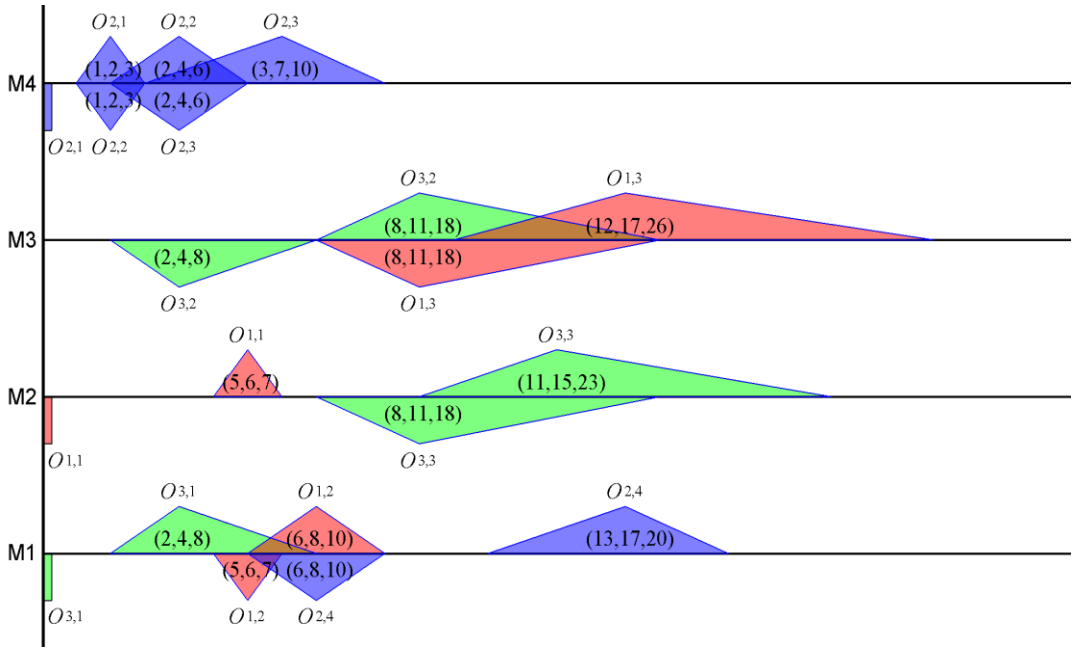

Fig. 7 - 1 Fuzzy Gantt chart by proposed heuristic rule

7.4 Improvising new harmony

In the basic harmony search algorithm, a new harmony is generated based on harmony memory or randomly. In this Chapter, we generate two new harmonies based on two harmonies in harmony memory at the same time. The steps are shown as follows:

Procedure: Improvising two new harmonies x^1_{new} and x^2_{new}

Step1: For each operation $j, 1 \le j \le Toper$. $Toper$ is the total number of operations.

Step2: Generate one random number $R1$. If $R1 < HMCR$, go to Step3; else, go to Step6. x_a and x_b are two harmonies in harmony memory.

Step3: If there is no value in j^{th} position of x^1_{new}, $x^1_{new}(j) = x_a(j)$; else, find the first position (p) place with no value in x^1_{new} and set $x^1_{new}(p) = x_a(j)$. If there is no value in j^{th} position of x^2_{new}, $x^2_{new}(j) = x_b(j)$; else, find the first position (p') with no value in x^2_{new} and set $x^2_{new}(p') = x_b(j)$.

Step4: Generate one random number $R2$. If $R2 < PAR$, go to Step5; else, go to Step7.

Step5: Search the place ($p1$) of $x^1_{new}(j)$ in x_b, exchange $x^1_{new}(j)$ and $x^1_{new}(p1)$; search the place ($p2$) of $x^2_{new}(j)$ in x_a, exchange $x^2_{new}(j)$ and $x^2_{new}(p2)$.

Step6: If the processing machine for $x^1_{new}(j)$ has the minimum processing time, random select a candidate machine for $x^1_{new}(j)$. If the processing machine for $x^2_{new}(j)$ has not the minimum processing time, select the machine with minimum processing time for $x^2_{new}(j)$.

Step7: Repair x^1_{new} and x^2_{new} to make sure that the operations of the same job can satisfy the processing precedence.

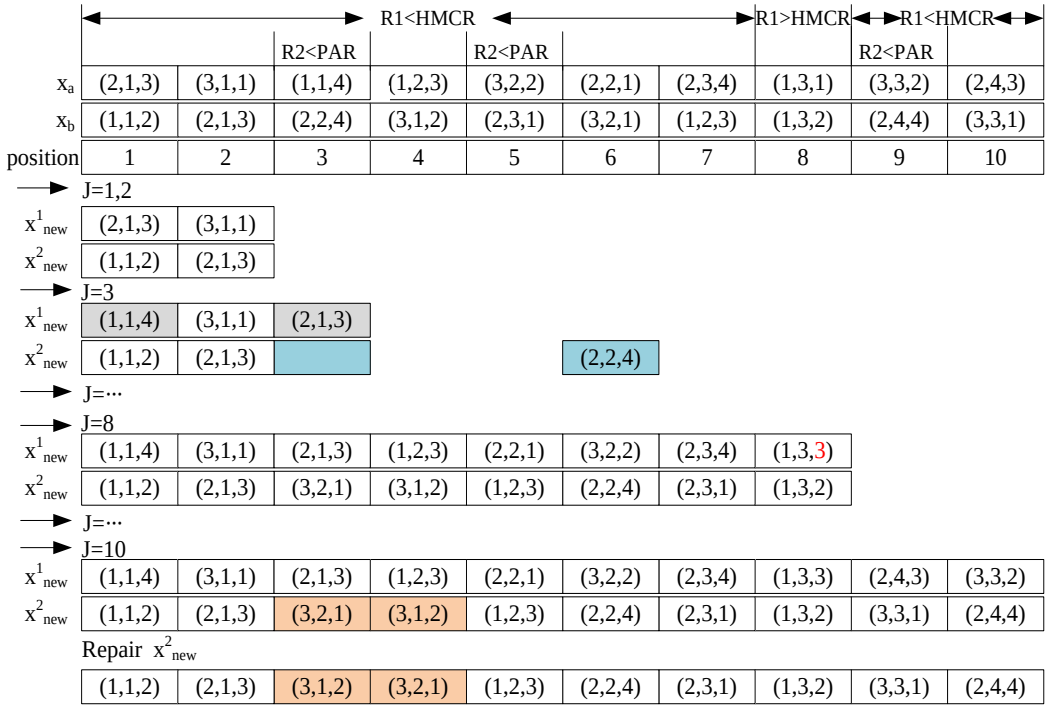

Fig. 7 - 2 Improvising two new harmonies

To represent the steps for improvising new harmony in detail, an example is shown in Fig. 7 - 2. There are 3 jobs, 4 machines and 10 operations in this example. When $j=1$ and $j=2$, the random $R1$ is less than $HMCR$. The operation in $x^1_{new}(j)$ equals that in $x_a(j)$ and the operation in $x^2_{new}(j)$ equals that in $x_b(j)$. When $j=3$, the random number $R2$ is less than PAR. The operation in $x^1_{new}(3)$ equals that $x_a(3)$ and the operation in $x^2_{new}(3)$ equals that in $x_b(3)$. The position of operation ($p1$) of $x^1_{new}(3)$ in x_b is 1. Hence, the values in $x^1_{new}(1)$ and $x^1_{new}(3)$ are changed. The position of operation ($p2$) of $x^2_{new}(3)$ in x_a is 6. Hence, the values of $x^2_{new}(3)$ is put into $x^2_{new}(6)$ and $x^2_{new}(3)$ remains null. The x^1_{new} and x^2_{new} can be obtained in the same way until $j=8$. When $j=8$, the random $R1$ is larger than $HMCR$. The processing machine of operation in $x^1_{new}(8)$ is changed from machine M_1 to M_3. After considering all operations, x^1_{new} and x^2_{new} can be obtained. However, the first and second operations of job 3 in x^2_{new} do not satisfy the processing priority. After changing the positions of these two operations, the two new harmonies x^1_{new} and x^2_{new} are obtained finally.

7.5 Procedure of Proposed FDHS

Based on the above design, the FDHS in this Chapter employs random rule, global minimal processing time rule and the MinEnd heuristic proposed in Section 4.2 to initialize harmony memory or population. The computational procedure of proposed FDHS for FJSSP with fuzzy processing time can be described as follows:

Procedure: Proposed FDHS

Step1: Set parameters, HMS , $HMCR$, PAR .

Step2: Initialize harmony memory HM using MinEnd heuristic, global minimal processing time rule and random rule.

Step3: Evaluate fuzzy completion time of all harmony and update the best fuzzy completion time and the best harmony.

Step4: Improvising new harmonies based on the method proposed in Section 4.3.

Step5: Evaluate fuzzy completion times of all new harmonies and update the best fuzzy completion time and the best harmony.

If the number of iterations is less than the maximum generation, go to Step 4; else, output the best solution.

7.6 Experiment evaluation and comparisons

7.6.1 Experiment setup

To evaluate the performance of proposed FDHS algorithm, experimental evaluation and comparisons are conducted. Two sets of instances are studied in this Chapter. Set one includes five FJSSP cases with fuzzy processing time [82,147]. The sizes are ranging from 10-jobs, 10-machines and 40-operations to 15-jobs, 10-machines and 80-operations. Set two includes eight instances from remanufacturing engineering. The sizes are from 5-jobs, 4-machines and 23-operations to 20-jobs, 15-machines and 355-operations. The proposed FDHS algorithm is coded in C++ and run on Intel 2.40 GHz PC with 2 GB memory. The parameters are set based on our parameter tuning simulations as follows: HMS=150, HMCR=0.95 and PAR=0.5. For instances in set one and the first four instances in Set two, the generation is set to 1000 to meet the same number of evolutions as compared algorithms. For the last four instances in Set two, the generation is set to 4000 because the sizes of these four instances are larger than other instances. All experiments are carried out with 30 replications.

7.6.2 Performance of Proposed MinEnd Heuristic

To test the performance of the proposed MinEnd heuristic, we compare the average fuzzy completion time results of five cases in set one by six heuristic rules, MinEnd, GS, LS, MReW, MReO

and random. The results are shown in Table 7 - 2. Compared to random rule, MReW and MReO rules cannot improve the average fuzzy completion times. GS rule can find better average fuzzy completion times than LS, MReW, MReO and random rules for case 2 and case 3. LS rule can get better fuzzy completion times than GS, MReW, MReO and random rules for case 1, case 4 and case 5. GS and LS have better performance than MReW, MReO and random rules. Compared to GS and LS rules, MinEnd heuristic can obtain better average fuzzy completion time results for all five cases and the improvement is obvious. The CPU times of MinEnd heuristic and all the compared heuristics are very limited and can be ignored. Hence, the proposed MinEnd heuristic rule is effective for initializing harmony memory.

Table 7 - 2 Maximum fuzzy completion time results by six heuristic rules

Instance	Algorithm	Ave Fuzzy completion time			Instance	Algorithm	Ave Fuzzy completion time		
Case1	MinEnd	45.6	61.9	81.0	Case4	MinEnd	34.9	51.9	72.2
	GS	52.0	74.3	96.2		GS	55.9	79.4	105.6
	LS	49.4	71.2	90.9		LS	54.7	78.1	103.1
	MReW	59.3	88.9	117.5		MReW	64.4	87.7	113.4
	MReO	59.7	88.8	117.0		MReO	61.8	85.6	110.6
	Random	59.2	88.0	116.2		Random	71.4	98.5	127.0
Case2	MinEnd	45.6	61.9	81.0	Case5	MinEnd	53.7	78.5	108.2
	GS	80.7	108.3	138.5		GS	80.9	113.5	150.4
	LS	81.5	109.9	140.3		LS	81.2	111.3	146.8
	MReW	98.4	132.6	168.8		MReW	92.1	129.1	170.8
	MReO	102.9	139.3	177.9		MReO	95.5	134.1	177.5
	Random	99.0	134.4	171.6		Random	94.4	132.7	174.7
Case3	MinEnd	46.1	65.9	85.6					
	GS	72.1	97.7	125.1					
	LS	77.3	105.6	135.7					
	MReW	88.3	117.9	151.5					
	MReO	83.3	111.7	143.7					
	Random	85.5	114.1	146.9					

7.6.3 Comparison and discussion

For the five cases in set one, the proposed FDHS algorithm is compared to six existing algorithms: EDA [77], HABC [137], DIGA [147], CGA [55], PEGA [16] and PSO+SA [23]. The results of average fuzzy completion times, best fuzzy completion times and the worst fuzzy completion times are shown in Table 7 - 3.

It can be seen from Table 7 - 3 that the proposed FDHS algorithm performs better than all compared algorithms. For five cases, FDHS algorithm can obtain better average fuzzy completion times and best fuzzy completion times than those by six compared algorithms. In 30 runs, FDHS can find minimum

worst fuzzy completion time for case 2, case 3 and case 4. EDA algorithm finds minimum worst fuzzy completion time for case 1 and case 5. The best solution obtained by FDHS for case 1 is (17, 29, 39) and the fuzzy Gantt chart is illustrated in Fig. 7 - 3.

The average CPU time of all compared algorithms are listed in Table 7 - 4. Compared to the six algorithms, FDHS has the minimum average CPU time. For case 1 to case 4, the average time of FDHS is less than 1.5 seconds. For the largest case, case 5, the average CPU time is just 2.65 seconds. Although the CPU frequencies are different among seven algorithms, we can conclude that the proposed FDHS algorithm is the most efficient one.

To further test the performance of proposed FDHS algorithm, the eight instances in set two are solved. The average, best and worst results by FDHS algorithm and MinEnd heuristic are shown in Table 7 - 5. It can be seen from Table 7 - 5 that FDHS can improve MinEnd obviously for all instances. FDHS can improve the average fuzzy completion time results by MinEnd more than 30% for all instances. Fig. 7 - 4 shows the convergence curve of three ranking criteria results by FDHS. For the same value of the first ranking criterion, the values of the second and the third ranking criteria may become larger or smaller to show the change of results. For the same values of the first and second ranking criteria, the values of the third ranking criterion may become larger or smaller to present results changes. Hence, the first ranking criterion in Fig. 7 - 4 is non-increasing for all instances. The second and third ranking criteria may increase or decrease while the third ranking criterion has larger fluctuations than the second ranking criterion. To show the scheduling results more clearly for eight instances in set two, Fig. 7 - 5 illustrates the fuzzy Gantt chart of the best solution for instance Reman 2.

Table 7 - 3 Seven compared algorithms' results for five cases in set one

Instance	Algorithm	Average value			Best value			Worst value		
Case1	FDHS	**19.9**	**29.6**	**40.6**	**17**	**29**	**39**	19	31	44
	EDA	20.3	30.5	41.6	20	28	40	**22**	**32**	**43**
	HABC	21	32	43.6	19	30	43	23	33	46
	CGA	23.1	33.1	43.4	21	29	41	25	37	47
	DIGA	22.5	32.7	43.3	21	31	40	25	36	48
	PEGA	25	35.1	47.2	23	31	42	29	40	50
	PSO+SA	26.2	36.9	47.7	25	32	40	27	41	54
Case2	FDHS	**32.1**	**46.2**	**57.3**	**30**	**46**	**58**	**35**	**46**	**57**
	EDA	33.7	46.9	57.9	32	46	57	34	48	58
	HABC	33	47.8	62.2	33	46	58	36	48	65
	CGA	35	47.1	60.6	32	47	57	38	49	64
	DIGA	33.4	47.5	62.1	31	47	59	38	50	66
	PEGA	36.9	51	65.9	34	45	60	38	55	72
	PSO+SA	36.7	51.2	65.2	34	45	60	39	54	74
Case3	FDHS	**31.6**	**45.9**	**59.9**	**31**	**45**	**58**	**33**	**48**	**62**

	EDA	32.8	47.2	62.9	31	46	60	34	49	66
	HABC	33.9	50.8	67.3	33	47	64	36	54	70
	CGA	36.4	50.8	66	34	47	63	38	53	71
	DIGA	36.1	51.5	67.5	36	47	64	40	55	73
	PEGA	40.6	56.4	73.3	38	51	66	40	59	77
	PSO+SA	38.6	54.4	70	36	51	65	40	57	75
Case4	FDHS	**24.1**	**36.1**	**50.9**	**24**	**35**	**48**	**26**	**37**	**53**
	EDA	24.8	37.2	51.9	21	36	50	24	39	57
	HABC	25.5	40	56.3	23	38	53	25	44	59
	CGA	27.4	40.4	55	26	37	51	29	42	59
	DIGA	29.6	42.4	56.9	28	40	56	30	46	63
	PEGA	34.3	48.8	65.7	34	46	63	35	50	68
	PSO+SA	33.6	47.9	64.5	32	45	62	34	49	68
Case5	FDHS	**37.9**	**55.8**	**77.8**	**36**	**54**	**74**	42	59	84
	EDA	38.6	56.9	78.3	36	55	73	**40**	**60**	**81**
	CGA	47	65.4	86	42	62	82	49	70	91
	DIGA	45.8	66.3	88.7	42	63	84	49	71	92
	PEGA	50.3	74	96.5	48	68	94	50	74	100
	PSO+SA	51.2	74.6	97.6	48	72	93	52	73	101

Table 7 - 4 Seven algorithms' CPU time (s) for five cases in set one

Algorithm	CPU	Case1	Case2	Case3	Case4	Case5
FDHS	2.40GHz	1.00	1.03	1.30	1.34	2.65
EDA	2.3GHz	3.65	3.63	4.86	4.56	9.83
HABC	2.83GHz	9.87	10.88	14.8	13.85	-
CGA	1.7GHz	8.29	8.26	10.66	10.77	23.87
DIGA	1.7GHz	15.36	15.57	18.87	19.02	37.82
PEGA	1.7GHz	12.56	12.67	15.23	15.71	30.15
PSO+SA	1.7GHz	12.4	12.33	15.24	15.66	30.9

In addition, the average CPU times and generations for eight instances in set two are shown in Table 7 - 6. It can be seen from Table 7 - 6 that average CPU times for 1000 generations are less than 3 seconds for instances Reman 1 to Reman 4. For instances Reman 5 and Reman 6, the average CPU times are more than 20 seconds. The reason is that the instance size becomes larger and the maximum generation increases from 1000 to 4000. The instance sizes of Reman 7 and Reman 8 are 20-jobs, 10-machines, 308-operations and 20-jobs, 15-machines, 355-operations. The average CPU time is 73.83 seconds for Reman 7 and 103.56 seconds for Reman 8.

In a summary, the proposed FDHS algorithm has better performance than six compared algorithms for five cases in set one. Compared to MinEnd heuristic, FDHS can improve the fuzzy completion time results more than 30% for all instances in set two. At the same time, the average CPU times of

FDHS algorithm are less than those of six compared algorithms for 5 cases in set one. For the eight instances in set two, FDHS also has very limited average CPU time.

Table 7 - 5 FDHS and MinEnd results for eight instances in set two

Instance	Algorithm	Average value			Ave impro (%)			Best value			Worst value		
Re 1	FDHS	17.4	26.0	34.7	30.6	33.1	32.9	19	26	33	18	26	36
	MinEnd	25.1	38.9	51.8				18	30	42	37	52	68
Re 2	FDHS	24.7	40.7	56.8	34.0	33.4	33.1	17	39	58	28	43	59
	MinEnd	37.5	61.2	84.9				31	51	73	48	72	96
Re 3	FDHS	36.8	59.1	80.3	43.1	40.7	39.7	37	58	77	43	60	79
	MinEnd	64.8	99.7	133.2				57	89	115	69	110	144
Re 4	FDHS	34.3	51.0	70.2	35.4	37.0	37.1	31	49	70	35	54	75
	MinEnd	53.2	81	111.7				38	67	94	63	104	147
Re 5	FDHS	52.6	84.4	116.1	40.3	39.9	40.0	53	82	112	55	86	121
	MinEnd	88.1	140.6	193.6				82	132	184	103	150	206
Re6	FDHS	44.5	72.6	98.5	37.1	38.8	39.2	44	69	94	52	76	98
	MinEnd	70.8	118.7	162				62	100	139	67	134	186
Re7	FDHS	60.4	107.7	153.3	37.6	35.5	34.3	60	105	149	58	112	159
	MinEnd	96.9	167.2	233.6				94	152	214	100	179	254
Re8	FDHS	57.7	92.3	126.6	32.0	34.5	34.3	56	89	122	57	95	133
	MinEnd	84.9	141.1	192.8				83	132	172	95	153	206

Table 7 - 6 Average CPU time and generation of FDHS for eight instances in set two

Instance	Cpu time (s)	Generation	Instance	Cpu time (s)	Generation
Reman 1	0.40	1000	Reman 5	22.38	4000
Reman 2	1.44	1000	Reman 6	25.97	4000
Reman 3	1.89	1000	Reman 7	73.83	4000
Reman 4	2.68	1000	Reman 8	103.56	4000

Notes:2.40GHz CPU

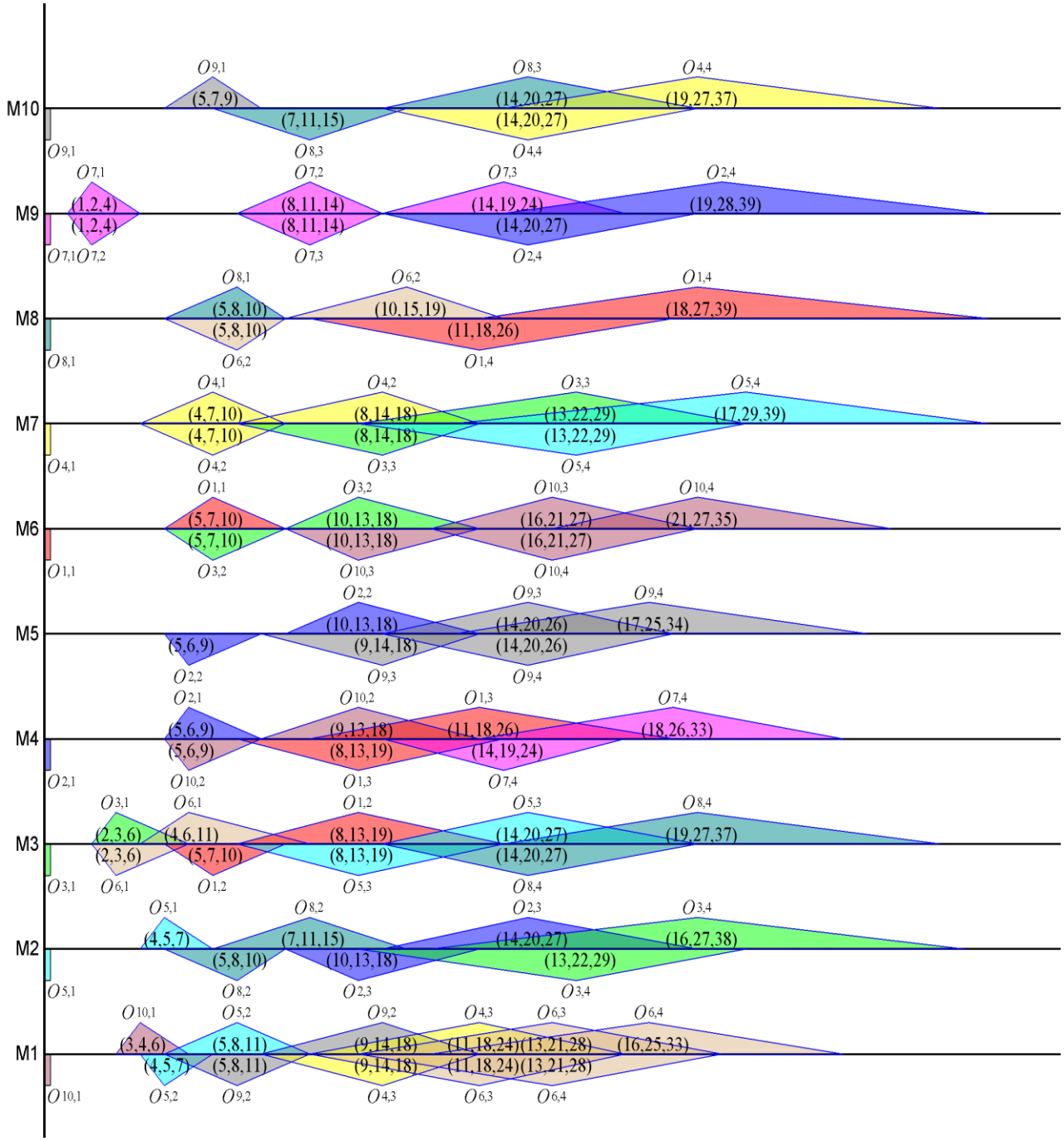

Fig. 7 - 3 Best solution's fuzzy Gantt chart of case 1

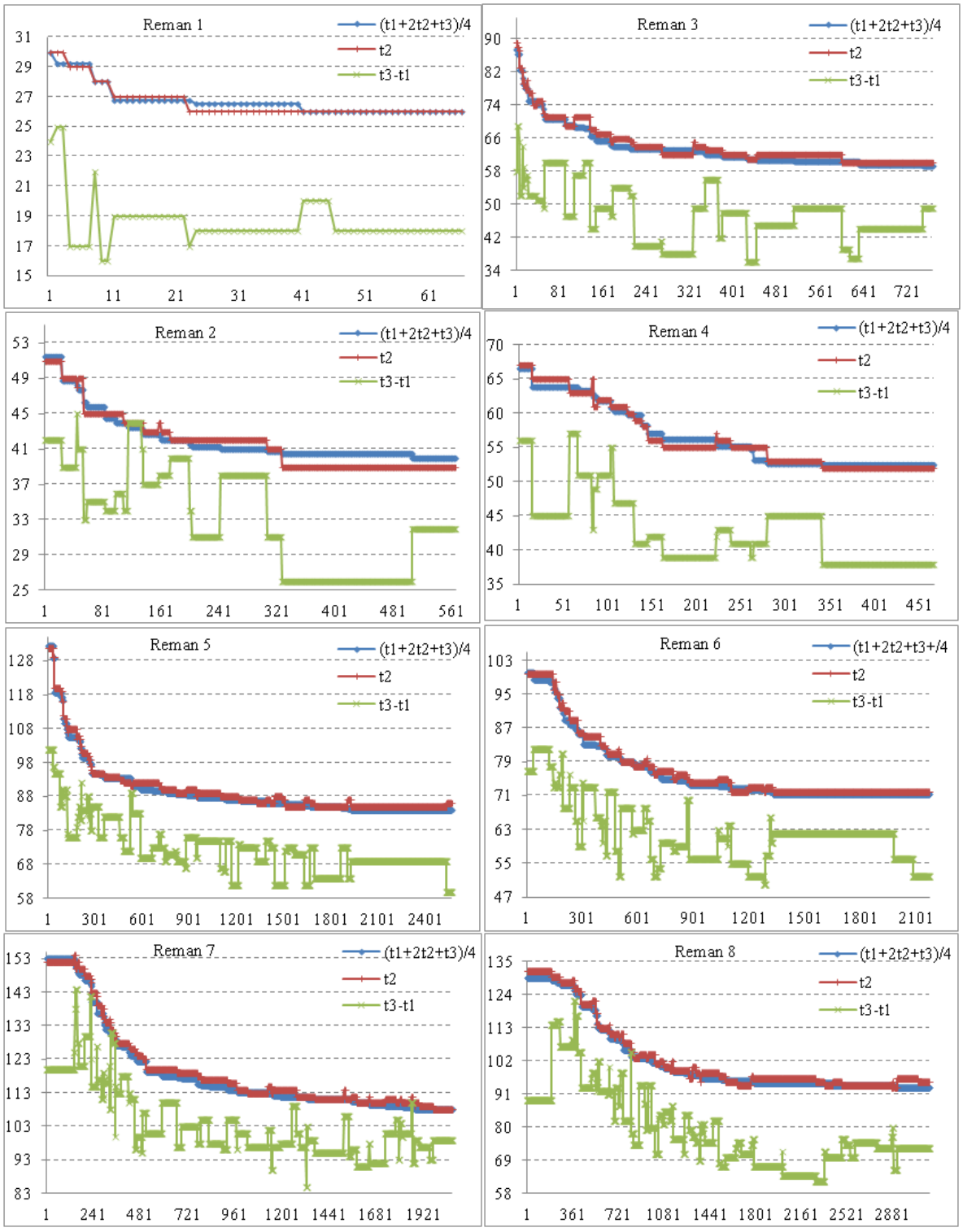

Fig. 7 - 4 Convergence curve of FDHS for eight instances in set two

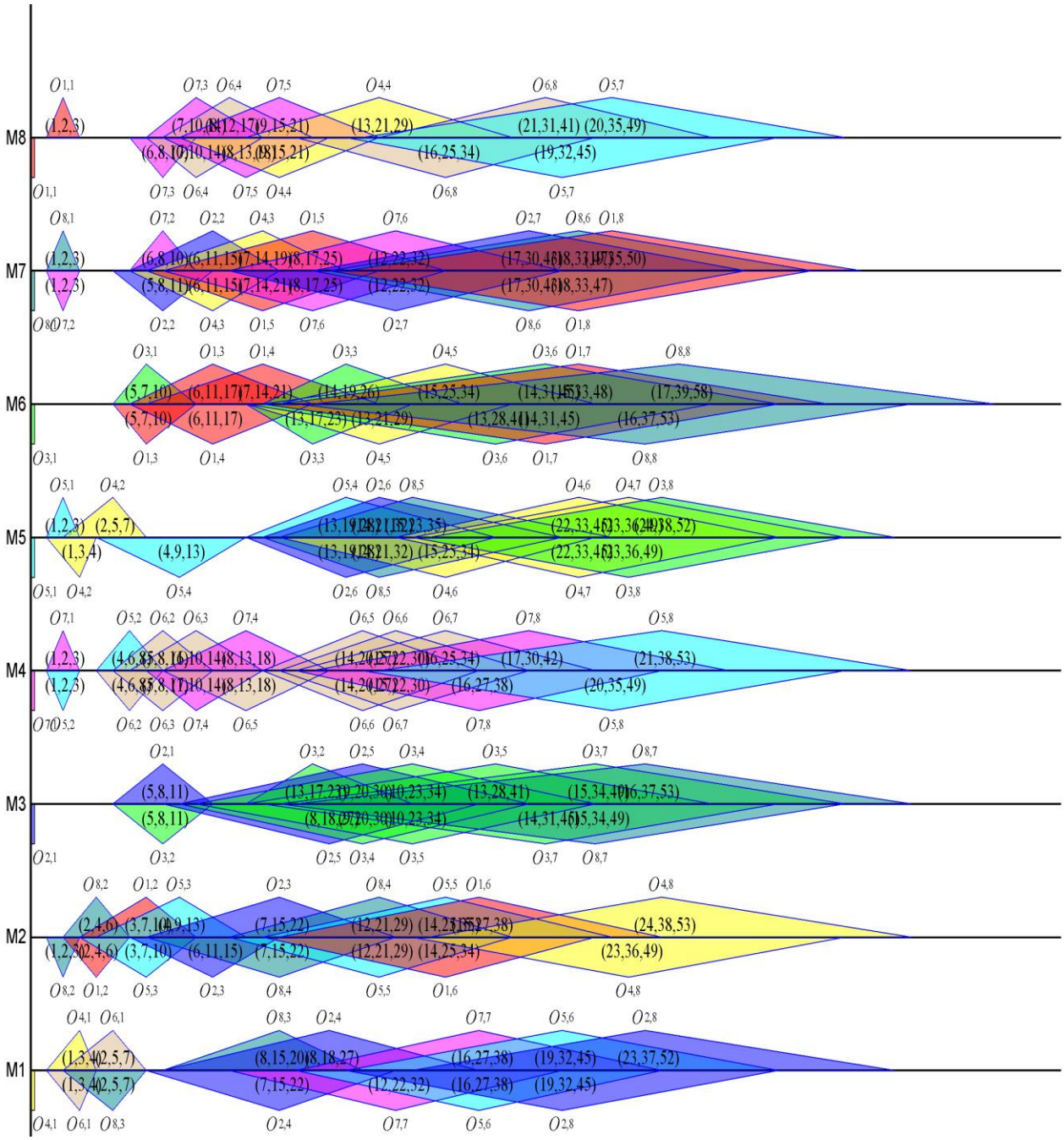

Fig. 7 - 5 Best solution's fuzzy Gantt chart of instance Reman 2

7.7 Conclusions

In this Chapter, a fuzzy discrete harmony search (FDHS) algorithm was proposed for solving flexible job shop scheduling problem with fuzzy processing time. The objective was to minimize the maximum fuzzy completion time. The proposed MinEnd heuristic was evaluated by comparing against the random rule and four existing heuristics. The FDHS compared against six existing algorithms for solving five benchmark cases. The results and comparisons showed superior performance of FDHS. As an example, the fuzzy Gantt chart of one case's new best solution was given. In addition, eight instances from remanufacturing were solved by MinEnd heuristic and FDHS algorithm. FDHS improved results by MinEnd obviously. The computer CPU time of FDHS algorithm was very low. The experiments also showed the convergence performance of FDHS algorithm. Fuzzy Gantt chart of one remanufacturing instance was shown as an example.

Chapter 8

FABC Algorithm for Multi-objective FJSSP with Fuzzy Processing Time

8.1 Introduction

This Chapter addresses multi-objective flexible job shop scheduling problem (FJSSP) with fuzzy processing time. The uncertainty in the processing time is one of the seven characteristics in remanufacturing. An effective fuzzy artificial bee colony (FABC) algorithm is proposed for FJSSP with fuzzy processing time. The objectives are to minimize the maximum fuzzy completion time and the maximum fuzzy machine workload. The three-value element encoding and decoding method is used to show the solution in FABC algorithm. The MinEnd heuristic proposed in Chapter 7 is employed to initialize population. Extensive computational experiments are carried out using five benchmark cases with eight instances from remanufacturing. The proposed heuristic rule is evaluated using five benchmark cases for minimizing the maximum fuzzy completion time and the maximum fuzzy machine workload objectives. FABC algorithm is compared to FDHS algorithm and six meta-heuristics for maximum fuzzy completion time criterion. For maximum fuzzy machine workload objective, FABC algorithm is compared to FDHS algorithm and six heuristics. The results and comparisons demonstrate improved performance of FABC algorithm for solving multi-objective flexible job shop scheduling problem with fuzzy processing time.

The rest of this Chapter is organized as follows: Section 8.2 briefly describes the two objectives of FJSSP with fuzzy processing time constraint. The proposed FABC algorithm is represented in Section 8.3. Section 8.4 is devoted to experimental results and comparisons. We conclude this Chapter in Section 8.5.

8.2 FJSSP with fuzzy processing time

The FJSSP with fuzzy processing time means that the operation processing time is not an exact value and the processing time is shown as a triangular fuzzy number (TFN) as follows:

$$t_{i,j,k} = (t^1_{i,j,k}, t^2_{i,j,k}, t^3_{i,j,k}) \tag{8-1}$$

where $t^1_{i,j,k}$, $t^2_{i,j,k}$ and $t^3_{i,j,k}$ are three probable processing time of operation $O_{i,j}$ on machine M_k.

The fuzzy completion time of operation $O_{i,j}$ is a TFN as follows:

$$C_{i,j} = (C^1_{i,j}, C^2_{i,j}, C^3_{i,j}) \tag{8-2}$$

where $C^1_{i,j}$, $C^2_{i,j}$ and $C^3_{i,j}$ are three probable completion time of operation $O_{i,j}$.

The fuzzy machine workload of machine M_k is also a TFN as follows:

$$w_k = (w^1_k, w^2_k, w^3_k) \tag{8-3}$$

w^1_k, w^2_k and w^3_k are three probable machine workload of machine M_k.

The objective is to minimize maximum fuzzy completion time and maximum fuzzy machine workload.

The maximum fuzzy completion time denoted by C_M can be calculated by the following formula:

$$\text{Min } C_M = \max_{1 \le i \le n} \{C_i\} \tag{8-4}$$

where C_i is the fuzzy completion time of job J_i.

The maximum fuzzy machine workload denoted by W_M, can be calculated by:

$$\text{Min } W_M = \max_{1 \le k \le m} \{W_k\} \tag{8-5}$$

where W_k is the workload of machine M_k.

8.3 FABC for FJSSP with fuzzy processing time

8.3.1 Initializing

The quality of initial population often affects the convergence of meta-heuristic for FJSSP. It is important to generate a good quality initial population. Many heuristic rules are proposed for machine assignment and operation sequencing in FJSSP, such as, local minimizing processing time (LS) rule, global minimizing processing time rule (GS), most work remaining rule (MReW) and most number of operations remaining (MReO) rule [16]. In this Chapter, we develop MinEnd heuristic to initialize

population. In this rule, the operation sequence is generated randomly. The processing machine is assigned based on the operation order in operation sequence. For each operation, the machine with the minimum fuzzy completion time will be selected to process this operation.

8.3.2 Employed bee phase

In this phase, employed bee X_i generates a new food source X_{new}. If the fitness of X_{new} is equal to or better than that of X_i, X_{new} will replace X_i as a new food source; otherwise, X_i is retained. To adapt FJSSP with fuzzy processing time and encoding rule, the procedure for new food source is shown as follows.

Procedure: Generating new food source

If (r1<P1) //r1 is a random in [0,1], P1 is the probability to generate new food source

Select two employed bee X_p and X_q, $X_p \neq X_q \neq X_i$.

If (r2<P2) //r2 is a random in [0, 1], P2 is the probability to select a better food source from X_p and X_q

If the fitness of X_p is better than that of X_q, $X_j = X_p$; else $X_j = X_q$.

Else

If the fitness of X_p is worse than that of X_q, $X_j = X_p$; else $X_j = X_q$.

For k=1 to toper //toper is the total operation number

If (r3<P3) // r3 is a random in [0,1], P3 is the probability to select element from X_j

$$X_{new}^k = X_j^k$$

Else

$$X_{new}^k = X_i^k$$

Repair X_{new} to make sure the operations of the same job can satisfy the processing precedence.

Else

For k=1 to toper //toper is the total operation number

If (r4<P4) // r4 is a random in [0,1], P4 is the probability for the minimum processing time machine

If current machine of X_i^k has minimum processing time,

Select another machine randomly for X_i^k

Else

Select minimum processing time machine for X_i^k

Else

Select two operations X_i^a and X_i^b, $a < b$.

If X_i^a and X_i^b are different jobs' operations

Insert X_i^a at X_i^b position.

Repair X_{new} to make sure the operations of the same job can satisfy the processing precedence.

End

One employed bee can generate a new food source by selecting operations from different bees or by changing assigned machines. The complexity by operation sequence is $O(log_2 \sum_{i=1}^{n} q_i * log_2 \sum_{i=1}^{n} q_i)$ while the complexity by changing machine is $O((\sum_{i=1}^{n} q_i)$. The computational to repair the new food source is $O((\sum_{i=1}^{n} q_i * log_2 \sum_{i=1}^{n} q_i)$.

8.3.3 Onlooker bee phase

In onlooker bee phase, the fitness of all the employed bees is evaluated. A food source X_i is selected depending on its probability value p_i calculated by expression (3-8) shown in Chapter 3. The procedure to select a food source is shown as follows.

Procedure: Select a food source

Select two employed bee X_p and X_q, $X_p \neq X_q$.

If (r2<P2)

If the fitness of X_p is better than that of X_q, $X_i = X_p$; else, $X_i = X_q$.

Else

If the fitness of X_p is worse than that of X_q, $X_i = X_p$; else, $X_i = X_q$.

End

8.3.4 Scout bee phase

If a food source X_i cannot be improved through a predetermined number of trials $Limit$, the food source is to be abandoned and the corresponding employed bee becomes a scout. A new food source will be generated randomly.

8.3.5 Procedure of FABC algorithm

In the FABC algorithm, the proposed MinEnd heuristic, GS heuristic rule, and random rule are used to initial population. In this Chapter, 60% solutions in population are generated randomly, 20% solutions by MinEnd heuristic and 20% solutions by GS heuristic rule. In the procedure to generate new food source, changing processing machine operator and operation inserting operator are employed for exploitation search. Better or worse food source selection and scout bee operator are used for exploration search. Based on the special design above, the procedure of FABC algorithm is as follows.

Procedure: FABC algorithm

Step1: Set parameters, employed bee number, onlooker bee number, scout bee number, probability P1, P2, P3 and P4.

Step2: Initialize population using MinEnd heuristic, global minimizing processing time rule and random rule.

Step3: Perform employed bee phase.

Step4: Perform onlooker bee phase.

Step5: Update the best solution.

Step6: If $Limit$ is met, perform scout bee phase

Step7: If the stop criterion is not satisfied, go to Step 3; else, output the best solution.

8.4 Experiment evaluation and comparisons

8.4.1 Experiment setup

To evaluate the performance of proposed FABC algorithm for minimizing the maximum fuzzy completion time and the maximum fuzzy machine workload criteria, experimental evaluation and comparisons are conducted. Two sets of instances are evaluated in this Chapter. Set one consist five FJSSP with fuzzy processing time cases [82, 147]. The sizes are ranging from 10-jobs, 10-machines and 40-operations to 15-jobs, 10-machines and 80-operations. Set two includes eight instances from remanufacturing industry. The size is from 5-jobs, 4-machines and 23-operations to 20-jobs, 15-machines and 355-operations. The proposed FABC algorithm is coded in C++ and run on Intel 2.40 GHz PC with 2 GB memory. The parameters are set based on our previous research and the compared algorithms as follows: employed bee 50; onlooker bee 100, scout bee 20, $Limit$ 50, P1=0.8, P2=0.95, P3=0.2 and P4=0.5. For instances in set one and the previous four instances in set two, the generation is set to 1000. For the last four instances in set two, the generation is set to 4000 because the problem size of these four instances is larger than that of other instances. All experiments are carried out with 30 replications.

8.4.2 Testing MinEnd heuristic

To test MinEnd heuristic performance, five cases in set one are evaluated by MinEnd heuristic, GS, LS, MReW, MReO and random six heuristic rules. The maximum fuzzy completion time results are

shown in Table 8 - 1 while the maximum fuzzy machine workload results are shown in Table 8 - 2. For the two objectives, the best values, the average of the expected values and worst values in 30 runs are counted.

Compared to random rule, MReW and MReO rules cannot improve maximum fuzzy completion time and maximum fuzzy machine workload results. MinEnd, GS and LS heuristic rules can get higher quality results than MReW, MReO and random rules. Among the six heuristics, MinEnd can find the best maximum fuzzy completion time results for all five cases except the best value of case1. MinEnd heuristic gets the best value (42, 53, 69) for case 1 while LS heuristic rule obtains the best value (22, 46, 58). GS rule can find the maximum fuzzy completion time results second only to MinEnd heuristic. For maximum fuzzy machine workload objective, GS heuristic rule obtains the best results for all five cases except the best value of case 3. GS finds the best value (32, 44, 55) for case 3 while MinEnd heuristic finds the best value (27, 42, 56). MinEnd heuristic can get the maximum fuzzy machine workload results second only to the GS rule. To show the best results clearly, the best results data in Table 8 - 1 and Table 8 - 2 are in bold. Considering the two objectives, MinEnd and GS are two effective heuristic rules for FJSSP with fuzzy processing time.

Table 8 - 1 Maximum fuzzy completion time of five cases by six heuristics

Instance	Algorithm	Average value			Best value			Worst value		
Case1	MinEnd	**45.6**	**61.9**	**81.0**	42	53	69	**51**	**73**	**94**
	GS	52.0	74.3	96.2	38	53	67	76	113	150
	LS	49.4	71.2	90.9	**22**	**46**	**58**	63	92	125
	MReW	59.3	88.9	117.5	49	75	102	75	109	142
	MReO	59.7	88.8	117.0	54	79	103	75	110	143
	Random	59.2	88.0	116.2	49	75	102	69	102	138
Case2	MinEnd	**45.6**	**61.9**	**81.0**	**42**	**53**	**69**	**51**	**73**	**94**
	GS	80.7	108.3	138.5	50	69	90	108	147	189
	LS	81.5	109.9	140.3	57	77	97	115	154	196
	MReW	98.4	132.6	168.8	89	118	151	114	152	195
	MReO	102.9	139.3	177.9	89	118	151	127	169	216
	Random	99.0	134.4	171.6	89	118	151	131	178	229
Case3	MinEnd	**46.1**	**65.9**	**85.6**	**40**	**57**	**72**	**54**	**79**	**101**
	GS	72.1	97.7	125.1	47	65	87	90	123	156
	LS	77.3	105.6	135.7	58	76	97	98	132	171
	MReW	88.3	117.9	151.5	68	91	117	109	142	185
	MReO	83.3	111.7	143.7	72	96	122	128	167	217
	Random	85.5	114.1	146.9	71	97	126	116	151	193
Case4	MinEnd	**34.9**	**51.9**	**72.2**	**28**	**45**	**62**	**39**	**59**	**88**
	GS	55.9	79.4	105.6	36	60	81	81	118	158
	LS	54.7	78.1	103.1	39	57	74	78	108	141

	MReW	64.4	87.7	113.4	54	70	90	86	126	163
	MReO	61.8	85.6	110.6	52	70	90	70	100	129
	Random	67.7	93.7	121.3	48	70	90	78	110	141
Case5	MinEnd	**53.7**	**78.5**	**108.2**	**45**	**68**	**94**	**61**	**93**	**126**
	GS	80.9	113.5	150.4	53	83	113	103	142	194
	LS	81.2	111.3	146.8	90	73	100	114	158	204
	MReW	92.1	129.1	170.8	76	111	147	104	144	188
	MReO	95.5	134.1	177.5	85	114	157	113	155	207
	Random	94.4	132.7	174.7	83	114	149	134	190	247

Table 8 - 2 Maximum fuzzy machine workload of five cases by six heuristics

Ins.e	Algorithm	Average value			Best value			Worst value		
Case1	MinEnd	23.0	34.2	45.9	21	30	40	26	41	54
	GS	**20.9**	**30.6**	**40.9**	**19**	**27**	**40**	**26**	**33**	**42**
	LS	33.0	50.0	64.0	33	50	64	33	50	64
	MReW	45.0	66.0	85.4	30	44	57	66	92	114
	MReO	42.3	62.6	81.3	27	41	55	65	102	132
	Random	41.0	60.1	78.3	27	39	51	60	81	106
Case2	MinEnd	36.6	50.2	65.1	31	42	53	46	61	80
	GS	**32.1**	**44.5**	**57.4**	**28**	**40**	**50**	**33**	**47**	**65**
	LS	54.0	75.0	96.0	54	75	96	54	75	96
	MReW	61.3	83.8	108.6	41	59	75	94	130	167
	MReO	69.2	94.6	122.0	44	56	77	109	147	188
	Random	65.0	88.9	113.9	42	60	78	93	129	165
Case3	MinEnd	36.4	51.4	66.6	**27**	**42**	**56**	40	61	78
	GS	**32.1**	**45.0**	**59.4**	32	44	55	**34**	**49**	**66**
	LS	49.0	72.0	89.0	49	72	89	49	72	89
	MReW	63.5	88.9	115.2	48	66	85	93	130	164
	MReO	67.1	94.5	123.2	39	58	77	95	131	172
	Random	61.9	86.7	112.3	41	62	83	90	122	161
Case4	MinEnd	27.7	39.7	55.1	25	35	45	35	46	61
	GS	**23.9**	**34.9**	**48.4**	**23**	**33**	**45**	**28**	**38**	**52**
	LS	25.0	39.0	57.0	25	39	57	25	39	57
	MReW	46.9	66.6	91.6	34	48	69	69	97	138
	MReO	45.0	65.3	90.6	28	42	60	71	98	133
	Random	41.8	60.2	82.5	30	42	61	60	83	110
Case5	MinEnd	45.2	64.9	88.7	42	60	80	50	73	97
	GS	**41.1**	**59.4**	**80.0**	**38**	**57**	**80**	**45**	**64**	**80**
	LS	52.0	76.0	109.0	52	76	109	52	76	109
	MReW	80.9	114.5	154.0	65	88	118	110	155	206
	MReO	80.9	113.2	151.4	55	86	119	105	141	189
	Random	83.4	118.3	158.5	60	85	114	145	201	268

8.4.3 Maximum fuzzy completion time objective

For the five cases in set one, FABC algorithm is compared to six existing algorithms, EDA [77], HABC [137], CGA [82], DIGA [147], PEGA [16] and PSO+SA [23]. The results of average, best and worst values in 30 runs are shown in Table 8 - 3. It can be seen from Table 8 - 3 that the proposed FABC algorithm performs better than six compared algorithms. For five cases, FABC algorithm can obtain better average values and best values than the six compared algorithms. In 30 runs, FABC can find the minimum worst values for case 1, case 2 case 3 and case 4. EDA algorithm finds minimum worst value for case 5. The best solution obtained by FABC for case 1 is (19, 28, 39) and the fuzzy Gantt chart is illustrated in Fig. 8 - 1. The average CPU times of all the compared algorithms are listed in Table 8 - 4. Compared to the six existing algorithms, FABC has the minimum average CPU times. For case 1 to case 4, the average time of FABC is less than 1.5 seconds. For the largest case, case 5, the average CPU time is just 2.41 seconds. Although the CPU frequencies are different among seven algorithms, we can conclude that the proposed FABC algorithm is the most efficient one.

The FABC algorithm is also compared to the FDHS algorithm proposed in Chapter 7. It can be seen from Table 8 - 3 that the FABC algorithm improves some measures' results of FDHS algorithm for five cases. FDHS algorithm gets better average and best results for case 2 than those by FABC algorithm. FABC algorithm obtains competitive results for all other cases and comparison measures except the worst measure of case 4. In addition, the FABC and FDHS algorithms have the same average CPU time for case 1. FABC has smaller average CPU times than FDHS for case 2 and case 5. For case 3 and case 4, FABC has larger average CPU times than FDHS algorithm. These differences between FDHS and FABC are not very obvious.

Literature [149] and [150] proposed SNSA and HGTS algorithms for FJSSP with fuzzy processing time. Except the five cases above, SNSA and HGTS solved case 6 from literature [149]. In literature [150], HGTS solved the expected makespan and the results are compared to those by SNSA. Among all compared algorithms in [150], only SNSA and HGTS solved case 6 for excepted makespan. Hence, FDHS and FABC are compared to SNSA and HGTS for case 6 with expected makespan objective. SNSA, HGTS, FDHS and FABC run 30 repeats for case 6. The minimum results and average results of these four algorithms are shown in Table 8 -3. To compare with SNSA and HGTS, the format of the results for case 6 is based on the ranking operation of triangle fuzzy number in Section 3.3.2 of Chapter 3. The average computation times are shown in Table 8 -4. Compared to SNSA, FDHS and FABC obtained better minimum results and average results for the sixth case. SNSA is run on a 2GB RAM 2.2 GHz computer while FDHS and FABC are run on a 2.4GHz computer. FDHS and FABC have smaller execution time than SNSA for case 6. Compared to HGTS, FDHS and FABC do not obtained better than HGTS. However, the differences among the three results are not obvious. HGTS needs much more computer CPU time than FDHS and FABC even the running times are not directly

comparable due to difference in target machines. The running time of FDHS and FABC are 2.46 to 2.48 seconds while the running time of HGTS is larger than 50 seconds. It means that FDHS and FABC can obtain solutions that are very near to optimal solutions in very short CPU time.

Table 8 - 3 Maximum fuzzy completion time by eight algorithms for five cases

Instance	Algorithm	Average value			Best value			Worst value		
Case1	FABC	**20.1**	**29.4**	**40.3**	**19**	**28**	**39**	**22**	**30**	**42**
	FDHS	19.9	29.6	40.6	17	29	39	19	31	44
	EDA	20.3	30.5	41.6	20	28	40	22	32	43
	HABC	21	32	43.6	19	30	43	23	33	46
	CGA	23.1	33.1	43.4	21	29	41	25	37	47
	DIGA	22.5	32.7	43.3	21	31	40	25	36	48
	PEGA	25	35.1	47.2	23	31	42	29	40	50
	PSO+SA	26.2	36.9	47.7	25	32	40	27	41	54
Case2	FABC	**32.3**	**46.2**	**57.3**	**33**	**45**	**58**	**35**	**46**	**57**
	FDHS	**32.1**	**46.2**	**57.3**	**30**	**46**	**58**	35	46	57
	EDA	33.7	46.9	57.9	32	46	57	34	48	58
	HABC	33	47.8	62.2	33	46	58	36	48	65
	CGA	35	47.1	60.6	32	47	57	38	49	64
	DIGA	33.4	47.5	62.1	31	47	59	38	50	66
	PEGA	36.9	51	65.9	34	45	60	38	55	72
	PSO+SA	36.7	51.2	65.2	34	45	60	39	54	74
Case3	FABC	**31.8**	**45.8**	**59.6**	**31**	**45**	**57**	**33**	**47**	**63**
	FDHS	31.6	45.9	59.9	31	45	58	33	48	62
	EDA	32.8	47.2	62.9	31	46	60	34	49	66
	HABC	33.9	50.8	67.3	33	47	64	36	54	70
	CGA	36.4	50.8	66	34	47	63	38	53	71
	DIGA	36.1	51.5	67.5	36	47	64	40	55	73
	PEGA	40.6	56.4	73.3	38	51	66	40	59	77
	PSO+SA	38.6	54.4	70	36	51	65	40	57	75
Case4	FABC	**24.1**	**36.1**	**50.9**	**25**	**34**	**49**	**24**	**38**	**55**
	FDHS	24.1	36.1	50.9	24	35	48	**26**	**37**	**53**
	EDA	24.8	37.2	51.9	21	36	50	24	39	57
	HABC	25.5	40	56.3	23	38	53	25	44	59
	CGA	27.4	40.4	55	26	37	51	29	42	59
	DIGA	29.6	42.4	56.9	28	40	56	30	46	63
	PEGA	34.3	48.8	65.7	34	46	63	35	50	68
	PSO+SA	33.6	47.9	64.5	32	45	62	34	49	68
Case5	FABC	**37.8**	**55.8**	**77.7**	**36**	**54**	**74**	42	59	84
	FDHS	37.9	55.8	77.8	36	54	74	42	59	84
	EDA	38.6	56.9	78.3	36	55	73	**40**	**60**	**81**
	HABC	-	-	-	-	-	-	-	-	-
	CGA	47	65.4	86	42	62	82	49	70	91
	DIGA	45.8	66.3	88.7	42	63	84	49	71	92

	PEGA	50.3	74	96.5	48	68	94	50	74	100
	PSO+SA	51.2	74.6	97.6	48	72	93	52	73	101
Case6	FABC		52.50			55.83			-	
	FDHS		52.40			55.94			-	
	HGTS		50.25			51.50			-	
	SNSA		63.75			65.65				

Table 8 - 4 CPU time of eight algorithms for five cases

Algorithm	Case1	Case2	Case3	Case4	Case5	Case6
FABC [a]	1.00	1.00	1.35	1.37	2.41	2.46
FDHS [a]	1.00	1.03	1.30	1.34	2.65	2.48
EDA[b]	3.65	3.63	4.86	4.56	9.83	-
HABC[c]	9.87	10.88	14.8	13.85	-	-
CGA[d]	8.29	8.26	10.66	10.77	23.87	-
DIGA[d]	15.36	15.57	18.87	19.02	37.82	-
PEGA[d]	12.56	12.67	15.23	15.71	30.15	-
PSO+Sa[d]	12.4	12.33	15.24	15.66	30.9	-
HGTS [e]	-	-	-	-	-	53.30
SNSA [f]	-	-	-	-	-	14.35

Notes: [a] 2.40GHz CPU, [b] 2.3GHz CPU, [c] 2.83GHz CPU, [d] 1.7GHz CPU, [e] XeE5520 processor, [f] 2.2GHz CPU

To further test FABC algorithm, the eight instances in set two are solved. The average, best and worst values by FABC algorithm and six heuristics are shown in Table 8 - 5. It can be seen from Table 8 - 5 that MinEnd heuristic obtains better results than other five heuristics. FABC can improve MinEnd obviously for all instances. FABC can improve the average fuzzy completion time results by MinEnd more than 30% for all instances. To show the scheduling results more clearly for eight instances in set two, Fig. 8 - 2 illustrates the best solution (26, 38, 54) in fuzzy Gantt chart of instance Reman 2 for maximum fuzzy completion time objective. In addition, the average CPU times and generation for eight instances in set two are shown in Table 8 - 6. It can be seen from Table 8 - 6 that average CPU times for 1000 generations are less than 3.5 seconds for instance Reman 1 to Reman 4. For instance Reman 5 and Reman 6, the average CPU times are more than 30 seconds. The reason is that the instances become larger and the generation increases from 1000 to 4000. The instance sizes of Reman 7 and Reman 8 are 20-jobs, 10-machines, 308-operations and 20-jobs, 15-machines, 355-operations. The average CPU times are 82.99 seconds for Reman 7 and 102.77 seconds for Reman 8. Considering the results and average CPU times, FABC algorithm is effective for eight remanufacturing instances with maximum fuzzy completion time objective.

FABC algorithm is also compared to FDHS algorithm in Chapter 7 for solving eight instances from remanufacturing industry. FABC algorithm improves some measures' results by FDHS algorithm. For example, FABC obtains better results than those by FDHS for average and worst measures of Reman

2. For Reman 4, FABC algorithm improves the average and best results by FDHS algorithm. FABC algorithm also improves the best, worst and best results by FDHS algorithm for Reman 5, Reman 6 and Reman 8, respectively. For Reman 1 to Reman 7, the average CPU times of FABC algorithm are larger than those of FDHS algorithm. FABC algorithm has smaller average CPU time than that of FDHS algorithm for Reman 8. However, the differences of average CPU times are not very obvious.

To show the convergence of FABC algorithm, Fig. 8 - 3 shows the convergence curves of three ranking criteria results by FABC for case 5 and Reman 8. For the same value of the first ranking criterion, the values of the second and the third ranking criteria may become larger or smaller to show the change of results. For the same values of the first and second ranking criteria, the third ranking criterion may become larger or smaller to present results changes. Hence, the first ranking criterion is non-increasing for two instances. The second and third ranking criteria may increase or decrease while the third ranking criterion has larger fluctuations than the second one. In summary, FABC algorithm is effective for fuzzy maximum completion time objective of FJSSP with fuzzy processing time.

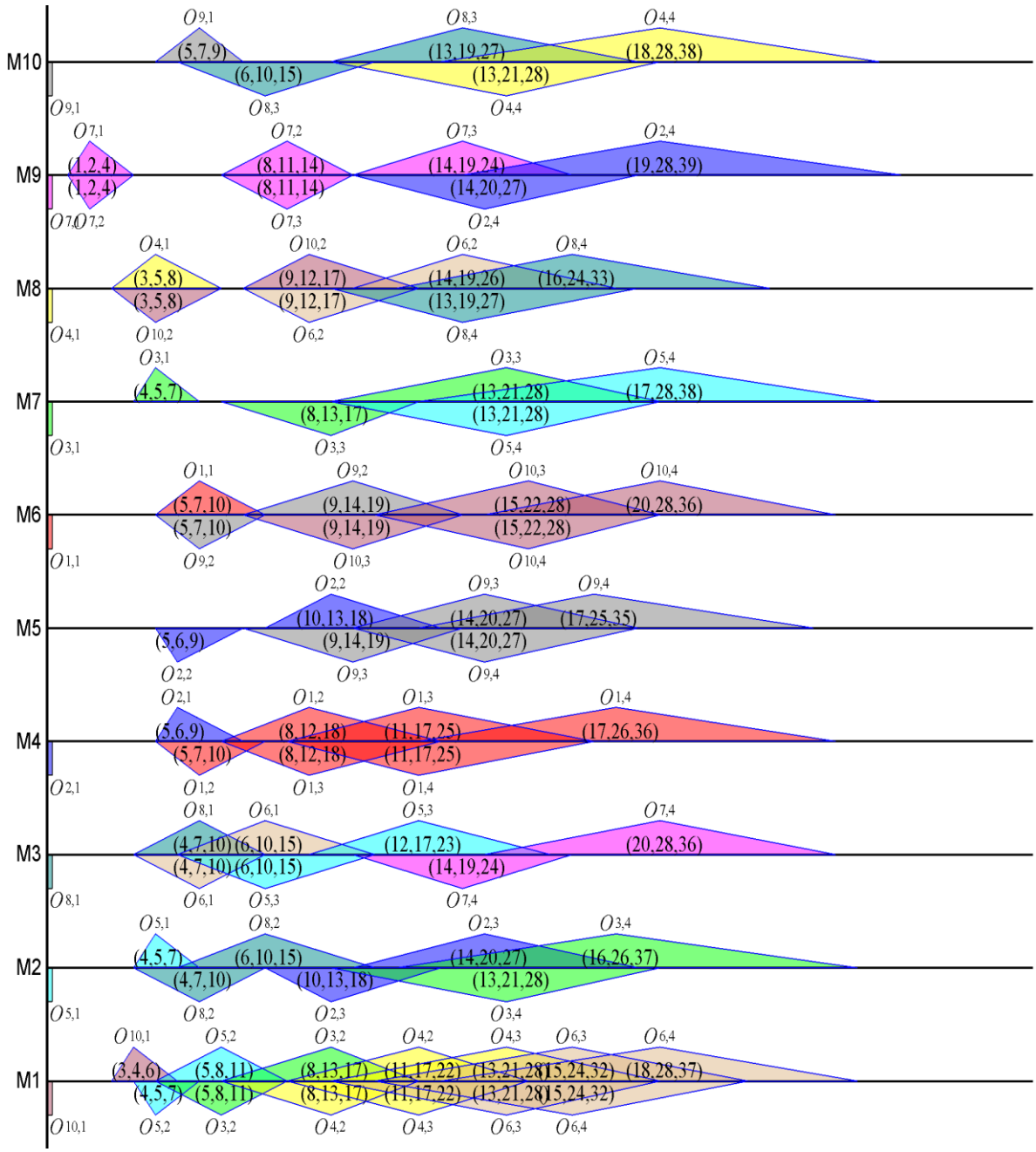

Fig. 8 - 1 Best maximum fuzzy completion time Gantt chart of case 1 by FABC

Table 8 - 5 Maximum fuzzy completion time for eight Reman instances

Instance	Algorithm	Average value			Best value			Worst value		
Reman 1	FABC	**17.6**	**26.0**	**34.6**	**19**	**26**	**33**	**19**	**26**	**36**
	FDHS	17.4	26.0	34.7	19	26	33	18	26	36
	MinEnd	25.1	38.9	51.8	18	30	42	37	52	68

	GS	31.7	47.5	63.2	25	36	48	41	61	81
	LS	29.0	46.6	61.2	17	34	45	40	61	79
	MReW	29.6	47.3	64.7	20	33	45	58	80	107
	MReO	28.9	41.8	55.9	16	30	42	57	76	96
	Random	33.4	50.9	67.8	14	30	42	74	96	124
Reman 2	FABC	**24.9**	**40.6**	**56.7**	**26**	**38**	**54**	**26**	**43**	**60**
	FDHS	24.7	40.7	56.8	17	39	58	28	43	59
	MinEnd	37.5	61.2	84.9	31	51	73	48	72	96
	GS	53.6	83.8	116.3	44	66	91	73	116	160
	LS	59.3	95.2	134.3	47	78	108	69	109	153
	MReW	69.1	106.7	147.2	48	75	107	119	167	228
	MReO	49.9	75.5	104.8	37	59	80	77	100	140
	Random	71.6	108.6	148.7	39	65	97	122	169	223
Reman 3	FABC	**37.6**	**59.6**	**80.8**	**37**	**59**	**76**	**46**	**61**	**83**
	FDHS	36.8	59.1	80.3	37	58	77	43	60	79
	MinEnd	64.8	99.7	133.2	57	89	115	69	110	144
	GS	76.7	117.7	159.5	56	91	120	91	133	179
	LS	82.9	128.7	173.9	67	101	135	108	160	210
	MReW	94.9	144.6	194.0	61	102	138	142	198	262
	MReO	69.7	104.4	137.2	47	83	116	115	146	185
	Random	93.1	142.8	191.3	61	99	132	165	223	289
Reman 4	FABC	**33.6**	**50.9**	**70.3**	**34**	**49**	**67**	**35**	**54**	**75**
	FDHS	34.3	51.0	70.2	31	49	70	35	54	75
	MinEnd	53.2	81.0	111.7	38	67	94	63	104	147
	GS	67.2	103.9	141.7	58	91	126	94	131	176
	LS	74.0	118.9	166.4	55	101	144	91	142	198
	MReW	91.0	140.0	192.6	56	102	143	146	199	275
	MReO	64.4	95.0	129.6	48	77	106	112	147	188
	Random	92.7	141.5	193.5	60	100	139	160	222	301
Reman 5	FABC	**52.3**	**84.5**	**116.3**	**44**	**82**	**119**	**59**	**88**	**116**
	FDHS	52.6	84.4	116.1	53	82	112	55	86	121
	MinEnd	88.1	140.6	193.6	82	132	184	103	150	206
	GS	110.6	172.4	234.8	100	151	205	138	209	273
	LS	143.5	217.5	291.4	120	182	242	166	248	334
	MReW	147.6	234.8	317.4	111	173	230	237	324	418
	MReO	106.2	165.2	223.9	88	139	194	166	233	314
	Random	141.5	224.8	305.3	103	181	250	227	317	416
Reman 6	FABC	**44.0**	**72.8**	**98.9**	**39**	**71**	**98**	**44**	**75**	**104**
	FDHS	44.5	72.6	98.5	44	69	94	52	76	98
	MinEnd	70.8	118.7	162.0	62	100	139	67	134	186
	GS	96.0	156.1	213.9	84	133	173	108	178	246
	LS	109.6	178.0	243.3	83	154	215	136	214	293
	MReW	142.2	223.1	302.8	94	162	219	219	313	415
	MReO	102.8	153.5	204.8	66	114	153	162	210	271
	Random	133.2	212.2	288.0	86	162	221	232	314	411

Reman7	FABC	**61.0**	**107.9**	**153.2**	**63**	**105**	**148**	**60**	**113**	**157**
		60.4	107.7	153.3	60	105	149	58	112	159
	MinEnd	96.9	167.2	233.6	94	152	214	100	179	254
	GS	132.9	228.4	315.7	112	205	286	153	255	354
	LS	149.8	265.4	368.5	136	236	330	169	297	414
	MReW	189.8	322.5	446.9	134	258	366	329	471	631
	MReO	159.1	248.1	336.0	100	177	248	231	342	446
	Random	200.6	331.1	453.7	139	250	351	351	492	646
Reman8	FABC	**57.8**	**93.5**	**127.0**	**53**	**88**	**122**	**63**	**96**	**133**
	FDHS	57.7	92.3	126.6	56	89	122	57	95	133
	MinEnd	84.9	141.1	192.8	83	132	172	95	153	206
	GS	120.2	202.2	278.0	102	178	251	139	231	317
	LS	143.5	239.1	329.8	131	207	282	173	284	390
	MReW	228.9	358.1	485.7	152	274	374	326	443	581
	MReO	130.2	204.9	279.3	103	172	231	199	274	367
	Random	208.9	331.9	451.7	134	240	330	350	482	643

Table 8 - 6 CPU time and generation for maximum fuzzy completion time

Instance	CPU time		Generation	Instance	Cpu time		Generation
	FABC	FDHS			FABC	FDHS	
Reman 1	0.44	0.40	1000	Reman 5	30.45	22.38	4000
Reman 2	1.72	1.44	1000	Reman 6	33.75	25.97	4000
Reman 3	2.29	1.89	1000	Reman7	82.99	73.83	4000
Reman 4	3.23	2.68	1000	Reman8	102.77	103.56	4000

Notes: 2.40GHz CPU

8.4.4 Maximum fuzzy machine workload objective

In both scheduling and rescheduling stages, we calculated the maximum machine workload for current scheduling stage and non-for summary of all scheduling and rescheduling stages. For maximum fuzzy machine workload objective, we compared FABC algorithm to six heuristics, namely MinEnd, GS, LS, MReW, MaxReO and random rule. To show FABC algorithm performance, the following formula is employed:

$$\boldsymbol{Imp(i)} = \frac{(\boldsymbol{t_1^i+2t_2^i+t_3^i})/\boldsymbol{4}-(\boldsymbol{t_1^0+2t_2^0+t_3^0})/\boldsymbol{4}}{(\boldsymbol{t_1^0+2t_2^0+t_3^0})/\boldsymbol{4}} \times \boldsymbol{100}\% \tag{8-6}$$

where t_1^i, t_2^i and t_3^i are the results of i^{th} heuristic represented by TFN; t_1^0, t_2^0 and t_3^0 are the result of FABC algorithm represented by TFN. $(t_1^i + 2t_2^i + t_3^i)/4$ is the first criterion to compare two TFNs. Since the results by FABC algorithm are better than those by six heuristics, the first criterion is enough to compare them. It is clear that the larger the $Imp(i)$ is, FABC algorithm is more competitive than i^{th} heuristic. For five cases in set one, the maximum fuzzy machine workload results of FABC algorithm and $Imp(i)$ values of six heuristics are shown in Table 8 - 7. It can be seen from Table 8 - 7

that the average values by six heuristics are larger than those obtained by FABC by at least 10.6% for five cases. The best values and worst values by six heuristics are larger than those obtained by FABC by at least 4.6% and 14% for five cases, respectively.

For eight instances from remanufacturing, the maximum fuzzy machine workload results of FABC and six heuristics are evaluated. The $Imp(i)$ results of six heuristics to FABC algorithm are also computed. All the results are shown in Table 8 - 8. It can be seen from Table 8 - 8 that FABC algorithm can improve six heuristics average, best and worst values. For instance, Reman 1, FABC finds the same result in 30 runs because this instance is a small-scale problem with 5-jobs, 4-machines and 23-operations. The average values of six heuristics are larger than those obtained by FABC by at least 16.7%. The best values of six heuristics are larger than those of FABC by at least 8.0%. Except for instance Reman 1, the best values of six heuristics are larger than those of FABC by at least 14.4%. The worst values of six heuristics are larger than those of FABC by at least 20.3%.

To show the maximum machine workload results of FABC clearly, Fig. 8 - 4 and Fig. 8 - 5 show the best fuzzy Gantt charts of case 1 and instance Reman 2. The maximum fuzzy machine workload of case1 is (17, 25, 35) in the best solution while the maximum fuzzy machine workload of instance Reman 2 is (13, 30, 41). In addition, the CPU times and generations of FABC algorithm are shown in Table 8 - 9. For five cases in set one and the previous four instances in set two, the generations are 1000 and the CPU times are less than 3.5 seconds. For the last four instances in set two, the generations are 4000 and the CPU times are 25.23, 30.07, 66.65 and 107.97 seconds respectively. With respect to the scale of problems, the CPU time is very short for all thirteen problems.

To show the convergence of FABC for maximum fuzzy machine workload objective, Fig. 8 - 6 presents the curves of three ranking criteria results by FABC for case 5 and Reman 8. Similar to the convergence of maximum fuzzy completion time, the first ranking criterion is non-increasing for two instances. The second and third ranking criteria may increase or decrease while the third ranking criterion has larger fluctuations than the second one. In summary, FABC algorithm has good performance for fuzzy maximum machine workload objective for FJSSP with fuzzy processing time.

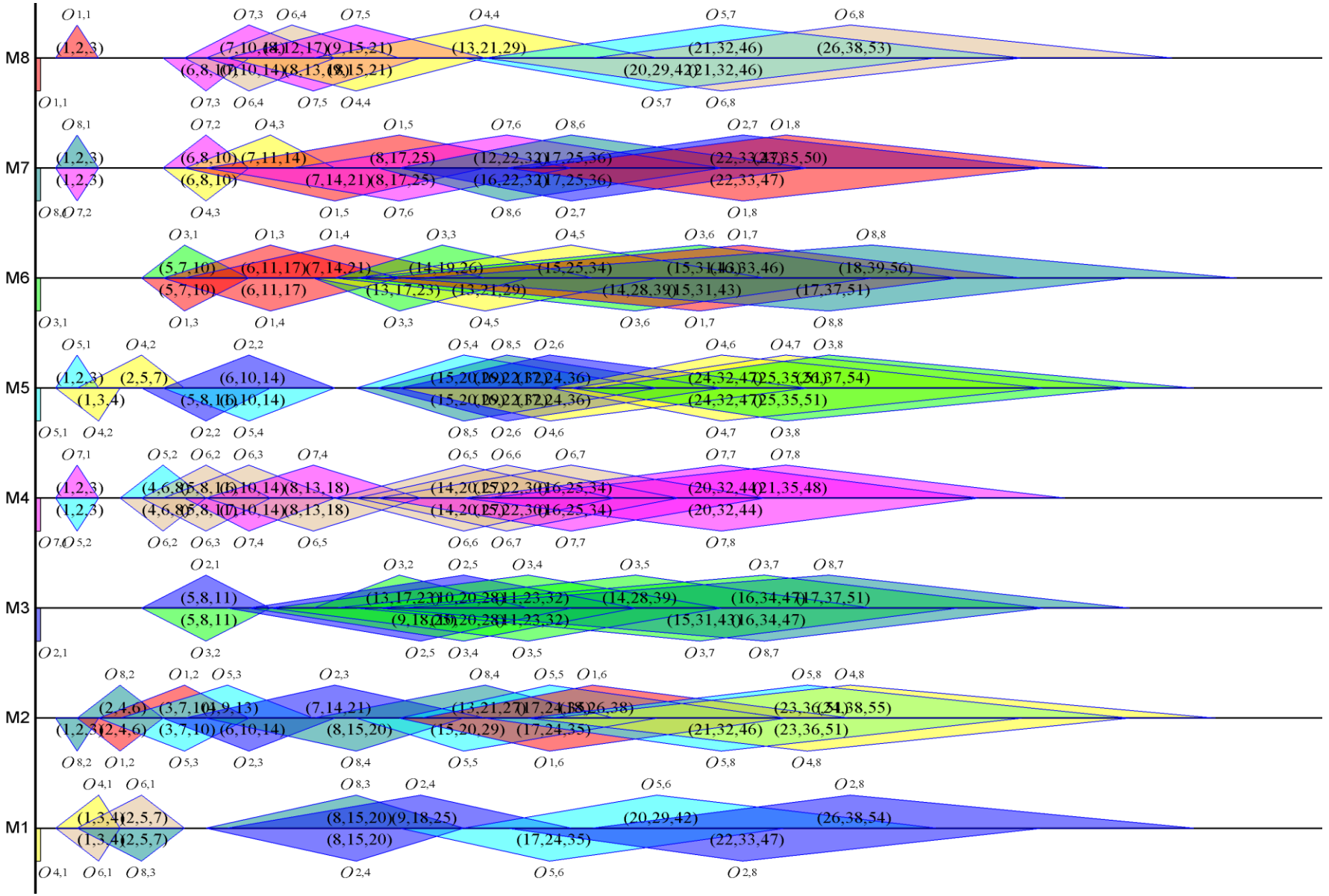

Fig. 8 - 2 Best maximum fuzzy completion time Gantt chart of Reman 2 by FABC

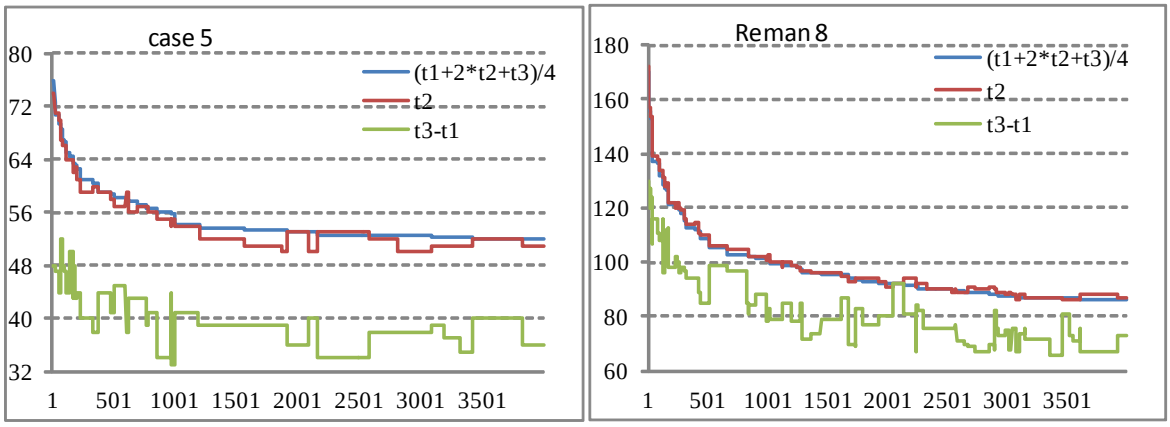

Fig. 8 - 3 Maximum fuzzy completion time convergence curves of case 5 and Reman 8

Table 8 - 7 Maximum fuzzy machine workload by EABC and improvement to six heuristics

Instance	Algorithm	Average value			%	Best value			%	Worst value			%
Case1	FABC	17.3	25.6	35.4		17	25	35		19	26	36	
	MinEnd				32.1				18.6				51.4
	GS				18.4				10.8				25.2
	LS				89.6				93.1				84.1
	MReW				152.6				71.6				240.2
	MReO				139.5				60.8				274.8
	Random				130.5				52.9				206.5
Case2	FABC	27.7	38.2	49.7		29	38	46		28	39	50	
	MinEnd				31.4				11.3				59.0
	GS				16.1				4.6				23.1
	LS				95.1				98.7				92.3
	MReW				119.4				55.0				234.0
	MReO				147.3				54.3				278.8
	Random				131.9				58.9				230.8
Case3	FABC	27.8	39.9	52.3		28	38	51		27	40	57	
	MinEnd				28.7				7.7				46.3
	GS				13.5				12.9				20.7
	LS				76.4				81.9				72.0
	MReW				123.0				71.0				215.2
	MReO				137.2				49.7				222.6
	Random				117.4				60.0				201.8
Case4	FABC	19.2	29.5	42.3		17	29	43		19	30	44	
	MinEnd				34.6				18.6				52.8
	GS				17.9				13.6				26.8
	LS				32.8				35.6				30.1
	MReW				125.5				68.6				226.0
	MReO				120.9				45.8				225.2
	Random				103.1				48.3				173.2

Case5	FABC	35.8	53.5	74.2		34	51	73		35	54	79	
	MinEnd				21.5				15.8				32.0
	GS				10.6				11.0				14.0
	LS				44.2				49.8				41.0
	MReW				113.8				71.8				182.0
	MReO				111.4				65.6				159.5
	Random				120.5				64.6				267.1

Table 8 - 8 Maximum fuzzy machine workload results for eight Reman instances

Instance	Algorithm	Average value			%	Best value			%	Worst value			%
Reman 1	FABC	13	22	30		13	22	30		13	22	30	
	MinEnd	19.2	29.6	39.8	35.9	13	24	33	8.0	29	39	55	86.2
	GS	16.6	27.2	36.9	24.0	13	24	33	8.0	20	32	45	48.3
	LS	15	28	39	26.4	15	28	39	26.4	15	28	39	26.4
	MReW	22.6	34.9	47	60.2	14	24	33	9.2	46	64	82	194.3
	MReO	21	33.1	44.8	51.7	14	24	33	9.2	41	55	71	155.2
	Random	21.6	33.7	45.1	54.1	15	26	37	19.5	40	55	71	154.0
Reman 2	FABC	16.5	29.5	41.1		13	30	41		17	30	42	
	MinEnd	27.8	44.4	60.3	51.7	23	38	51	31.6	38	56	80	93.3
	GS	23.6	37.9	52.3	30.1	20	34	48	19.3	31	46	65	58.0
	LS	31	45	65	59.5	31	45	65	63.2	31	45	65	56.3
	MReW	32.1	52	71.1	77.7	24	39	53	36.0	58	86	111	186.6
	MReO	33.6	53.5	73.8	83.9	26	41	54	42.1	63	88	120	201.7
	Random	33.5	53.2	73.2	82.8	27	39	55	40.4	57	78	109	170.6
Reman 3	FABC	32.3	53.3	73.1		34	53	71		36	54	70	
	MinEnd	45.7	72.2	97	35.4	41	63	82	18.0	53	82	111	53.3
	GS	42.8	67.9	91.3	27.3	37	61	88	17.1	50	75	99	39.7
	LS	51	76	102	43.9	51	76	102	44.5	51	76	102	42.5
	MReW	56.4	86.6	114.8	62.5	40	64	85	19.9	96	129	167	143.5
	MReO	58.2	90.4	121.1	69.9	33	62	88	16.1	93	132	174	148.1
	Random	56.5	87.2	116.7	64.0	43	68	91	28.0	91	126	165	137.4
Reman 4	FABC	21.4	37.5	53.1		28	36	48		22	37	56	
	MinEnd	36	53.2	73.5	44.4	33	46	65	28.4	50	67	89	79.6
	GS	36.9	50.9	68.8	38.8	29	45	60	20.9	42	60	85	62.5
	LS	46	72	103	96.0	46	72	103	98.0	46	72	103	92.8
	MReW	49.9	73.2	99.7	98.0	39	58	81	59.5	89	111	146	200.7
	MReO	55.9	80.9	109.5	118.9	42	59	79	61.5	76	108	145	187.5
	Random	52.2	74.2	100.9	101.7	40	56	76	54.1	73	103	143	177.6
Reman 5	FABC	43.5	74.4	104.9		36	75	109		43	74	108	

	MinEnd	64.7	103.7	141.5	39.2	53	95	128	25.8	74	121	165	60.9
	GS	55.6	89.5	122.5	20.2	50	84	120	14.6	60	96	130	27.8
	LS	72	101	137	38.3	72	101	137	39.3	72	101	137	37.5
	MReW	89.2	138	186.3	85.6	67	104	139	40.3	128	183	244	146.8
	MReO	95	146.1	196.1	96.3	67	105	144	42.7	149	208	273	180.3
	Random	83.4	134.3	182.4	79.8	61	101	135	34.9	132	190	255	156.5
Reman 6	FABC	34.5	62	86.2		30	62	87		38	63	85	
	MinEnd	52.8	87.4	119	41.6	45	77	103	25.3	63	100	140	61.8
	GS	45.7	76.2	104.1	23.5	41	71	100	17.4	54	84	111	33.7
	LS	56	88	119	43.4	56	88	119	45.6	56	88	119	41.0
	MReW	77.4	118	159.4	93.2	52	84	115	39.0	119	173	229	178.7
	MReO	69.1	109	148.5	78.0	46	79	107	29.0	136	186	250	204.4
	Random	64.9	105.6	143.4	71.4	50	85	119	40.7	127	178	241	190.8
Reman7	FABC	50.5	97	138.5		49	97	137		50	97	145	
	MinEnd	75.5	129.4	181.6	34.7	63	116	166	21.3	90	144	196	47.6
	GS	67.7	115.6	162.2	20.4	65	111	158	17.1	76	122	166	24.9
	LS	66	122	172	25.8	66	122	172	26.8	66	122	172	23.9
	MReW	115.8	188.8	259.2	96.5	79	146	198	49.7	179	251	340	162.5
	MReO	107.8	181.5	249.2	88.0	77	137	195	43.7	169	263	354	169.7
	Random	101.4	174.4	240.5	80.3	78	139	200	46.3	157	233	313	140.6
Reman8	FABC	40.6	75.3	106.8		42	74	101		44	77	107	
	MinEnd	59.6	100.3	138.9	33.9	52	91	129	24.7	71	120	163	55.4
	GS	50	88.1	121.5	16.7	42	85	121	14.4	49	95	128	20.3
	LS	62	101	139	35.2	62	101	139	38.5	62	101	139	32.1
	MReW	90.8	151.1	207.3	101.4	61	110	156	50.2	158	215	286	186.6
	MReO	81.4	140	192.4	85.8	51	110	161	48.5	139	200	266	163.9
	Random	93.6	152.7	206.8	103.3	55	108	149	44.3	154	227	301	198.0

Table 8 - 9 Average CPU time and generation for maximum fuzzy machine workload objective

Instance	CPU time	Generation	Instance	CPU time	Generation
case 1	1.01	1000	Reman 3	2.24	1000
case 2	1.01	1000	Reman 4	3.32	1000
case 3	1.37	1000	Reman 5	25.23	4000
case 4	1.36	1000	Reman 6	30.07	4000
case 5	2.59	1000	Reman7	66.65	4000
Reman 1	0.48	1000	Reman8	107.97	4000
Reman 2	1.76	1000			

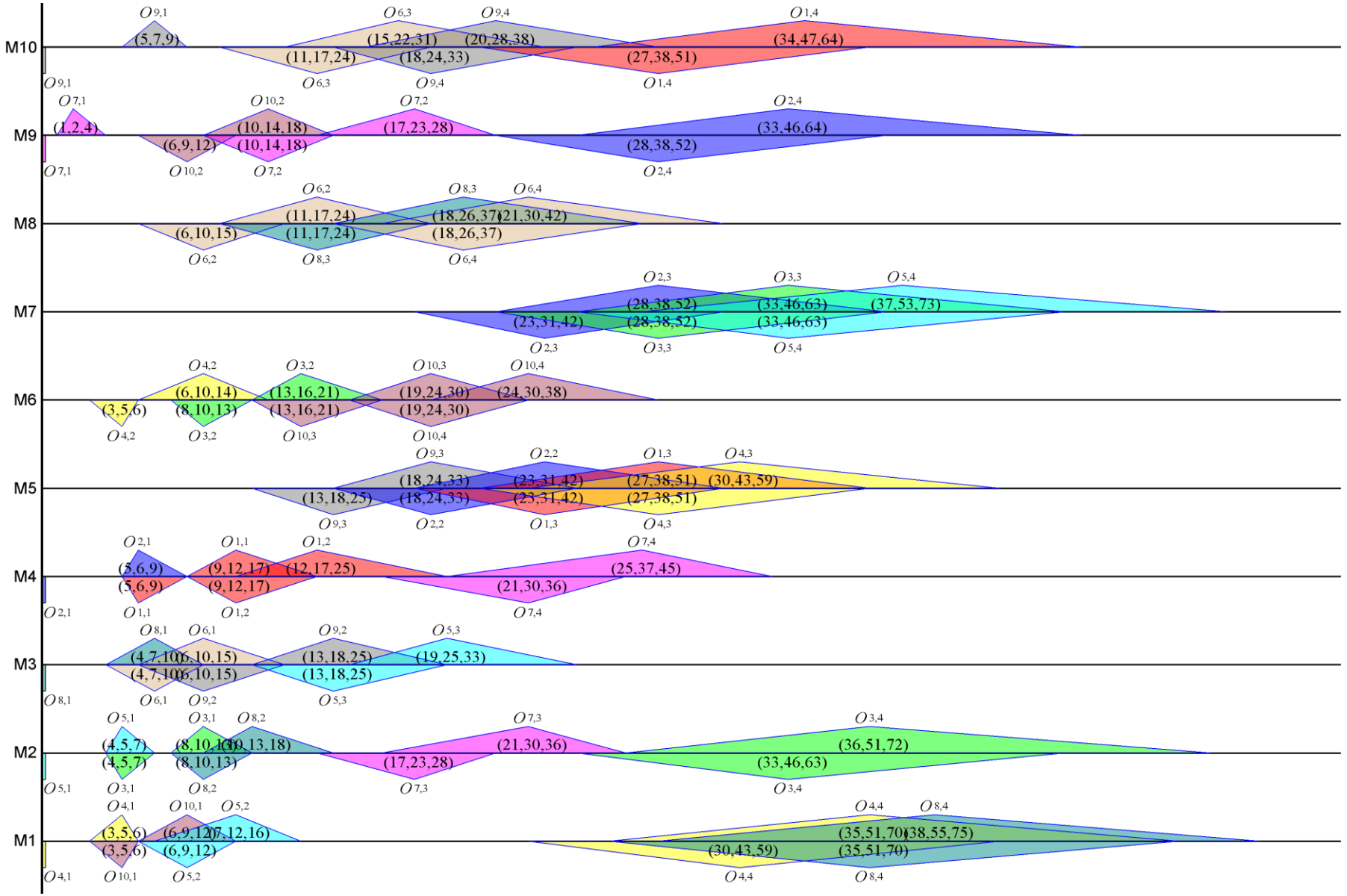

Fig. 8 - 4 Best maximum fuzzy machine workload Gantt chart of case 1 by FABC

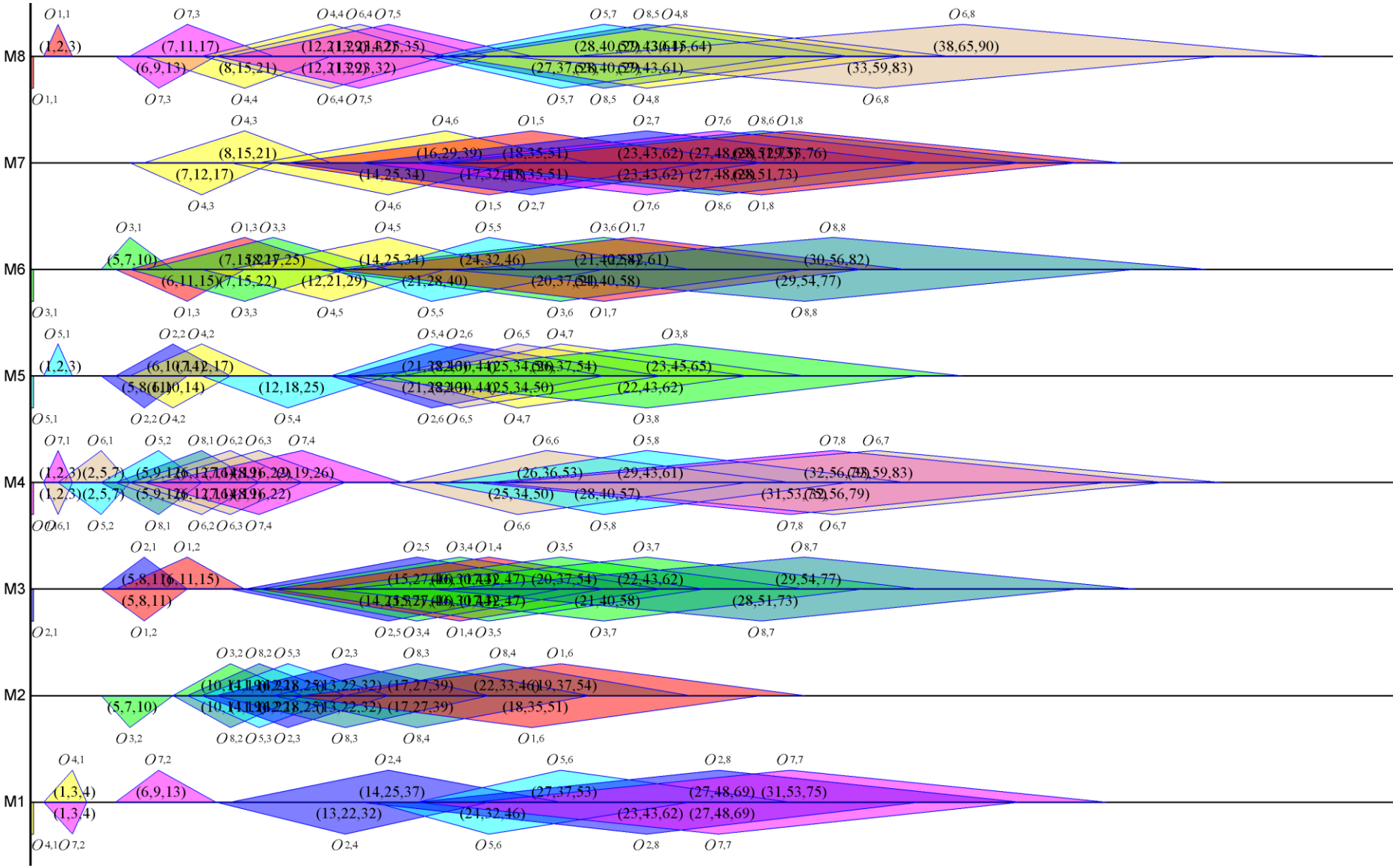

Fig. 8 - 5 Best maximum fuzzy machine workload Gantt chart of Reman 2 by FABC

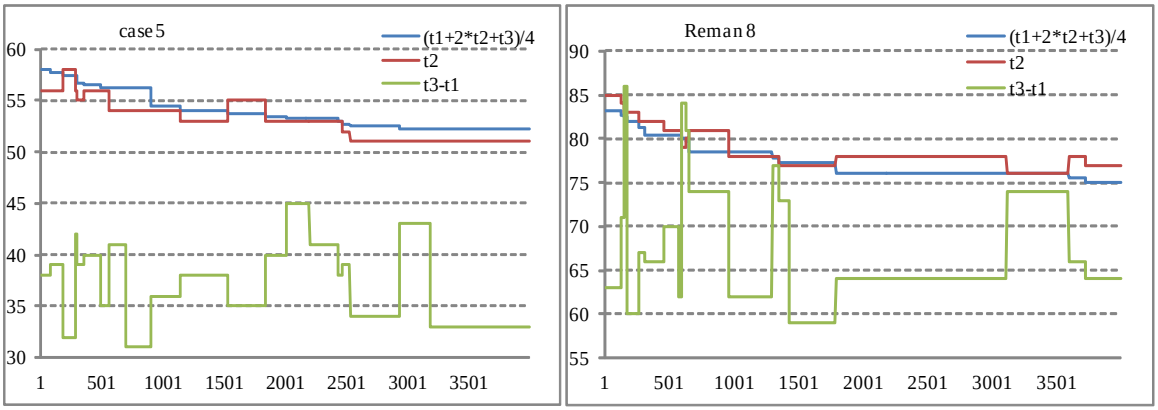

Fig. 8 - 6 Maximum fuzzy machine workload convergence curves of case 5 and Reman 8

8.5 Conclusions

In this Chapter, an effective fuzzy artificial bee colony (FABC) algorithm was proposed to solve flexible job shop scheduling problem with fuzzy processing time. The objectives were to minimize the maximum fuzzy completion time and the maximum fuzzy machine workload, respectively. The MinEnd heuristic was evaluated by comparing with five simple heuristics for maximum fuzzy completion time and maximum fuzzy machine workload objectives. FABC was compared to six existing algorithms for solving five benchmark cases. The results and comparisons showed the effectiveness of FABC. Eight instances from remanufacturing were solved by FABC algorithm and six heuristics for maximum fuzzy completion time objective. FABC improved the results of heuristics in short CPU time. For maximum fuzzy machine workload objective, five benchmark cases and eight remanufacturing instances were solved by FABC and six heuristics. The results and comparisons demonstrate improved performance of FABC algorithm. The experiments also investigated the convergence of FABC algorithm. Several convergence curves were shown. In addition, several fuzzy Gantt charts of best solution were shown for maximum fuzzy completion time and maximum fuzzy machine workload objectives.

Chapter 9

Two-stage FABC Algorithm for Multi-objective and Multi-constraint FJSSP

9.1 Introduction

This Chapter researches multi-objective multi-constraints flexible job shop scheduling problem (FJSSP). The constraints are fuzzy processing time and new job insertion. These two constraints are modeled from remanufacturing. The objectives are to minimize the maximum fuzzy completion time and the maximum fuzzy machine workload. Based on the fuzzy artificial bee colony algorithm (FABC) in Chapter 8, we proposed a two-stage FABC (TFABC) algorithm for multi-objective multi-constraints FJSSP. The three-value element encoding and decoding method is used to show the solution in TFABC algorithm. The MinEnd heuristic is employed to initialize population. Extensive computational experiments are carried out using eight instances from remanufacturing. The TFABC algorithm is compared to five heuristics. The results and comparisons demonstrate improved performance of TFABC algorithm for solving multi-objective multi-constraints FJSSP.

The rest of this Chapter is organized as follows: Section 9.2 briefly describe the two objectives and two constraints of FJSSP. The proposed TFABC algorithm is presented in Section 9.3. Section 9.4 is for experimental results and comparisons. We conclude this Chapter in Section 9.5.

9.2 Multi-objective and multi-constraint FJSSP

In this Chapter, we address FJSSP with two constraints, fuzzy processing time and new job insertion. The fuzzy processing time constraint is modeled as a triangular fuzzy number (TFN) as follows.

$$t_{i,j,k} = (t^1_{i,j,k}, t^2_{i,j,k}, t^3_{i,j,k}) \tag{9-1}$$

where $t^1_{i,j,k}$, $t^2_{i,j,k}$ and $t^3_{i,j,k}$ are three probable processing time of operation $O_{i,j}$ on machine M_k. The three operators for fuzzy number in Chapter 3 are employed to compare and rank the triangular fuzzy numbers. New job insertion constraint is solved by rescheduling. We employ the rescheduling strategy II proposed in Chapter 5 to obtain the rescheduling solution after new job insertion. The crucial idea of rescheduling strategy II is to reschedule the operations of new job(s) and the existing jobs' operations that are not yet started on the new job insertion time.

There is an example to show the effectiveness of rescheduling for new job(s) insertion constraint. Fig. 9 - 1 shows a Gantt chart of 5-jobs, 4-machines and 23-operations FJSSP with fuzzy processing time.

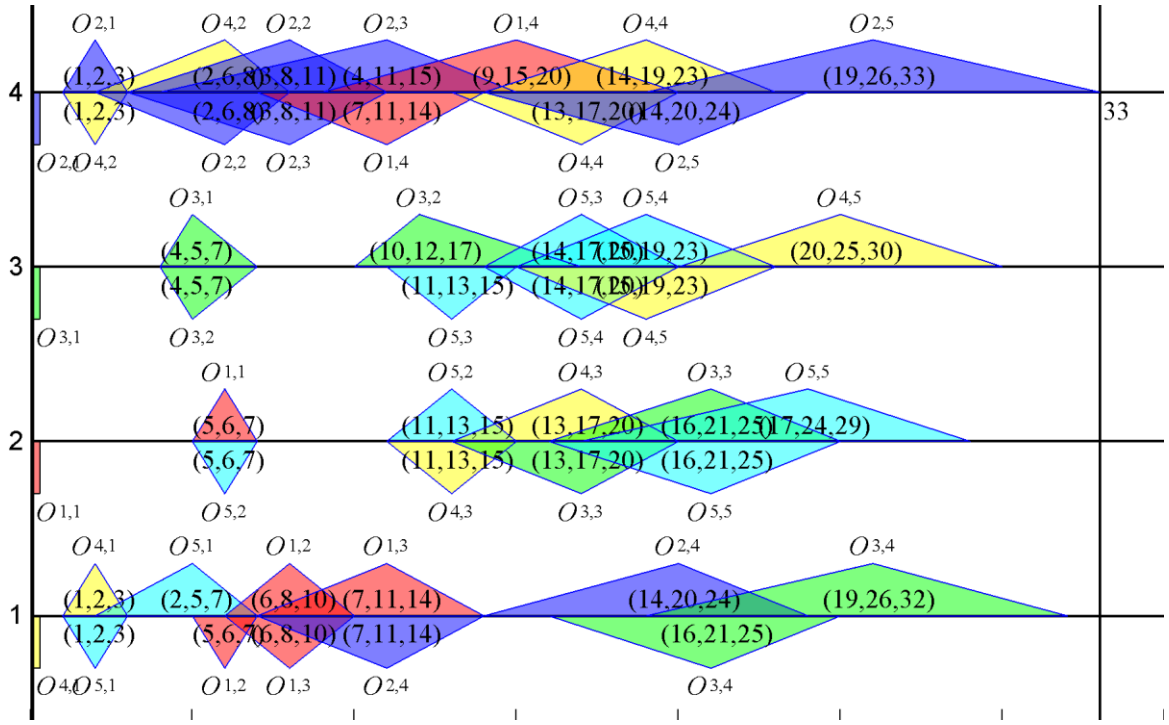

Fig. 9 - 1 An example Gantt chart for FJSSP with fuzzy processing time

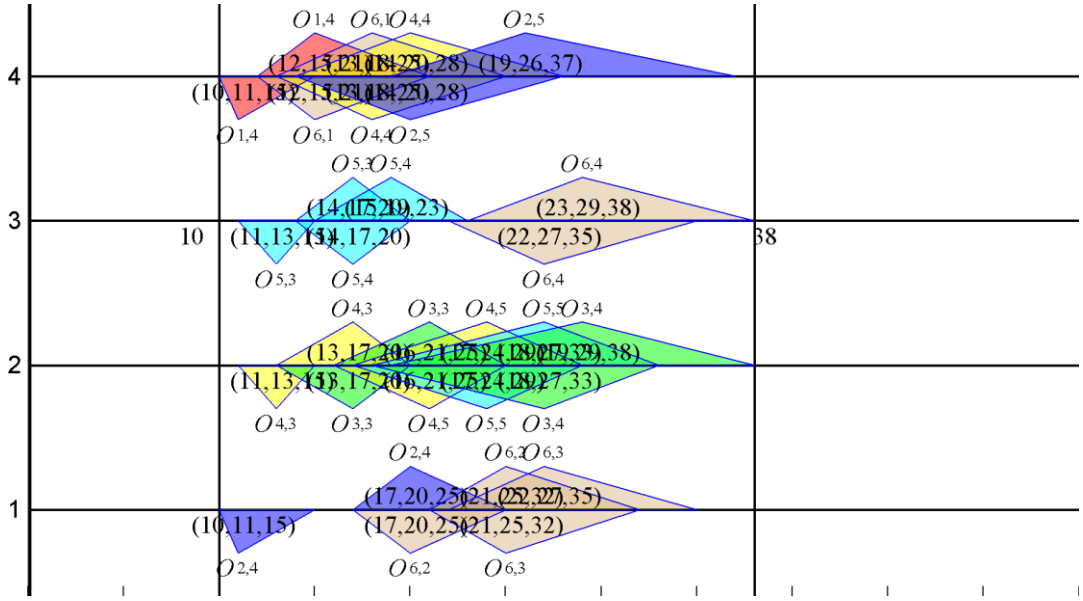

Fig. 9 - 2 The Gantt chart by rescheduling strategy II

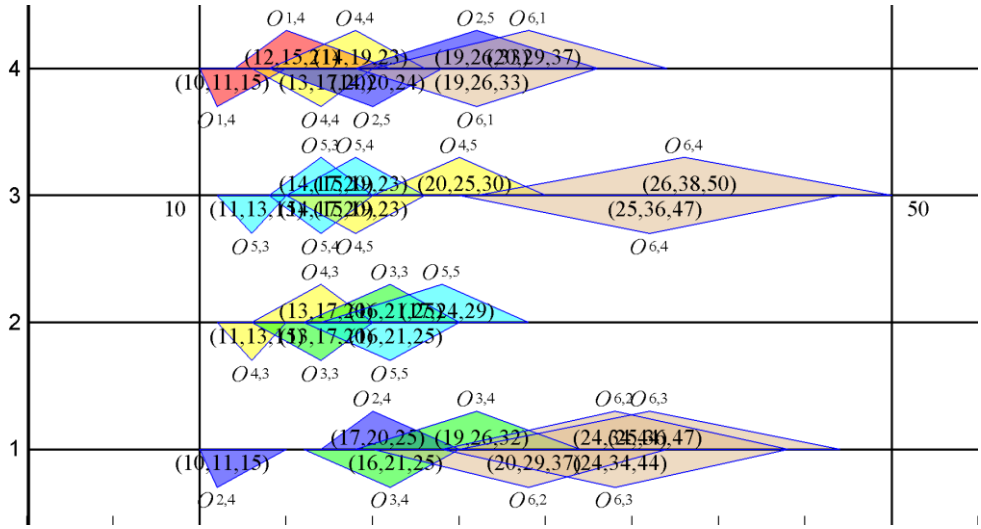

Fig. 9 - 3 The Gantt chart after rescheduling new job only

In this Gantt chart, the maximum fuzzy completion time is (19, 26, 33). At time 10, a new job, job6, comes and is inserted into the existing solution. Rescheduling operator is activated to solve a new feasible solution for new job and existing jobs' operations that are not yet started at the new job insertion time. The Gantt chart after rescheduling is shown in Fig. 9 - 2. The rescheduling maximum fuzzy processing time is (23, 29, 38). The result by rescheduling strategy II is compared to that by rescheduling strategy I in Chapter 5. The Gantt chart of rescheduling strategy I is shown in Fig. 9 - 3. It can be seen from Fig. 9 - 3 that the maximum fuzzy completion time of rescheduling strategy I is (26, 38, 50). The rescheduling strategy II can improve the result of rescheduling strategy I by (11.5%, 23.6%, 24%). Hence, it is necessary to execute rescheduling and rescheduling strategy II is effective.

Two objectives, maximum fuzzy completion time and maximum fuzzy machine workload, are considered. The fuzzy completion time of operation $O_{i,j}$ is a TFN as follows.

$$C_{i,j} = (C^1_{i,j}, C^2_{i,j}, C^3_{i,j}) \tag{9-2}$$

where $C^1_{i,j}$, $C^2_{i,j}$ and $C^3_{i,j}$ are three probable completion times of operation $O_{i,j}$.

The fuzzy machine workload of machine M_k is also a TFN as follows.

$$w_k = (w^1_k, w^2_k, w^3_k) \tag{9-3}$$

w^1_k, w^2_k and w^3_k are three probable machine workload of machine M_k.

The objective is to minimize maximum fuzzy completion time and maximum fuzzy machine workload. The maximum fuzzy completion time denoted by C_M can be calculated by the following formula.

$$\text{Min } C_M = \max_{1 \le i \le n} \{C_i\} \tag{9-4}$$

where C_i is the fuzzy completion time of job J_i.

The maximum fuzzy machine workload denoted by W_M, can be calculated by:

$$\text{Min } W_M = \max_{1 \le k \le m} \{W_k\} \tag{9-5}$$

where W_k is the workload of machine M_k.

9.3 Two-stage FABC algorithm for FJSSP

The proposed TFABC algorithm includes two stages: scheduling stage and re-scheduling stage for new job insertion. The TFABC algorithm employed MinEnd heuristic, global minimizing processing time rule, and random rule to initialize the population. 90% solutions in the population are generated randomly, 5% solutions by MinEnd heuristic and 5% solutions by GS heuristic rule. In the procedure to generate new food source, changing processing machine operator and operation inserting operator are employed for exploitation search. PBetter or worse food source selection and scout bee operator are used for exploration search.

At the scheduling stage, the start times of all jobs and machines are the same. After the scheduling stage, the solution with the best result will be output, and all operations will be processed on the corresponding machines based on the best solution. When a new job comes and is inserted into existing schedule, the re-scheduling stage will be activated. All non-started operations of existing jobs and new job's operations will be rescheduled based on the re-scheduling strategy II. After rescheduling, new scheduling result replaces the existing one from the new job insertion time and is executed in the shop floor.

The computational procedure of TFABC algorithm for scheduling and re-scheduling can be described as follows:

Scheduling stage:

Step1: Set parameters, employed bee number, onlooker bee number, scout bee number and so on.

Step2: Initialize population using MinEnd heuristic, GS rule and random rule.

Step3: Perform employed bee phase.

Step4: Perform onlooker bee phase.

Step5: Update the best solution.

Step6: If $Limit$ is met, perform scout bee phase

Step7: If the stop criterion is not satisfied, go to Step 3; else, output the best solution.

Re-scheduling stage:

Step8: The new job comes and is inserted into the existing scheduling solution.

Step9: Calculate and record the available time of each job and each machine based on the new job insertion time.

Step10: For existing jobs, Re-calculate the number of operations that are not yet started. Add the new job(s) to the re-scheduling job set.

Step11: Execute re-scheduling using the algorithm in scheduling stage.

Step12: Link up the re-scheduling results to executing scheduling results from the insertion time and execute the rescheduling solution in the shop floor.

For the TFABC algorithm, the computational complexity of the first stage is the same to the complexity of FABC algorithm in Chapter 8. When the new job is inserted into the executing sequence, the first stage will be executed. Hence, the computational complexity of the TFABC algorithm is based on the number of times of new job insertion.

9.4 Experiment evaluation and comparisons

9.4.1 Experiment setup

Eight instances from remanufacturing industry is solved by the proposed TFABC algorithm. The instance size is from 5-jobs, 4-machines and 23-operations to 20-jobs, 15-machines and 355-operations. 35 new jobs are inserted to existing scheduling sequence of eight instances when the instances are processing in the shop floor. Each insertion has different insertion time, job number and operation number. The detail information of eight instances with 35 new jobs is shown in Table 9 - 1. The first two columns are instance number and new job insertion times. The third column is the insertion time of new jobs. The last three columns are the corresponding job number, machine number

and operation number. The first row of each instance shows the job number, machine number and operation number at scheduling stage.

Table 9 - 1 Information of eight instances and new jobs

Instance	Insertion Times	Insertion time	Job Number	Machine Number	Operation number
1	0	0	5	4	23
	1	10	1		4
	2	15	1		5
2	0	0	8	8	64
	1	12	1		8
	2	24	1		8
3	0	0	10	6	81
	1	15	1		8
	2	25	1		7
	3	35	1		6
4	0	0	10	10	100
	1	15	2		20
	2	30	2		20
5	0	0	15	8	171
	1	21	2		18
	2	47	2		17
6	0	0	15	10	185
	1	14	2		20
	2	31	2		19
	3	44	2		19
7	0	0	20	10	308
	1	21	3		38
	2	52	3		32
8	0	0	20	15	355
	1	18	3		42
	2	34	3		42
	3	49	2		25

The TFABC algorithm is coded in C++ and runs on Intel 2.40 GHz PC with 2 GB memory. The parameters are set based on our previous research and the compared algorithms as follows: employed bee 50; onlooker bee 100, scout bee 20, $Limit$ 50, P1=0.8, P2=0.95, P3=0.2 and P4=0.5. For the previous four instances, the generation is set to 1000. For the last four instances, the generation is set to 4000 because the problem size of last four instances is larger than that of the previous instances. All experiments are carried out with 30 replications.

9.4.2 Maximum fuzzy completion time objective

To solve the eight instances with 19 times new job(s) insertion, TFABC algorithm is employed obtain the scheduling and rescheduling results. TFABC is compared to five heuristics, MinEnd, global minimizing processing time rule (GS), minimizing processing time (LS), most number of operations remaining (MReO) and most work remaining rule (MReW). The average maximum fuzzy completion time, best maximum fuzzy completion time and worst maximum fuzzy completion time by FABC

algorithm and five heuristics are counted. The detail results for eight instances and 19 times new job(s) insertion are shown in Table 9 - 2 to Table 9 - 9.

Table 9 - 2 Maximum fuzzy processing time results of instance 1

Algorithm	Insertion time	Average			Best			Worst			CPU Time (s)
TFABC	0	17.8	26.0	34.2	19	26	33	17	26	35	0.414
MinEnd		24.6	38.4	51.1	20	31	42	35	48	63	
GS		31.7	47.5	63.2	25	36	48	41	61	81	
LS		29.0	46.6	61.2	17	34	45	40	61	79	
MReO		28.9	41.8	55.9	16	30	42	57	76	96	
MReW		29.6	47.3	64.7	20	33	45	58	80	107	
TFABC	1	22.8	29.0	38.0	21	29	38	23	29	38	0.266
MinEnd		22.1	34.3	46.0	18	30	40	25	40	55	
GS		26.7	42.7	57.6	22	34	47	39	55	75	
LS		28.6	42.8	57.8	23	34	45	35	56	74	
MReO		31.0	40.7	53.8	22	35	48	43	55	72	
MReW		29.8	45.0	60.2	20	34	46	57	77	102	
TFABC	2	24.0	32.0	42.0	24	32	42	24	32	42	0.286
MinEnd		27.9	37.3	51.5	26	32	46	32	43	56	
GS		32.2	47.1	64.2	26	36	51	48	70	97	
LS		30.0	46.7	63.9	25	37	52	38	57	78	
MReO		34.8	46.5	60.5	28	39	53	47	65	86	
MReW		37.5	51.1	67.3	26	37	46	61	81	114	

Table 9 - 2 shows the scheduling results and rescheduling results after two times new job insertion for instance 1. The first and second columns are compared algorithms and new job(s) insertion times. The value "0" in the second column means the scheduling results for instance 1 while the values "1" and "2" mean the two times rescheduling results after two times new job(s) insertion. It can be seen from Table 9 - 2 that TFABC algorithm can obtain better results for average, best and worst measures than those by five compared heuristics in both scheduling stage and two times rescheduling stages. We can get the same conclusions for remaining seven instances from Table 9 - 3 to Table 9 - 9. For the remaining seven instances and 17 times new job(s) insertion, the TABC algorithm can obtain better scheduling and rescheduling results than those by five heuristics.

In addition, TFABC algorithm becomes more competitive than the five compared heuristics when the instance scale becomes larger from instance 1 to instance 8. For example, TABC gets the average (17.8, 26.0, 34.2), best (19, 26, 33) and worst (17, 26, 35) results for instance 1. The best results of five heuristics are average (24.6, 38.4, 51.1) by MinEnd, best (16, 30, 42) by MReO and worst (35,

48, 63) by MinEnd. The TFABC algorithm can improve the best results of five heuristics by average (6.8, 12.4, 16.9), best (-3, 4, 9) and worst (18, 22, 28). For instance 8, the TFABC algorithm can improve the best results of five heuristics by average (32.6, 52.5, 71.7), best (26, 41, 54) and worst (30, 68, 96). After new job(s) insertion, the rescheduling results have the same conclusions. For example, the TFABC algorithm can improve the best instances 1 results of five heuristics by average (3.9, 5.3, 9.5), best (2, 0, 4) and worst (8, 11, 14) after the second time new job insertion. For instance 8, the TFABC algorithm can improve average, best and worst results by (33.8, 53.7 69.8), (36, 42, 44) and (50, 78, 91).

The CPU times of five heuristics for eight instances are very small and can be cancelled. For eight instances and 19 times new job(s) insertion, the TFABC algorithm's average CPU times in 30 repeat are shown in the last column of Table 9 - 2 to Table 9 - 9. The maximum CPU time is less than 2 minutes. Compared to the operation processing time in practical shop floor, the CPU times of TFABC algorithm can be cancelled.

For maximum fuzzy completion time objective, Fig. 9 - 4 to Fig. 9 - 6 show TFABC algorithm scheduling results for instance Reman2 and two times new job insertions. Fig. 9 - 4 shows the scheduling results of instance Reman2 and the maximum fuzzy completion time is (17, 29, 58) on machine 6. Fig. 9 - 5 shows the rescheduling results after new job insertion at time 12. For all jobs and machines, the start time is at least 12. After rescheduling, the maximum completion time after new job 9 insertion is (31, 44, 62) on machine 5 and machine 8. Fig. 9 - 6 shows the Gantt chart after the second time new job insertion on time 24. The rescheduling result of maximum fuzzy processing time is (38, 51, 75) on machine 6.

In summary, TFABC algorithm solves the eight instances from remanufacturing industry effectively. The results of TFABC are compared to those by five heuristics. The comparisons show that TFABC algorithm can obtain satisfactory results for real instances in remanufacturing industry and the TFABC algorithm needs very limited CPU time to get scheduling and rescheduling results.

Table 9 - 3 Maximum fuzzy processing time results of instance 2

Algorithm	Insertion time	Average			Best			Worst			CPU Time (s)
TFABC	0	24.3	40.2	56.7	17	39	58	25	43	60	1.483
MinEnd		39.5	64.0	88.9	30	55	78	50	77	108	
GS		53.6	83.8	116.3	44	66	91	73	116	160	
LS		59.3	95.2	134.3	47	78	108	69	109	153	
MReO		49.9	75.5	104.8	37	59	80	77	100	140	
MReW		69.1	106.7	147.2	48	75	107	119	167	228	

TFABC	1	31.9	45.0	59.2	31	44	62	30	45	61	0.895
MinEnd		45.8	68.5	92.4	41	54	77	55	83	114	
GS		60.6	88.9	122.5	48	73	100	70	109	147	
LS		61.4	94.1	130.6	59	79	112	75	118	164	
MReO		62.3	85.9	114.7	43	68	92	95	129	170	
MReW		73.3	109.4	150.3	53	83	114	114	164	213	
TFABC	2	38.1	51.1	75.0	38	51	75	40	53	75	0.679
MinEnd		53.3	73.2	99.8	45	62	81	60	87	119	
GS		64.9	90.4	126.0	59	76	105	87	126	178	
LS		67.7	93.5	129.7	59	76	107	81	113	155	
MReO		65.3	87.5	119.0	47	67	100	91	121	164	
MReW		74.8	106.4	145.6	52	75	106	119	169	221	

Table 9 - 4 Maximum fuzzy processing time results of instance 3

Algorithm	Insertion time	Average			Best			Worst			CPU Time (s)
TFABC	0	37.0	59.1	80.0	34	57	79	41	60	82	1.971
MinEnd		62.4	98.7	133.8	49	84	113	74	119	166	
GS		76.7	117.7	159.5	56	91	120	91	133	179	
LS		82.9	128.7	173.9	67	101	135	108	160	210	
MReO		69.7	104.4	137.2	47	83	116	115	146	185	
MReW		94.9	144.6	194.0	61	102	138	142	198	262	
TFABC	1	43.3	61.5	83.4	44	60	82	45	62	84	1.396
MinEnd		66.6	99.2	132.9	62	78	101	74	116	154	
GS		81.8	119.4	160.2	65	103	144	93	138	189	
LS		89.0	134.0	181.1	72	104	138	116	176	237	
MReO		68.5	100.5	133.6	52	85	115	101	138	177	
MReW		98.4	147.1	196.3	65	98	134	171	235	310	
TFABC	2	52.9	68.7	91.7	54	68	90	56	71	91	1.101
MinEnd		75.2	103.8	138.4	66	90	121	85	117	153	
GS		95.4	131.9	176.5	81	106	147	115	167	225	
LS		99.2	140.8	187.9	83	119	158	117	169	224	
MReO		85.4	114.9	150.4	75	94	122	113	148	194	
MReW		112.3	154.3	203.3	87	119	155	164	222	291	
TFABC	3	58.8	71.5	98.9	59	71	99	60	73	99	0.833
MinEnd		82.0	106.6	140.9	74	97	124	95	129	168	
GS		97.9	132.4	179.1	78	106	140	114	157	208	
LS		105.8	143.9	189.1	94	126	164	130	177	239	
MReO		84.9	110.4	145.7	76	93	124	117	148	195	
MReW		111.8	151.1	200.2	84	120	166	161	216	290	

Table 9 - 5 Maximum fuzzy processing time results of instance 4

Algorithm	Insertion time	Average			Best			Worst			CPU Time (s)
TFABC	0	33.8	50.8	70.6	28	49	73	54	69	92	2.768
MinEnd		49.8	78.1	109.1	40	66	97	62	91	126	
GS		67.2	103.9	141.7	58	91	126	94	131	176	
LS		74.0	118.9	166.4	55	101	144	91	142	198	
MReO		64.4	95.0	129.6	48	77	106	112	147	188	
MReW		91.0	140.0	192.6	56	102	143	146	199	275	
TFABC	1	39.4	56.8	76.9	38	55	74	37	59	84	1.948
MinEnd		60.8	83.9	114.3	50	71	105	74	101	136	
GS		78.1	113.1	154.8	70	91	123	94	136	182	
LS		75.7	119.5	167.6	67	104	145	91	140	193	
MReO		80.6	115.3	156.5	61	86	113	108	147	200	
MReW		97.8	142.1	194.0	58	100	142	144	193	254	
TFABC	2	53.0	68.7	92.0	51	68	93	38	54	73	1.565
MinEnd		66.5	90.4	123.3	56	78	107	81	115	154	
GS		90.0	121.1	162.1	73	95	130	105	141	173	
LS		88.6	132.5	181.2	78	110	150	104	154	208	
MReO		93.0	124.3	165.5	70	96	132	116	158	210	
MReW		112.1	158.4	215.1	79	121	166	173	227	296	

Table 9 - 6 Maximum fuzzy processing time results of instance 5

Algorithm	Insertion time	Average			Best			Worst			CPU Time (s)
TFABC	0	51.8	83.4	115.0	51	81	114	56	87	115	23.617
MinEnd		83.2	132.2	181.3	68	119	169	105	165	227	
GS		110.6	172.4	234.8	100	151	205	138	209	273	
LS		143.5	217.5	291.4	120	182	242	166	248	334	
MReO		106.2	165.2	223.9	88	139	194	166	233	314	
MReW		147.6	234.8	317.4	111	173	230	237	324	418	
TFABC	1	63.1	90.4	125.6	60	92	120	66	91	128	16.216
MinEnd		98.7	142.5	194.4	89	128	174	113	160	219	
GS		117.9	172.7	234.4	94	144	205	149	205	286	
LS		133.2	203.2	276.3	112	165	225	153	234	320	
MReO		125.0	181.9	247.2	93	147	206	186	246	332	
MReW		154.5	231.8	314.5	118	186	261	253	341	450	
TFABC	2	75.8	98.0	132.8	77	97	128	79	99	135	9.743
MinEnd		109.8	150.4	202.8	100	132	179	119	164	223	
GS		123.0	168.9	229.3	107	143	200	142	197	267	
LS		150.4	210.3	282.5	132	188	250	172	248	325	

MReO	140.4	187.7	248.5	112	155	209	190	240	321	
MReW	177.5	243.5	329.7	130	195	270	250	334	442	

Table 9 - 7 Maximum fuzzy processing time results of instance 6

Algorithm	Insertion time	Average			Best			Worst			CPU Time (s)
TFABC	0	44.2	72.1	98.5	38	70	100	45	74	103	28.123
MinEnd		68.6	113.1	154.7	57	102	144	85	142	193	
GS		96.0	156.1	213.9	84	133	173	108	178	246	
LS		109.6	178.0	243.3	83	154	215	136	214	293	
MReO		102.8	153.5	204.8	66	114	153	162	210	271	
MReW		142.2	223.1	302.8	94	162	219	219	313	415	
TFABC	1	49.6	76.6	104.5	52	74	101	48	78	109	21.796
MinEnd		77.3	121.9	167.0	72	110	152	82	137	189	
GS		97.2	156.7	214.0	84	129	176	105	176	243	
LS		113.1	180.1	246.0	99	162	220	139	212	288	
MReO		103.1	154.3	210.0	79	122	169	158	218	281	
MReW		164.0	247.4	331.5	109	174	238	261	364	472	
TFABC	2	58.4	80.9	112.6	54	81	112	60	81	116	15.813
MinEnd		90.0	129.3	177.8	88	115	161	102	142	192	
GS		100.5	154.2	213.9	89	132	184	109	178	244	
LS		117.0	184.5	252.1	100	163	223	144	220	303	
MReO		110.9	157.9	209.6	83	120	165	159	211	264	
MReW		164.2	243.4	326.3	109	174	245	240	325	427	
TFABC	3	107.7	160.1	217.0	65	85	115	64	86	120	12.475
MinEnd		91.1	127.6	173.6	88	120	158	96	136	183	
GS		108.7	157.7	216.9	101	138	185	121	177	238	
LS		118.1	185.5	254.4	92	152	209	140	215	296	
MReO		115.4	159.6	216.1	95	136	180	169	226	302	
MReW		153.3	225.8	307.7	124	186	259	234	317	412	

Table 9 - 8 Maximum fuzzy processing time results of instance 7

Algorithm	Insertion time	Average			Best			Worst			CPU Time (s)
TFABC	0	59.2	107.5	153.4	53	104	155	64	111	155	80.307
MinEnd		98.4	169.6	236.2	88	157	209	111	182	255	
GS		132.9	228.4	315.7	112	205	286	153	255	354	
LS		149.8	265.4	368.5	136	236	330	169	297	414	
MReO		159.1	248.1	336.0	100	177	248	231	342	446	
MReW		189.8	322.5	446.9	134	258	366	329	471	631	
TFABC	1	74.6	121.3	171.2	72	118	170	81	123	174	61.628

MinEnd		114.1	183.1	252.2	96	168	229	129	203	284	
GS		144.5	237.0	327.2	126	208	292	162	274	374	
LS		169.6	279.0	386.6	141	244	347	206	324	445	
MReO		156.4	237.0	323.4	117	175	245	252	347	459	
MReW		216.0	343.7	469.5	144	273	394	336	491	645	
TFABC	2	95.6	130.5	183.6	96	130	179	94	135	188	34.865
MinEnd		133.9	194.4	265.9	123	184	252	151	208	282	
GS		167.9	255.4	350.3	152	236	316	181	279	376	
LS		190.6	296.3	410.3	177	265	378	226	330	452	
MReO		164.0	235.2	325.3	126	191	270	243	328	443	
MReW		222.5	334.8	464.4	164	260	367	332	443	615	

Table 9 - 9 Maximum fuzzy processing time results of instance 8

Algorithm	Insertion time	Average			Best			Worst			CPU Time (s)
TFABC	0	56.8	92.7	127.1	53	89	124	58	95	134	113.064
MinEnd		89.4	145.1	198.8	79	130	178	88	163	230	
GS		120.2	202.2	278.0	102	178	251	139	231	317	
LS		143.5	239.1	329.8	131	207	282	173	284	390	
MReO		130.2	204.9	279.3	103	172	231	199	274	367	
MReW		228.9	358.1	485.7	152	274	374	326	443	581	
TFABC	1	64.4	100.5	141.6	64	93	133	71	100	134	81.105
MinEnd		97.3	152.0	207.4	91	136	189	101	164	228	
GS		130.9	211.4	290.3	111	191	266	146	243	341	
LS		145.0	242.6	335.6	114	201	278	169	289	395	
MReO		142.6	217.3	294.5	102	170	231	221	292	396	
MReW		212.0	336.4	457.9	159	287	398	341	461	591	
TFABC	2	76.2	107.2	147.9	75	101	141	74	108	145	60.931
MinEnd		108.9	158.9	214.5	97	147	199	124	174	227	
GS		138.9	217.8	300.6	112	189	269	155	254	354	
LS		155.0	242.4	336.9	134	208	291	182	280	388	
MReO		148.3	218.1	294.1	111	174	242	233	314	414	
MReW		222.3	340.6	462.0	152	260	364	346	477	624	
TFABC	3	81.8	107.2	149.6	76	105	154	82	108	158	38.420
MinEnd		115.6	161.0	219.4	112	147	198	132	186	249	
GS		145.4	215.3	298.1	134	190	261	171	243	331	
LS		158.1	234.6	325.6	142	206	278	185	268	369	
MReO		158.1	222.5	300.3	126	186	244	229	298	401	
MReW		206.7	319.2	439.1	180	272	382	337	468	603	

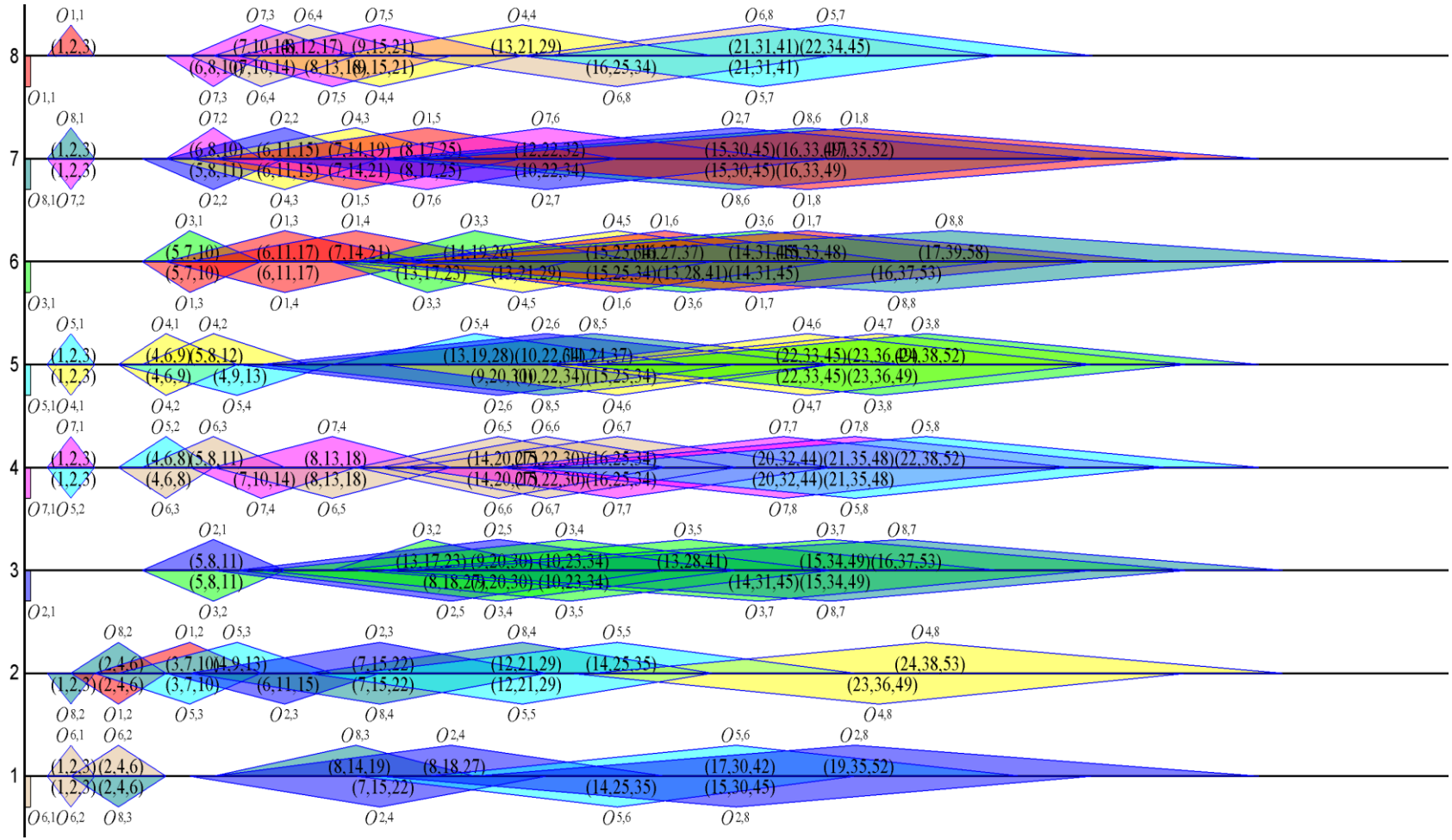

Fig. 9 - 4 Gantt chart of instance Reman 2 with minimum maximum fuzzy completion time

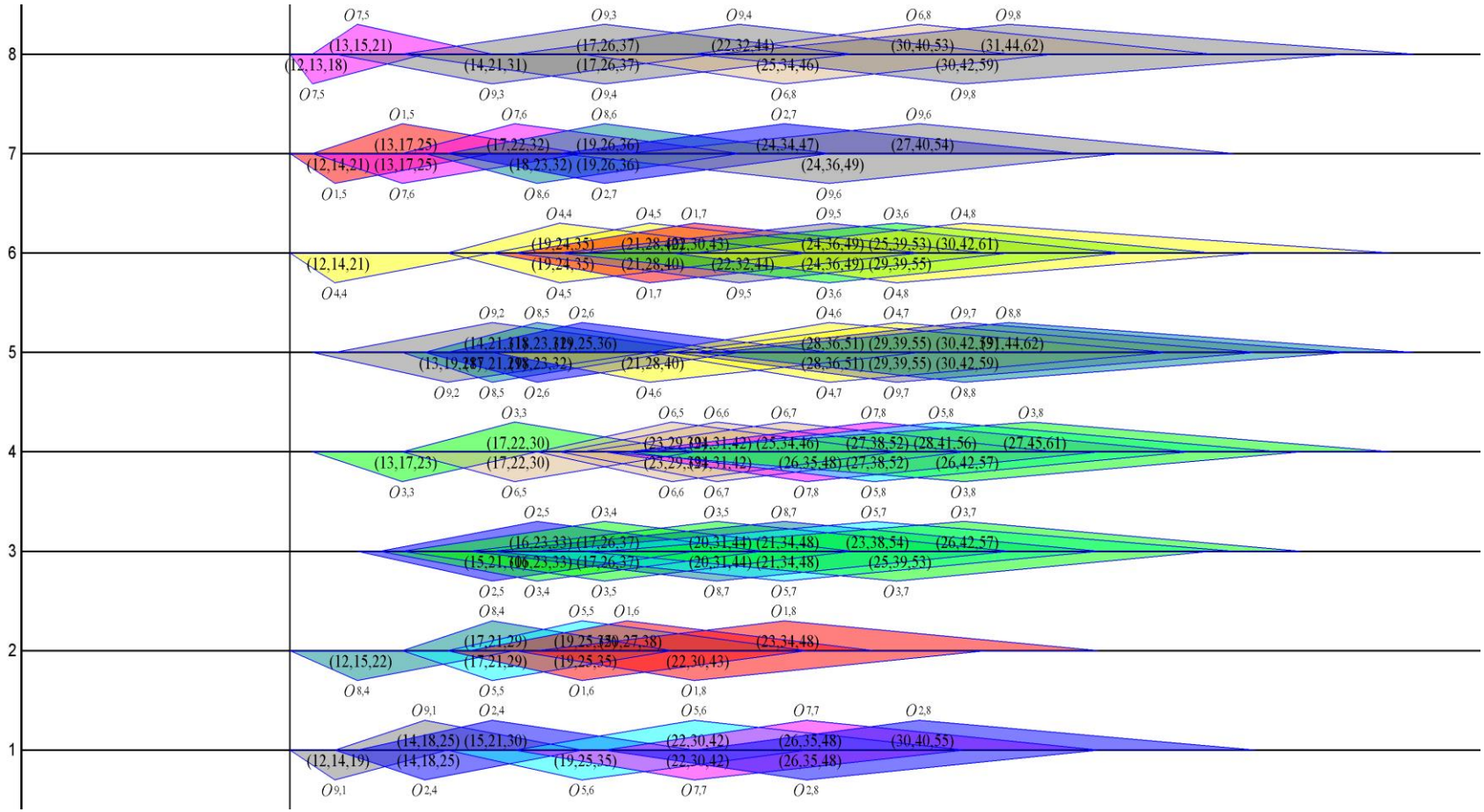

Fig. 9 - 5 First rescheduling Gantt chart with best maximum fuzzy completion time

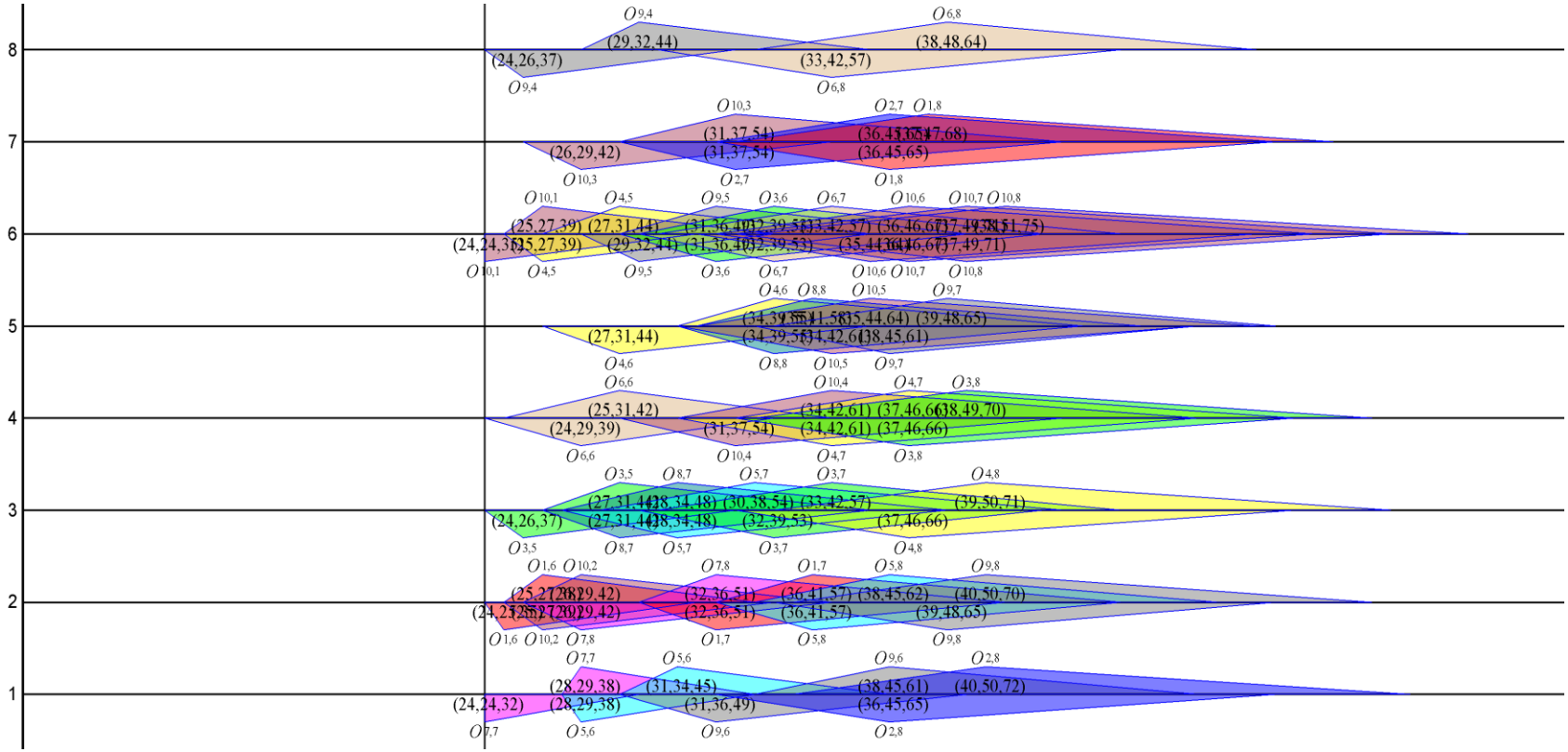

Fig. 9 - 6 Second rescheduling Gantt chart with best maximum fuzzy completion time

9.4.3 Maximum fuzzy machine workload objective

In both scheduling and rescheduling stages, we calculate the maximum machine workload for current scheduling stage. For maximum fuzzy machine workload objective, we compare FABC algorithm to five heuristics, MinEnd, GS, LS, MReW and MaxReO. To show FABC algorithm performance, the following formula is employed:

$$\mathrm{Imp(i)} = \frac{(t_1^i+2t_2^i+t_3^i)/4-(t_1^0+2t_2^0+t_3^0)/4}{(t_1^0+2t_2^0+t_3^0)/4} \times 100\% \quad (9\text{-}6)$$

where t_1^i, t_2^i and t_3^i are the results of i^{th} heuristic represented by TFN; t_1^0, t_2^0 and t_3^0 are the result of TFABC algorithm. $(t_1^i + 2t_2^i + t_3^i)/4$ is the first criterion to compare two TFNs. Because the results by TFABC algorithm are better than those by six heuristics, the first criterion is enough to compare them. It is clear that the larger the $\mathrm{Imp(i)}$ is, more competitive TFABC algorithm than i^{th} heuristic.

For eight instances, the maximum fuzzy machine workload results of TFABC and five heuristics are evaluated. The $Imp(i)$ results of five heuristics to TFABC algorithm are also shown in statistics. All the results are shown in Table 9 - 10 to Table 9 - 17. Each table shows the maximum machine workload of one instance in scheduling and rescheduling stages.

Table 9 - 10 shows the maximum machine workload results for instance 1and two times new job insertion. The first and second columns are compared algorithms and new job(s) insertion times. The value "0" in the second column means the scheduling results for instance 1 while the values "1" and "2" mean the two times rescheduling results after two times new job(s) insertion. It can be seen from Table 9 - 10 that TFABC algorithm can obtain the better results for average and worst measures than those by five compared heuristics in both scheduling stage and two times rescheduling stages. For the best measure, TFABC can just find competitive results in scheduling stage. The MReW heuristic get the most competitive results in the first rescheduling stage while MinEnd, GS and MReO heuristics obtain the most competitive results in the second rescheduling stage.

It can be seen from Table 9 - 11 to Table 9 - 17 that the TFABC algorithm can obtain the most competitive results for average, best and worst measures in scheduling and rescheduling stages. For instance 2, the average values of five heuristics are larger than those of FABC by at least 29.8%, 6.4% and 23.2% in scheduling and two times rescheduling stages. The corresponding values of best and worst measures are by at least 20.2%, 5.7% and 13.4% and at least 45.4%, 35.5% and 37.1% respectively. For instance 3 to instance 8 and 15 times new job insertion, We can get the similar comparison conclusion from Table 9 - 12 to Table 9 - 17.

The CPU times of five heuristics for eight instances and new job insertion are very small and can be cancelled. For eight instances and 19 times new job(s) insertion, the TFABC algorithm's average CPU

times in 30 repeat are shown in Table 9 - 18. The maximum CPU time is less than 2 minutes. Compared to the operation processing time in practical shop floor, the CPU times of TFABC algorithm are very limited.

For maximum fuzzy machine workload objective, Fig. 9 - 7 to Fig. 9 - 9 show TFABC algorithm scheduling results for instance Reman2 and two times new job insertions. Fig. 9 - 7 shows the scheduling results of instance Reman2 and the maximum fuzzy machine workload is (13, 30, 41) on machine 4. Fig. 9 - 8 shows the rescheduling results after new job insertion at time 12. After rescheduling, the maximum machine workload after new job 9 insertion is (17, 26, 37) on machine 1. Fig. 9 - 9 shows the Gantt chart after the second time new job insertion on time 24. The rescheduling result of maximum fuzzy machine workload is (15, 24, 34) on machine 7.

In summary, TFABC can obtain most competitive maximum machine workload results for all instances except the two times new job insertion in instance 1. The comparisons show that TFABC algorithm can obtain satisfactory results for real instances in remanufacturing industry and the TFABC algorithm needs very limited CPU time to get scheduling and rescheduling maximum machine workload results.

Table 9 - 10 Maximum fuzzy machine workload results of instance 1

Algorithm	Insertion times	Average			Imp (%)	Best			Imp (%)	Worst			Imp (%)
TFABC	0	13.0	22.0	30.0		13	22	30		13	22	30	
MinEnd		17.9	28.7	38.6	31.0	14	24	33	9.2	21	35	47	58.6
GS		17.0	27.4	36.7	24.6	14	24	33	9.2	21	31	41	42.5
LS		15.0	28.0	39.0	26.4	15	28	39	26.4	15	28	39	26.4
MReO		22.5	35.1	47.8	61.5	14	24	34	10.3	53	71	90	227.6
MReW		21.2	33.8	46.2	55.2	17	26	35	19.5	41	56	75	162.1
													0.0
TFABC	1	10.0	18.0	26.0		10	18	26		10	18	26	0.0
MinEnd		15.2	23.0	30.9	28.1	12	18	25	1.4	24	33	43	84.7
GS		13.0	20.3	26.7	11.6	13	19	24	4.2	16	24	31	31.9
LS		16.0	31.0	43.0	68.1	16	31	43	68.1	16	31	43	68.1
MReO		15.5	24.3	32.7	34.4	10	19	27	4.2	30	39	53	123.6
MReW		18.3	28.8	39.5	60.3	10	16	23	-9.7	42	56	71	212.5
													0.0
TFABC	2	9.0	19.0	27.0		9	19	27		9	19	27	0.0
MinEnd		12.7	20.5	28.2	10.6	10	17	25	-6.8	21	27	35	48.6
GS		12.8	19.4	26.4	5.4	10	17	25	-6.8	19	24	30	31.1
LS		12.0	24.0	34.0	27.0	12	24	34	27.0	12	24	34	27.0
MReO		16.4	25.8	35.5	40.0	10	17	25	-6.8	32	42	53	128.4
MReW		17.9	28.0	38.4	51.7	11	18	26	-1.4	39	48	68	174.3

Table 9 - 11 Maximum fuzzy machine workload results of instance 2

Algorithm	Insertion times	Average			Imp (%)	Best			Imp (%)	Worst			Imp (%)
TFABC	0	16.5	29.5	41.1		13	30	41		17	30	42	
MinEnd		28.3	44.9	60.7	53.3	23	37	49	28.1	37	56	78	90.8
GS		23.4	38.1	51.9	29.8	23	35	44	20.2	27	43	60	45.4
LS		31.0	45.0	65.0	59.5	31	45	65	63.2	31	45	65	56.3
MReO		33.5	54.7	75.1	86.8	27	40	54	41.2	56	80	107	171.4
MReW		37.7	57.7	79.6	99.6	22	36	49	25.4	67	91	126	215.1
TFABC	1	18.6	30.3	41.7		17	26	37		19	27	34	
MinEnd		23.3	37.4	50.3	22.9	19	31	40	14.2	32	48	64	79.4
GS		20.7	32.0	43.9	6.4	21	28	35	5.7	20	38	52	38.3
LS		23.0	35.0	52.0	20.0	23	35	52	36.8	23	35	52	35.5
MReO		33.3	50.5	68.9	68.2	17	31	44	16.0	65	86	113	227.1
MReW		29.9	49.1	66.9	61.5	20	36	52	35.8	55	80	107	200.9
TFABC	2	15.0	24.0	34.0		15	24	34		15	24	34	
MinEnd		22.1	34.9	47.4	43.6	18	29	42	21.6	29	46	64	90.7
GS		20.0	29.7	40.0	23.2	16	28	38	13.4	20	33	47	37.1
LS		27.0	39.0	57.0	67.0	27	39	57	67.0	27	39	57	67.0
MReO		29.6	46.8	64.4	93.4	20	32	44	32.0	50	78	108	223.7
MReW		31.3	51.3	70.6	110.7	21	35	48	43.3	64	93	122	283.5

Table 9 - 12 Maximum fuzzy machine workload results of instance 3

Algorithm	Insertion times	Average			Imp (%)	Best			Imp (%)	Worst			Imp (%)
TFABC	0	32.3	53.3	73.1		34	53	71		36	54	70	
MinEnd		46.0	72.5	97.2	36.0	40	60	83	15.2	55	87	115	60.7
GS		41.7	67.1	90.4	25.7	40	61	83	16.1	49	74	96	36.9
LS		51.0	76.0	102.0	43.9	51	76	102	44.5	51	76	102	42.5
MReO		55.3	85.0	113.1	59.7	37	65	89	21.3	95	136	175	153.3
MReW		54.3	84.1	112.8	58.2	40	62	85	18.0	99	130	169	146.7
TFABC	1	26.3	45.2	63.4		27	45	62		26	46	63	
MinEnd		39.6	63.6	85.2	40.0	33	54	70	17.9	50	74	93	60.8
GS		38.6	59.6	81.4	32.9	33	52	73	17.3	43	66	94	48.6
LS		47.0	73.0	100.0	62.8	47	73	100	63.7	47	73	100	61.9
MReO		49.0	75.5	100.9	67.2	30	52	71	14.5	89	121	154	168.0
MReW		47.8	77.1	103.2	69.5	37	57	79	28.5	91	127	165	181.8
TFABC	2	28.7	45.2	61.3		31	45	59		25	46	65	
MinEnd		40.5	63.0	83.6	38.5	36	55	72	21.1	47	73	97	59.3
GS		37.7	57.4	78.2	27.8	30	51	73	13.9	44	68	96	51.6
LS		41.0	67.0	90.0	46.8	41	67	90	47.2	41	67	90	45.6

MReO		47.7	73.3	99.0	62.5	31	54	79	21.1	94	130	166	185.7
MReW		53.9	80.7	107.7	79.0	37	58	81	30.0	94	127	164	181.3
TFABC	3	25.9	40.2	54.5		22	40	57		27	41	54	
MinEnd		38.2	59.9	79.5	47.7	34	50	63	23.9	49	71	94	74.8
GS		34.4	51.7	68.6	28.4	30	48	67	21.4	35	57	78	39.3
LS		38.0	59.0	77.0	44.9	38	59	77	46.5	38	59	77	42.9
MReO		47.4	72.6	96.5	79.7	34	52	68	29.6	80	116	147	181.6
MReW		48.1	72.2	96.2	79.5	32	51	70	28.3	83	116	154	187.7

Table 9 - 13 Maximum fuzzy machine workload results of instance 4

Algorithm	Insertion times	Average			Imp (%)	Best			Imp (%)	Worst			Imp (%)
TFABC	0	21.4	37.5	53.1		28	36	48		22	37	56	
MinEnd		36.6	54.6	74.8	47.5	27	48	68	29.1	50	74	101	96.7
GS		36.9	52.9	71.1	43.0	30	43	59	18.2	45	65	89	73.7
LS		46.0	72.0	103.0	96.0	46	72	103	98.0	46	72	103	92.8
MReO		47.0	68.8	93.7	86.2	35	50	69	37.8	91	120	159	222.4
MReW		49.3	72.8	99.5	96.9	32	52	76	43.2	91	120	162	224.3
TFABC	1	24.5	36.4	49.7		28	35	47		23	36	53	
MinEnd		37.1	54.2	73.9	49.2	35	47	63	32.4	45	65	87	77.0
GS		29.3	44.3	59.2	20.5	25	39	55	9.0	34	49	62	31.1
LS		44.0	67.0	96.0	86.3	44	67	96	89.0	44	67	96	85.1
MReO		55.4	81.5	111.6	124.4	42	53	71	51.0	69	109	153	197.3
MReW		53.8	78.1	105.7	114.6	34	49	67	37.2	116	157	207	330.4
TFABC	2	19.2	32.7	44.6		19	32	45		17	33	49	
MinEnd		34.1	51.8	67.9	59.0	25	41	57	28.1	39	60	79	80.3
GS		30.5	46.0	61.3	42.2	27	40	53	25.0	39	55	72	67.4
LS		46.0	71.0	99.0	122.0	46	71	99	124.2	46	71	99	117.4
MReO		55.6	80.7	109.0	152.1	44	60	78	89.1	95	134	178	309.8
MReW		44.1	70.5	95.7	117.3	28	52	67	55.5	79	106	147	231.8

Table 9 - 14 Maximum fuzzy machine workload results of instance 5

Algorithm	Insertion times	Average			Imp (%)	Best			Imp (%)	Worst			Imp (%)
TFABC	0	43.5	74.4	104.9		36	75	109		43	74	108	
MinEnd		63.7	102.9	140.9	38.1	59	92	121	23.4	73	110	153	49.2
GS		55.8	89.8	123.7	20.8	51	85	121	15.9	57	96	133	27.8
LS		72.0	101.0	137.0	38.3	72	101	137	39.3	72	101	137	37.5
MReO		86.4	136.3	184.2	82.8	61	104	147	41.0	133	186	251	152.8
MReW		82.8	133.7	181.1	78.7	64	98	137	34.6	140	202	267	171.2
TFABC	1	37.1	65.8	92.5		42	64	90		40	65	93	

MinEnd		57.8	93.0	128.1	42.4	50	83	113	26.5	71	109	149	66.5
GS		49.8	82.1	112.3	24.9	44	78	110	19.2	55	87	117	31.6
LS		56.0	92.0	123.0	38.9	56	92	123	39.6	56	92	123	38.0
MReO		87.8	132.9	178.7	103.7	51	96	137	46.2	135	181	240	180.2
MReW		92.2	141.5	189.8	116.2	62	103	137	55.8	129	192	261	194.3
TFABC	2	34.6	57.9	80.3		36	57	79		38	57	81	
MinEnd		47.8	75.5	102.5	30.6	45	69	93	20.5	49	82	115	40.8
GS		48.3	77.3	104.6	33.2	49	72	97	26.6	50	82	111	39.5
LS		53.0	95.0	121.0	57.7	53	95	121	59.0	53	95	121	56.2
MReO		68.5	109.1	146.5	87.7	52	84	112	45.0	121	164	219	186.7
MReW		86.3	133.1	179.4	130.5	57	88	115	52.0	133	184	240	218.0

Table 9 - 15 Maximum fuzzy machine workload results of instance 6

Algorithm	Insertion times	Average			Imp (%)	Best			Imp (%)	Worst			Imp (%)
TFABC	0	34.5	62.0	86.2		30	62	87		38	63	85	
MinEnd		52.2	86.7	118.1	40.4	50	78	106	29.5	60	101	139	61.0
GS		45.9	76.6	104.5	24.0	40	72	99	17.4	54	84	111	33.7
LS		56.0	88.0	119.0	43.4	56	88	119	45.6	56	88	119	41.0
MReO		74.0	113.8	153.1	85.8	51	85	112	38.2	109	160	217	159.4
MReW		74.5	116.2	156.8	89.5	51	80	111	33.6	134	181	237	194.4
TFABC	1	30.8	57.4	81.1		29	58	80		32	58	81	
MinEnd		48.8	79.7	110.0	40.3	40	71	102	26.2	62	100	134	72.9
GS		41.6	71.0	97.5	24.0	38	67	89	16.0	47	77	105	33.6
LS		47.0	85.0	121.0	49.1	47	85	121	50.2	47	85	121	47.6
MReO		71.3	112.7	153.6	98.6	48	83	117	47.1	138	191	249	235.8
MReW		73.7	117.2	159.7	106.2	54	84	109	47.1	127	183	242	221.0
TFABC	2	28.5	54.1	76.7		30	54	74		30	55	75	
MinEnd		44.1	72.5	99.7	35.2	38	62	85	16.5	57	82	112	54.9
GS		37.8	62.3	85.6	16.2	34	56	77	5.2	44	69	91	27.0
LS		51.0	88.0	124.0	64.5	51	88	124	65.6	51	88	124	63.3
MReO		70.6	108.4	146.9	103.5	42	72	101	35.4	112	145	189	174.9
MReW		76.0	121.3	166.1	127.1	50	98	134	79.2	110	155	204	190.2
TFABC	3	25.7	49.4	70.9		23	49	73		26	49	74	
MinEnd		38.1	66.8	92.2	35.1	31	60	88	23.2	41	75	103	48.5
GS		34.5	59.7	82.9	21.2	34	57	78	16.5	39	65	87	29.3
LS		53.0	90.0	131.0	86.3	53	90	131	87.6	53	90	131	83.8
MReO		53.3	90.5	128.1	85.5	46	70	98	46.4	84	126	168	154.5
MReW		76.6	122.6	167.1	150.2	40	85	121	70.6	119	170	222	243.9

Table 9 - 16 Maximum fuzzy machine workload results of instance 7

Algorithm	Insertion times	Average			Imp (%)	Best			Imp (%)	Worst			Imp (%)
TFABC	0	50.5	97.0	138.5		49	97	137		50	97	145	
MinEnd		74.6	130.3	183.1	35.3	68	119	165	23.9	76	143	201	44.7
GS		68.1	117.8	165.7	22.5	64	112	159	17.6	73	123	172	26.2
LS		66.0	122.0	172.0	25.8	66	122	172	26.8	66	122	172	23.9
MReO		127.1	201.0	275.5	110.1	81	143	193	47.4	205	307	399	213.1
MReW		102.9	177.7	245.5	83.7	78	142	200	47.9	165	236	310	143.4
TFABC	1	49.9	95.6	136.2		46	94	140		48	96	141	
MinEnd		76.4	129.1	178.7	36.0	68	121	165	27.0	87	140	192	46.7
GS		65.5	116.1	163.2	22.1	56	110	164	17.6	60	122	178	26.5
LS		70.0	123.0	163.0	26.9	70	123	163	28.1	70	123	163	25.7
MReO		111.1	188.3	262.3	98.8	75	144	205	51.9	166	247	337	161.7
MReW		108.0	180.3	247.9	89.9	71	137	201	46.0	179	251	333	166.1
TFABC	2	42.4	80.5	115.2		42	80	112		47	80	118	
MinEnd		65.7	112.6	156.5	40.4	63	106	144	33.4	74	119	169	48.0
GS		63.0	104.6	146.1	31.2	51	100	141	24.8	72	110	153	36.9
LS		63.0	118.0	166.0	45.9	63	118	166	48.1	63	118	166	43.1
MReO		114.6	182.4	245.7	127.6	73	137	189	70.7	155	225	299	178.2
MReW		97.7	166.1	225.6	105.7	66	121	171	52.5	173	262	345	220.6

Table 9 - 17 Maximum fuzzy machine workload results of instance 8

Algorithm	Insertion times	Average			Imp (%)	Best			Imp (%)	Worst			Imp (%)
TFABC	0	40.6	75.3	106.8		42	74	101		44	77	107	
MinEnd		58.2	98.9	136.7	31.8	55	90	122	22.7	69	109	148	42.6
GS		50.2	86.4	120.1	15.1	48	84	114	13.4	57	91	125	19.3
LS		62.0	101.0	139.0	35.3	62	101	139	38.5	62	101	139	32.1
MReO		87.9	145.4	198.9	93.9	58	109	154	47.8	161	233	315	208.9
MReW		99.7	158.2	212.8	111.1	61	115	153	52.6	148	203	266	168.9
TFABC	1	38.3	72.7	103.9		38	70	102		37	74	109	
MinEnd		54.7	93.4	128.4	28.7	50	85	116	20.0	52	104	144	37.4
GS		47.7	84.7	117.6	16.4	44	78	119	13.9	52	90	130	23.1
LS		61.0	107.0	149.0	47.5	61	107	149	51.4	61	107	149	44.2
MReO		96.3	154.1	208.9	113.3	60	111	151	54.6	147	208	272	184.0
MReW		103.3	167.3	227.6	131.5	63	115	166	63.9	171	250	323	238.1
TFABC	2	36.8	69.7	101.0		37	69	96		37	71	103	
MinEnd		56.3	92.9	127.4	33.2	54	88	116	27.7	63	102	138	43.6
GS		49.9	84.1	115.2	20.2	44	81	109	16.2	56	89	121	25.9
LS		51.0	88.0	121.0	25.5	51	88	121	28.4	51	88	121	23.4

MReO		91.8	148.3	202.3	113.0	62	103	135	48.7	150	222	299	216.7
MReW		94.5	160.7	221.1	129.7	57	122	175	75.6	156	222	295	217.4
TFABC	3	34.9	64.7	91.7		38	63	89		38	66	92	
MinEnd		49.7	85.0	117.3	31.6	45	80	106	22.9	53	93	136	43.1
GS		45.5	77.6	107.0	20.1	40	74	101	14.2	48	82	114	24.4
LS		47.0	81.0	111.0	4.0	47	81	111	26.5	47	81	111	22.1
MReO		78.0	131.7	181.5	70.0	47	104	151	60.5	127	182	249	182.4
MReW		93.8	161.3	223.7	108.1	58	115	157	75.9	178	256	334	290.8

Table 9 - 18 TFABC algorithm's average CPU times for all instances and new job insertions

Instance	Insertion times	CPU Time (s)	Instance	Insertion times	CPU Time (s)	Instance	Insertion times	CPU Time (s)
instance 1	0	0.474	instance 4	0	3.302	instance 7	0	66.605
	1	0.403		1	3.024		1	60.380
	2	0.400		2	2.699		2	43.889
instance 2	0	1.748	instance 5	0	25.266	instance 8	0	107.540
	1	1.443		1	20.260		1	98.295
	2	1.105		2	15.394		2	86.357
instance 3	0	2.222	instance 6	0	29.878		3	70.815
	1	1.767		1	27.350			
	2	1.628		2	24.690			
	3	1.364		3	22.181			

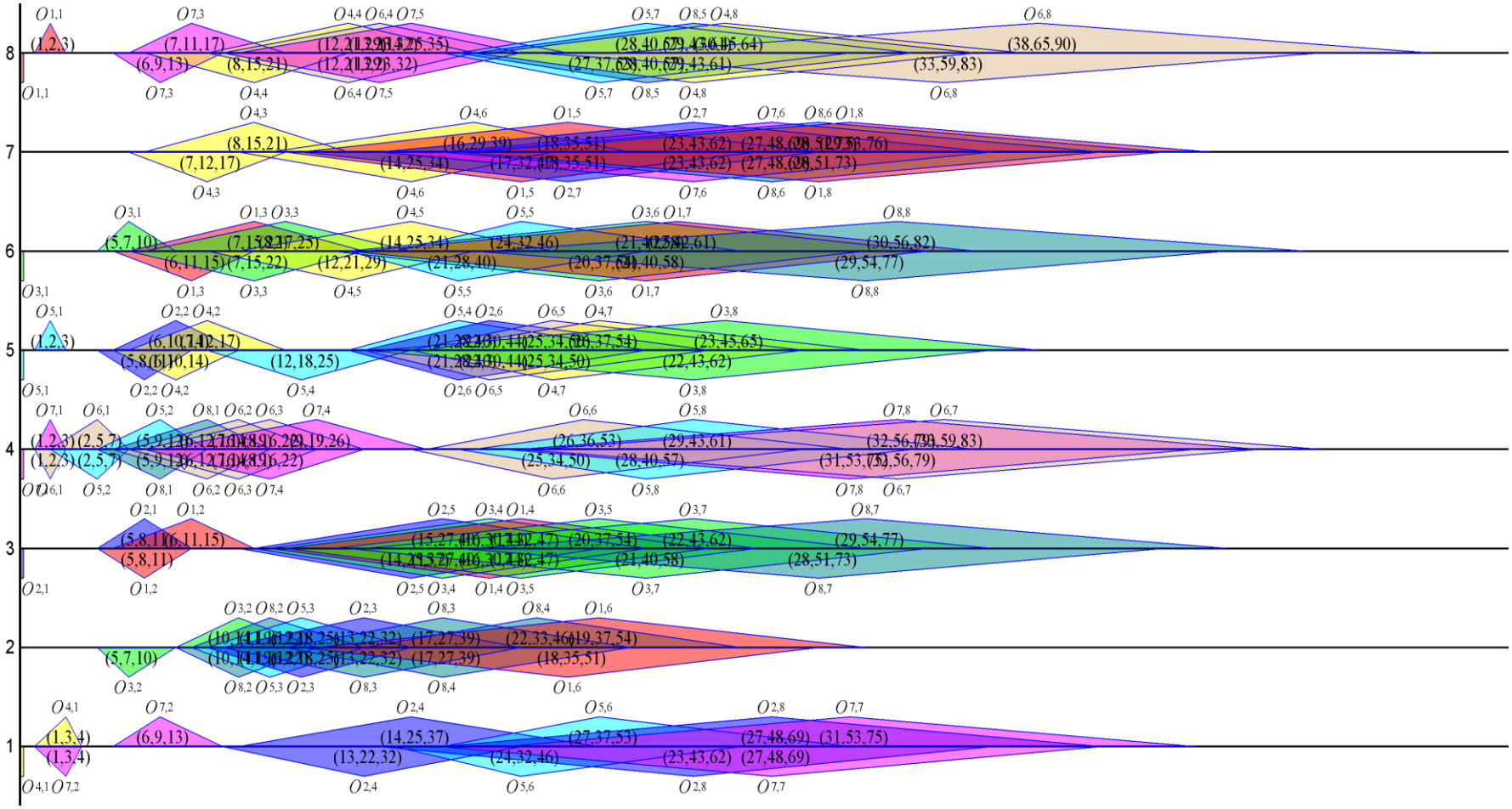

Fig. 9 - 7 Gantt chart of instance Reman 2 with minimum machine workload

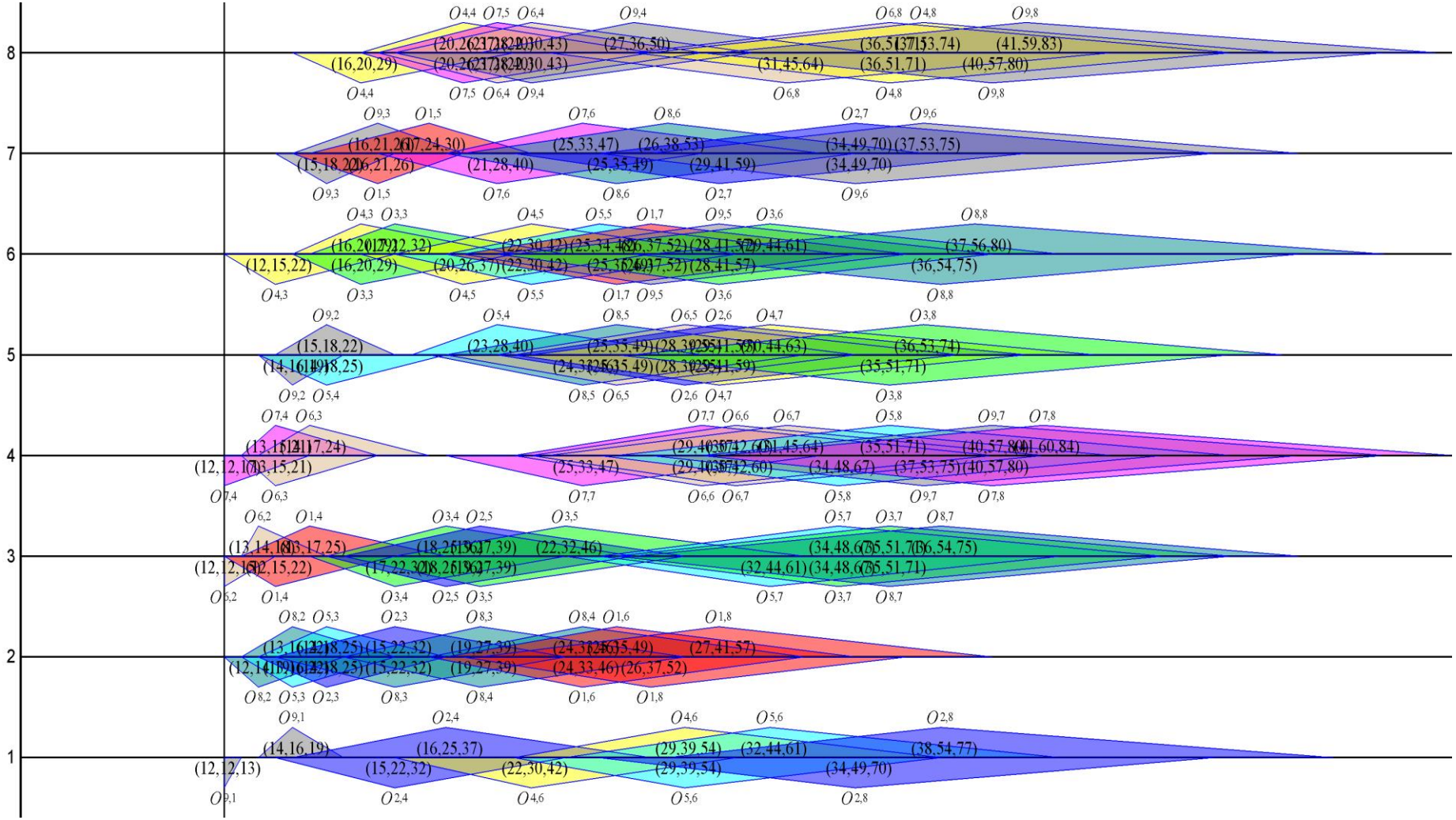

Fig. 9 - 8 First rescheduling Gantt chart with best fuzzy machine workload

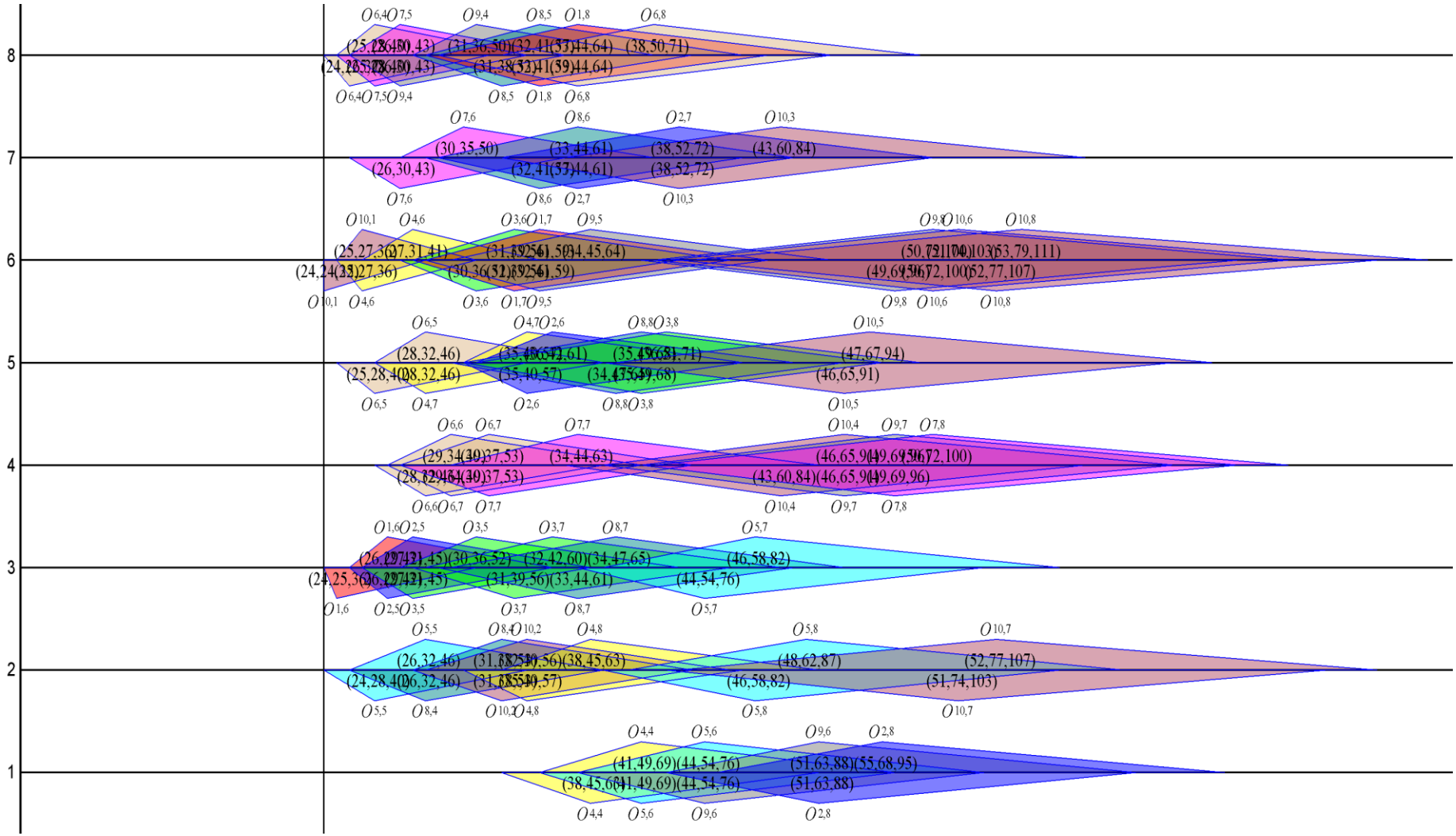

Fig. 9 - 9 Second rescheduling Gantt chart with best fuzzy machine workload

9.5 Conclusions

In this Chapter, a two-stage fuzzy artificial bee colony (TFABC) algorithm was proposed to solve flexible job shop scheduling problem with fuzzy processing time and new job(s) insertion constraints. The objectives were to minimize the maximum fuzzy completion time and the maximum fuzzy machine workload. The MinEnd heuristic and global minimum processing time heuristic were employed to initialize population. To test the performance, the TFABC was compared to five heuristics for solving eight instances from remanufacturing industry. The results and comparisons demonstrated the performance of TFABC algorithm. The TFABC algorithm improved the results of heuristics obviously. TFABC algorithm cost very low computer CPU time. In addition, several fuzzy Gantt charts of best solution were shown for the maximum fuzzy completion time and the maximum fuzzy machine workload objectives.

Chapter 10

Conclusions and Future Work

10.1 Conclusions

This thesis researched on multi-objective multi-constraint flexible job shop scheduling problem (FJSSP). Two scheduling related remanufacturing characteristics, namely uncertain processing time and new job insertion, were modeled as two constraints of FJSSP. The scheduling objectives were to minimize maximum completion time (makespan), maximum machine workload, mean of earliness and tardiness. The major contributions of this thesis are shown as follows.

- Two strategies, most probable processing time and fuzzy processing time, were employed to represent the uncertain processing time of operations in remanufacturing industry. One operation had two processing times. The first one was the most probable processing time and the second one was a fuzzy processing time that was represented by a triangle fuzzy number. Rescheduling was executed to deal with new job(s) insertion constraint.
- To solve general FJSSP, a discrete harmony search algorithm was proposed. In the DHS algorithm, several heuristics were used to initialize the population. Dynamic grouping, local search methods were developed to improve the algorithm performance. To test the performance, the DHS algorithm was employed to solve benchmark instances and was compared to several existing competitive algorithms.
- To solve FJSSP with most probable processing time and new job insertion constraints, we proposed a two-stage artificial bee colony algorithm. Three rescheduling strategies were proposed for rescheduling when new job arrived and was inserted into existing scheduling solution. The

three rescheduling strategies were evaluated and compared. To test the performance of the two-stage ABC algorithm, the scheduling and rescheduling results were compared to other heuristics and composite heuristics.

- A two-stage DHS (TDHS) algorithm was proposed to solve FJSSP with two constraints, processing time increasing and new job insertion. Two real remanufacturing instances were tested using the TDHS algorithm. Rescheduling was executed for both of most probable processing time increasing and new job insertion. The scheduling and rescheduling results could satisfy practical remanufacturing process.
- To solve FJSSP with fuzzy processing time, a fuzzy DHS (FDHS) algorithm was proposed. One heuristic, named MinEnd, was proposed to improve the quality of initial population. To evaluate performance, the FDHS algorithm was compared to five existing competitive algorithms for solving benchmark instances. Eight instances from remanufacturing industry were solved by the FDHS algorithm. The results were satisfactory for the practical process in the shop floor.
- To solve FJSSP with fuzzy processing time and new job insertion constraints, we proposed a fuzzy ABC (FABC) algorithm and a two-stage fuzzy ABC algorithm. The objectives were to minimize the maximum fuzzy completion time and maximum fuzzy machine workload. The FABC algorithm was compared to FDHS algorithm and five heuristics for solving eight remanufacturing instances. The two-stage FABC algorithm was compared to five heuristics for solving eight instances with new job insertion.

In summary, this thesis focused on scheduling problem in remanufacturing industry. The problem was modeled as FJSSP with two constraints. Two meta-heuristics, harmony search algorithm and artificial bee colony algorithm were employed and improved to solve the FJSSP with remanufacturing constraints. Benchmark instances and real instances from remanufacturing were solved. The experiments and comparisons showed the effectiveness and efficiency of proposed algorithms. The remanufacturing scheduling problem was solved satisfactorily. The proposed algorithms had been integrated to existing commercial planning and scheduling systems to improve the scheduling results.

10.2 Future work

Flexible job shop scheduling problem (FJSSP) is widespread in many fields. In future, we will focus on the following topics:

- We will research on more effective and efficient ensemble heuristics and meta-heuristics to solve the scheduling and rescheduling problem in remanufacturing industry. We will improve the convergence capability of the proposed DHS and ABC algorithms by considering additional local

search methods. These algorithms will be used to solve dynamic flexible job shop scheduling problems and other classes of scheduling problems.

- We will address more scheduling objectives in remanufacturing environment, for example cost related objectives and a hybrid of different type's objectives. Weighted multi-objective and Pareto-based multi-objective will be considered in different scheduling and rescheduling problems.
- More general practical constraints will be considered in future research, for example, machine setup time, machine breakdown, maintenance activity, limited resources, overlapping operations, sequence-dependent setup and transportation times and so on.
- We will use our scheduling algorithm to solve more scheduling problems with practical constraints in different fields. We will co-operate scheduling solution for flexible job shop scheduling problem with processing time uncertainty, in addition to new job insertion. We will also consider other meta-heuristics and propose new heuristics for more scheduling problems in remanufacturing industry.
- Beside harmony search and artificial bee colony algorithms, we will research on new meta-heuristics, for example, teaching learning based optimization. We will also employ the new meta-heuristics for solving job shop scheduling problem, flow shop scheduling problem and scheduling problem in different fields.

Author's Publications

Journal papers (Accepted and Published):

- K.Z. Gao, P.N. Suganthan, M.F. Tasgetiren, Q.K. Pan, et al. Effective ensembles of heuristics for scheduling flexible job shop problem with new job insertion. Computers & Industrial Engineering, 2015, 90: 107-117.
- K.Z. Gao, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren. An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time, International Journal of Production Research, 2015, 53(19): 5896-5911.
- K.Z. Gao, P.N. Suganthan, T.J. Chua, et al. A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion, Expert Systems with Applications, 2015, 42: 7652-7663.

1. **K.Z. Gao**, P.N. Suganthan, Q.K. Pan, T.J. Chua, et al. Pareto-based Grouping Discrete Harmony Search Algorithm for Multi-objective Flexible Job Shop Scheduling, Information Sciences, 2014, 289: 76-90.
2. **K.Z. Gao**, P.N. Suganthan, Q.K. Pan, T.J. Chua, et al. Discrete Harmony Search Algorithm for Flexible Job Shop Scheduling Problem with Weighted Combination of Multiple Objectives, Journal of Intelligence Manufacturing, Published online.
3. **K.Z. Gao**, P.N. Suganthan, T.J. Chua, Discrete harmony search algorithm for the disassembly scheduling in remanufacturing engineering, Applied Mechanics and Materials, 236-237: 169-174, 2012.

Conference papers:

1. **K.Z. Gao,** P.N. Suganthan, T.J. Chua, et al. Hybrid discrete harmony search algorithm for scheduling re-processing problem in remanufacturing, Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference (GECCO), page 1261-1268, Amsterdam, Netherlands, July 06-10, 2013.
2. **Kai Zhou Gao**, Ponnuthurai Nagaratnam Suganthan, Tay Jin Chua, Discrete harmony search algorithm for the dynamic FJSSP in remanufacturing engineering, Lecture Notes in Computer Science, 7677: 9-16, 2012.
3. **Kai Zhou Gao**, Ponnuthurai Nagaratnam Suganthan, Tay Jin Chua, Pareto-based discrete harmony search algorithm for flexible job shop scheduling, Proceeding of the 12th conference on

Intelligent Systems, Design and Applications (ISDA), page 953-956, Kochi, India, Nov 27-29, 2012.

Submitted papers:

1. **Kai Zhou Gao**, Ponnuthurai Nagaratnam Suganthan, Tay Jin Chua, et al. An improved artificial bee colony algorithm for multi-objective flexible job shop scheduling problem with fuzzy processing time. Submitted to "Applied Soft Computing" of the Journal (Near accepted).

Bibliography

[1] Lund R. I., Remanufacturing. Technology Review, 87 (1984) 18-23.

[2] Krupp J. A., Core obsolescence forecasting in remanufacturing. Production and Inventory Management Journal, 33 (1992) 12-17.

[3] Krupp J. A., Structuring bills of material for automotive remanufacturing. Production and Inventory Management Journal, 34 (1993) 46-52.

[4] DePuy G.W., Usher J.S., Walker R. L., Taylor G.D., Production planning for remanufactured products. Production Planning & Control, 18 (2007) 573-583.

[5] Guide J., Souza G.C.,Van L., Performance of static priority rules for shared facilities in a remanufacturing shop with disassembly and reassembly. European Journal of Operational Research, 164 (2005) 341-353.

[6] Guide J., V D.R., Srivastava.R., Spencer M.S., An evaluation of capacity planning techniques in a remanufacturing environment. International Journal of Production Research, 35 (1997) 67-82.

[7] Guide J., V D.R., Srivastava R., Kraus M.E., Product structure complexity and scheduling of operations in recoverable manufacturing. International Journal of Production Research, 35 (1997) 3179-3200.

[8] Teunter R., Tang O., Kaparis K., Heuristics for the economic lot scheduling problem with returns. International Journal of Production Economics, 118 (2009) 323-330.

[9] Grubbström R.W., Tang O., Optimal production opportunities in a remanufacturing system. International Journal of Production Research, 44 (2006) 3953-3966.

[10] Daniel V. R., Guide J., Production planning and control for remanufacturing: industry practice and research needs. Journal of Operations Management, 18 (2000) 467-483.

[11] Muris L. J., Moacir G. F., Production planning and control for remanufacturing: literature review and analysis. Production Planning & Control, 23 (2012) 419-435.

[12] Pinedo M, Scheduling: theory, algorithms, and systems. Prentice-Hall, Englewood cliffs, 2012.

[13]Garey M.R., Johnson D.S., Sethi R., The complexity of flow hop and job shop scheduling, Mathematics of Operations Research, 1 (1976) 117–129.

[14]Kacem I., Hammadi S., Borne P., Approach by localization and multi-objective evolutionary optimization for flexible job shop scheduling problems, IEEE transaction on system, Man, and Cybernetics, 32 (2002) 1-13.

[15]Brucker P., Schlie R., Job-shop scheduling with multi-purpose machines, Computing, 45(1990) 369-375.

[16]Pezzella F., Morganti G., Ciaschetti G., A genetic algorithm for the flexible job-shop scheduling problem, Computers and Operations Research, 35 (2008) 3202–3212.

[17] Chu C, Portmann M C, Proth J M. A splitting-up approach to simplify job-shop scheduling problems. International Journal of Production Research, 30 (1992) 859-870.

[18] Balas E. Machine scheduling via disjunctive graphs: an implicit enumeration algorithm. Operations Research, 17 (1969) 941-957.

[19] Smith W. E. Various optimizers for single stage production. Naval Research Logistics Quarterly, 3 (1956) 59-66.

[20] Panwalker S S, Iskander W. A survey of scheduling rules. Operations Research, 25 (1977) 45-61.

[21]Wang Y.M., Ong H., L. Yin, Qin K.D., A novel genetic algorithm for flexible job shop scheduling problems with machine discruption, International Journal of Advanced Manufacturing Technology, 68 (2013) 1317-1326.

[22]Xing L.N., Chen Y.W., Wang P. etc. al, A knowledge-based ant colony optimization for flexible job shop scheduling problems, Applied Soft Computing, 10 (2010) 888-896.

[23]Xia W.J., Wu Z.M., An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems, Computers and Industrial Engineering, 48 (2005) 409–425.

[24] Yuan Y., Xu H., Flexible job shop scheduling using hybrid differential evolution algorithms, Computers and Industrial Engineering, 65 (2013) 246–260.

[25]Gao K.Z., Suganthan P.N., Pan Q.K., etc.al, Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives, Journal of Intelligence Manufacturing, DOI: 10.1007/s10845-014-0869-8.

[26] Li J.Q., Pan Q.K., Gao K.Z., Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems, International Journal of Advanced Manufacturing Technology, 55 (2011) 1159-1169.

[27]Zhang G.H., Shao X.Y., Li P.G., Gao L., An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, Computers and Industrial Engineering, 56 (2009) 1309–1318.

[28] Gao J., Sun L., Gen M., A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems, Computers and Operations Research, 35 (2008) 2892–2907.

[29] Vilcot G., Billaut J.C., A tabu search algorithm for solving a multi-criteria flexible job shop scheduling problem, International Journal of Production Research, 49 (2011) 6963-6980.

[30] Gao J., Gen M., Sun L.Y., Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. Journal of Intelligent Manufacturing, 17 (2006) 493-507.

[31] Perez M. A. F., Raupp F. M. P., A newton-based heuristic algorithm for multi-objective flexible job shop scheduling problem. Journal of Intelligent Manufacturing, DOI: 10.1007/s10845-014-0872-0.

[32] Mati Y., Rezg N., Xie X., An integrated greedy heuristic for flexible job shop scheduling problem. Proceeding of IEEE International Conference on Systems, Man and Cybernetics, 2001, 2534-2539.

[33] Wang S. J., Zhou B. H., Xi L. F., A filtered-beam-search-based heuristic algorithm for flexible job-shop scheduling problem. International Journal of Production Research, 46 (2008) 3027-3058.

[34] Wang S. J., Yu J.B., An effective heuristic for flexible job-shop scheduling problem with maintenance activities. Computer and Industrial Engineering, 59 (2010) 436-447.

[35] Ziaee M., A heuristic algorithm for solving flexible job shop scheduling problem. International Journal of Advanced Manufacturing Technology, 71 (2014) 519-528.

[36] Ziaee M., A heuristic algorithm for the distributed and flexible job-shop scheduling problem. Journal Supercomputing, 67 (2014) 69-83.

[37] Clleja G., Pastor R., A dispatching algorithm for flexible job-shop scheduling with transfer batches: an industrial application, 25 (2014) 93-109.

[38] Chen J.C., Chen K.H., Wu J.J., Che CW, A study of the flexible job shop scheduling problem with parallel machines and reentrant process. International Journal of Advanced Manufacturing Technology, 39 (2008) 344-354.

[39] Ben H. A, Haouari M., Huguet M.J., Lopez P., Discrepancy search for the flexible job shop scheduling problem. Computer & Operations Research, 37 (2010) 2192-2201.

[40] Ozguven C., Yavuz Y., Ozbakir L, Mixed integer goal programming models for the flexi le job-shop scheduling problems with separable and non-separable sequence dependent setup times. Applied Mathematical Modelling, 36 (2012) 846-858.

[41] He W., Sun D.H., Scheduling flexible job shop problem subject to machine breakdown with route changing and right-shift strategies. International Journal of Advanced Manufacturing Technology, 66 (2013) 501-514.

[42] Sun D.H., He W., Zheng L.J., Liao X.Y., Scheduling flexible job shop problem subject to machine breakdown with game theory. International Journal of Production Research, 52 (2014) 3858-3876.

[43] Chan F.T.S., Wong T.C., Chan L.Y., Flexible job-shop scheduling problem under resource constraints. International o Journal of Production Research, 44 (2006) 2071-2089.

[44] Ho N.B., Tay J.C., Lai E.M.K., An effective architecture for learning and evolving flexible job-shop schedules. European Journal of Operational Research, 179 (2007) 316-333.

[45] De G. L., Pezzella F., An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. European Journal of Operational Research, 200 (2010) 395-408.

[46] Mati Y., Lahlou C., Dauzere-Peres S., Modelling and solving a practical flexible job shop scheduling problem with blocking constraints. International Journal of Production Research, 49 (2011) 2169-2182.

[47] AI-Hinai N., EIMekkawy T.Y., Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. International Journal of Production Research, 132 (2011) 279-291.

[48] Chen J. C., Wu C. C., Chen C. W. et al., Flexible job shop scheduling with parallel machines using genetic algorithm and grouping genetic algorithm. Experts systems with applications, 39 (2012) 10016-10021.

[49] Gholami M., Zandieh M., Integrating simulation algorithm to schedule a dynamic flexible job shop. Journal of Intelligent Manufacturing, 20 (2009) 481-498.

[50] Rahmati S.H.A., Zandieh M., Yazdani M., Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem. International Journal of Advanced Manufacturing Technology, 64 (2013) 915-932.

[51] Chiang T.C., Lin H.J., A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling. International Journal of Production Economics, 141 (2013) 87-98.

[52] Demir Y., Isleyen S.K., An effective genetic algorithm for flexible job shop scheduling with overlapping in operations. International Journal of Production Research, 52 (2014) 3905-3921.

[53] Zhang G.H., Gao L., Shi Y., An effective genetic algorithm for the flexible job shop scheduling problem. Experts systems with applications, 38 (2011) 3563-3573.

[54] Wang X.J. Gao L., Zhang C.Y., Shao X.Y., A multi-objective genetic algorithm based on immune and entropy pimple for flexible job shop scheduling problem. International Journal of Advanced Manufacturing Techonlogy, 51 (2010) 757-767.

[55] Lei D.M., Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling. Applied Soft Computing, 12 (2012) 2237-2245.

[56] Defersha F.M., Chen M.Y., A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups. International Journal of Advanced Manufacturing Technology, 49 (2009) 263-279.

[57] Rossi A, Dini G, Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimization method. Robotics and Computer-integrated Manufacturing, 23 (2007) 503-516.

[58] Yao B.Z. Yang C.Y., Hu J.J, et al., An improved ant colony optimization for flexible job shop scheduling problems. Advanced Science Letters, 4 (2011) 2127-2131.

[59] Luo D.L., Chen H.P., Wu S.X., Shi Y.X., Hybrid ant colony multi-objective optimization for flexible job shop scheduling problems. Journal of Intelligent Manufacturing, 11 (2010) 361-369.

[60] Rossi A., Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. International Journal of Production Economics, 153 (2014) 253-267.

[61] Huang R.H., Yang C.L., Cheng W.C., Flexible job shop scheduling with due window a two-pheromone ant colony approach. International Journal of Production Economics, 141 (2013) 685-697.

[62] Boukef H., Benrejeb M., Borne P., Flexible job-shop scheduling problems resolution inspired from particle swarm optimization. Studies in Informatics and Control, 17 (2008) 241-252.

[63] Liu H.B., Abraham A., Wang Z.W., A multi-swarm approach to multi-objective flexible job shop scheduling problems. Fundamental Informaticae, 95 (2009) 465-489.

[64] Moslehi G., Mahnam M., A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. International Journal of Production Economic, 129 (2011) 14-22.

[65] Nourali S., Imanipour N., A particle swarm optimization-based algorithm for flexible job shop scheduling problem with sequence dependent setup times. Scientia Iranica, 21 (2014) 1021-1033.

[66] Yuan Y., Xu H., Flexible job shop scheduling using hybrid differential evolution algorithms. Computes and Industrial Engineering, 65 (2013) 246-260.

[67] Xu X.L., Li L., Fan L.X. Zhang J., et al., Hybrid discrete differential evolution algorithm for lot splitting with capacity constraints in flexible job scheduling. Mathematical Problems in Engineering, 2013, article ID: 986218, 10pages.

[68] Fattahi P., Jolai F., Arkat J., Flexible job shop scheduling with overlapping in operations. Applied Mathematical Modelling, 33 (2009) 3076-3087.

[69] Logendran R., Sonthinen A., ATabu search-based approach for scheduling job-shop type flexible manufacturing systems. Journal of Operational Research Society, 48 (1997) 264-277.

[70] Saidi-Mehrabad M., Fattahi P., Flexible job shop scheduling with tabu search algorithms. International Journal of Advanced Manufacturing Technology, 32 (2007) 563-570.

[71] Li J.Q., Pan Q.K., Liang Y.C., An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. Computers & Industrial Engineering, 59 (2010) 647-662.

[72] Li J.Q., Pan Q.K., Suganthan P.N., Chua T.J., A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. International Journal of Advanced Manufacturing Technology, 52 (2001) 683-697.

[73] Jia S., Hu Z.H., Path-relinkinng tabu search for the multi-objective flexible job shop scheduling problem. Computers & Operations Research, 47 (2014) 11-26.

[74] Eshlaghy A.T., Sheibatolhamdy S.A., Scheduling in flexible job-shop manufacturing systems by improved tabu search. African Journal of Business Management, 5 (2011) 4863-4872.

[75] Wang L., Wang S.Y., Xu Y., Zhou G., A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. Comupter & Industrial Engineering, 62 (2012) 917-926.

[76] Wang L., Wang S.Y., Liu M., A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem. International Journal of Production Research, 51 (2013) 3574-3592.

[77] Wang S.Y., Wang L., Xu Y., Liu M., An effective estimation of distribution algorithm for the flexible job-shop scheduling problem with fuzzy processing time. International Jouornal of Production Research, 51 (2013) 3778-3793.

[78] Amiri M., Zandieh M., Yazdani M., Bagheri A., A variable neighborhood search algorithm for the flexible job-shop scheduling problem. International Journal of Production Research, 48 (2010) 5671-5689.

[79] Yazdani M., Amiri M., Zandieh M., Flexible job-shop scheduling with parallel variable neighborhood search algorithm. Experts systems with Applications, 37 (2010) 678-687.

[80] Bagheri A., Zandieh M., Bi-criterial flexible job-shop scheduling with sequence-dependent setup times-variable neighborhood search approach. Journal of Manufacturing Systems, 30 (2011) 8-15.

[81] Karimi H., Rahmati S.H.A., Zandieh M., An effective knowledge-based algorithm for the flexible job shop scheduling problem. Knowledge-based Systems, 36 (2012) 236-244.

[82] Lei D.M., Guo X.P., Variable neighborhood search for dual-resource constrained flexible job shop scheduling. International Journal of Production Research, 52 (2014) 2519-2529.

[83] Tanev I.T., Uozumi T., Morotome Y., Hybrid evolutionary algorithm-based real-wrold flexible job shop scheduling problem: application service provider approach. Applied Soft Computing, 5 (2004) 87-100.

[84] Fattahi P., Mehrabad M.S., Jolai F., Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. Journal of Intelligent Manufacturing, 18 (2007) 331-342.

[85] Frutos M., Olivera A.C., Tohme F., A memetic algorithm based on NSGAII scheme for the flexible job-shop scheduling. Annals of Operations Research, 181 (2010) 745-765.

[86] Roshanaei V., Azab A., EIMaraghy H., Mathematical modeling and a meta-heuristics for flexible job shop scheduling. International Journal of Production Research, 51 (2013) 6247-6274.

[87] Shahsavari-Pour N., Ghasemishabankareh B., A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling. Journal of Manufacturing Systems, 32 (2013) 771-780.

[88] Gao J., Gen M., Sun L.Y., Zhao X.H., A hybrid genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Computer and Industrial Engineering, 53 (2007) 149-162.

[89] Xing L.N., Chen Y.W., Yang K.W., Multi-population interactive co-evolutionary algorithm for flexible job shop scheduling problems. Computation Optimization and Applications, 48 (2011) 139-155.

[90] Zhang Q., Manier H., Manier M.A., A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing time. Computer and Operations Research, 39 (2012) 1713-1723.

[91] Barzegar B., Motameni H., Bozorgi H., Solving flexible job-shop scheduling problem using gravitational search algorithm and colored petri-net. Journal of Applied Mathematics, 2012, Article ID: 651310, 20 pages.

[92] Liouane N., Saad I., Hammadi S., Borne P., Ant system & local search optimization for flexible job shop scheduling production. International Journal of Computers Communications & Control, 2 (2007) 174-184.

[93] Xia W.J., Wu Z.M., An effective hybrid optimization approach for multi-objective flexible job shop scheduling problems. Computers and Industrial Engineering, 48 (2005) 409-425.

[94] Shao X.Y., Hybrid discrete particle swarm optimization for multi-objective flexible job shop scheduling problem. International Journal of Advanced Manufacturing Technology, 67 (2013) 2885-2901.

[95] Araghi M.E.T., Jolai F., Rabiee M., Incorporating learning effect and deterioration for solving a SDST flexible job-shop scheduling problem with a hybrid meta-heuristic approach. International Journal of Computer Integrated Manufacturing, 27 (2014) 733-746.

[96] Sadrzadeh A., Development of both the AIS and PSO for solving the flexible job shop scheduling problem. Arabian Journal for Science and Engineering, 38 (2013) 3593-3604.

[97] Dalfard V.M., Mohammadi G., Two meta-heuristic algorithms for solving multi-objective flexible job shop scheduling with parallel machine and maintenance constraints. Computers & Mathematics with Application, 64 (2012) 2111-2117.

[98]Wu Z.B., Weng M.X., Multi-agent scheduling method with earliness and tardiness objectives in flexible job shops, IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics, 35 (2005) 293-301.

[99] Yu X.F., Ram B., Jiang X.C., Parameter setting in a bio-inspired model for dynamic flexible job shop scheduling with sequence-dependent setups. European Journal of Industrial Engineering, 1 (2007) 182-199.

[100] Rajabinasab A, Mansour S, Dynamic flexible job shop scheduling with alternative process plans: an agent-based approach. International Journal of Advanced Manufacturing Technology, 54 (2011) 1091-1107.

[101] Moradi E., Ghomi S., Zandieh M., An effective architecture for scheduling flexible job-shop with machine availability constraints. International Journal of Advanced Manufacturing Technology, 51 (2010) 325-339.

[102] Bagheri A., Zandieh M., Mahdavi I., An artificial immune algorithm for the flexible job-shop scheduling problem. Future Generation Computers Systems-the International Journal of Grid Computing and Escience, 26 (2010) 533-541.

[103] Karthikeyan S., Asokan P., Nickolas S., Ahybrid discrete firefly algorithm for multi-objective flexible job shop scheduling problem with limited resource constraints. International Journal of Advanced Manufacturing Technology, 72 (2014) 1567-1579.

[104] Li J.Q., Pan Q.K., Xie S.X., An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems. Applied Mathematics and Computation, 218 (2012) 9353-9371.

[105] Li J.Q. Pan Q.K., Chemical-reaction optimization for flexible job-shop scheduling problems with maintenance activity. Applied Soft Computing, 12 (2012) 2896-2912.

[106] Rajkumar M., Asokan P., Vamsikrishn V., A GRASP algorithm for flexible job-shop scheduling with maintenance constraints. International Journal of Production Research, 48 (2010) 6821-6836.

[107] Ra jkumar M., Asokan P., Page T., A GRASP algorithms for flexible job-shop scheduling problem with limited resource constraints. International Journal of Production Research, 49 (2010) 2409-2423.

[108] Farughi H., Yegane B.Y., Fathian M., A new critical path method and a memetic algorithm for flexible job shop scheduling with overlapping operations. Simulation-Transactions of the Society for Modeling and Simulation International, 89 (2013) 264-277.

[109] Nie L., Gao L., Li P.G., Li X.Y., A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. Journal of Intelligent Manufacturing, 24 (2013) 763-744.

[110] Rahmati S.H.A., Zandieh M., A new biogeography-based optimization (BBO) algorithm for the flexible job shop scheduling problem. International Journal of Advanced Manufacturing Technology, 58 (2012) 1115-1129.

[111] Geem Z.W., Kim J.H., Loganathan G.V., A new heuristic optimization algorithm: harmony search, Simulation, 76 (2001) 60–68.

[112]Mahdavi M., Fesanghary M., Damangir E., An improved harmony search algorithm for solving optimization problems, Applied Mathematics and Computation, 188 (2007) 1567–1579.

[113]Lee K.S., Geem Z.W., Lee S.H., Bae K.W., The harmony search heuristic algorithm for discrete structural optimization, Engineering Optimization, 37 (2005) 663–684.

[114] Omran M.G.H., Mahdavi M., Global-best harmony search, Applied Mathematics and Computation, 198 (2008) 643–656.

[115]Das S., Mukhopadhyay A., Roy A., Abraham A., Panigrahi B.K.R., Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization, IEEE Transactions on System, Man, and Cybernetics, Part B: Cybernetics, 41 (2011) 89-106.

[116]Geem Z.W., Optimal scheduling of multiple dam system using harmony search algorithm, Lecture Notes in Computer Science, 4507 (2007) 316–323.

[117]Wang L., Pan Q.K., Tasgetiren M.F., Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms, Expert System with Applications, 37 (2010) 7929-7936.

[118]Pan Q.K., Suganthan P.N., Liang J.J., Tasgetiren M.F., A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem, Expert System with Applications, 38 (2011) 3252-3259.

[119]Gao K.Z., Pan Q.K., Li J.Q., Liang J.J., A hybrid harmony search algorithm for the no-wait flow shop scheduling problems, Asia-Pacific J of Operational research, 29 (2012) 1250012 1-23.

[120]Gao K.Z., Pan Q.K., Li J.Q., Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion, International Journal of Advanced Manufacturing Technology, 56 (2011) 683-692.

[121] Yuan Y., Xu H., Yang J.D., A hybrid harmony search algorithm for the flexible job shop scheduling problem. Applied Soft Computing, 13 (2013) 3259-3272.

[122] Yuan Y., Xu H., An integrated search heuristic for large-scale flexible job shop scheduling problems. Computer & Operations Research, 40 (2013) 2864-2877.

[123]Gao K.Z., Suganthan P.N., Pan Q.K., Chua T.J., Cai T.X., Chong C.S., Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling, Information Sciences, 289 (2014) 76-90.

[124] Karaboga, D., An idea based on honey bee swarm for numerical optimization, Technical Report TR06. Turkey: Computer Engineering Department, Erciyes University.

[125] Karaboga, N., A new design method based on artificial bee colony algorithm for digital IIR filters. Journal of the Franklin Institute 346 (2009) 328–348.

[126] Karaboga, D., and B. Akay, A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation, 24 (2009) 108–132.

[127] Karaboga, D., and B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization, 39 (2007) 459–471.

[128] Karaboga, D., and B. Basturk, On the performance of artificial bee colony (ABC) algorithm. Applied Soft Computing, 8 (2008) 687–697.

[129] Banharnsakun, A., T. Achalakul, and B. Sirinaovakul, The best-so-far selection in artificial bee colony algorithm. Applied Soft Computing, 11 (2011) 2888–2901.

[130] Wong, L. P., Low M. Y. H., and Chong C. S., Bee colony optimization with local search for traveling salesman problem. Proceedings of 6th IEEE International Conference on Industrial Informatics: 2008, 1019–1025.

[131] Pan, Q. K., Tasgetiren M. F., Suganthan P. N., and Chua T. J., A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. Information Sciences, 181 (2010) 2455–2468.

[132] Banharnsakun, A., B. Sirinaovakul, and T. Achalakul. "Job shop scheduling with the best-so-far ABC." Engineering Applications of Artificial Intelligence, 25 (2012) 583–593.

[133] Li J.Q., Pan Q.K., Xie S.X., Wang S., A hybrid artificial bee colony algorithm for flexible job shop scheduling problems. International Journal of Compuers Communications and Control, 6 (2011) 286-296.

[134] Li J.Q., Pan Q.K., Tasgetiren M.F., A discrete artificial bee colony algorithm for the multi-objective flexible job shop scheduling with maintenance activities. Applied Mathematical Modelling, 38 (2014) 1111-1132.

[135] Wang, L., G. Zhou, Y. Xu, S. Y. Wang, and M. Liu. An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. International Journal of Advanced Manufacturing Technology 60 (2012a) 303–315.

[136] Wang, L., G. Zhou, Y. Xu, and M. Liu. "An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling." International Journal of Advanced Manufacturing Technology, 60 (2012b) 1111–1123.

[137] Wang L., Zhou G., Xu Y., Liu M., A hybrid artificial bee colony algorithm for the fuzzy flexible job shop scheduling problem. International Journal of Production Research, 51 (2013) 3593-3608.

[138]Liang J.J., Suganthan P.N., Dynamic multi-swarm particle swarm optimizer, Proc. 2005 IEEE Conf. swarm intelligence symposium, SIS'05, Pasadena, California, USA, 2005, pp. 124-129.

[139]Zhang C.Y., Li P.G., Guan Z.L., Rao Y.Q., A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem, Computers and Operations Research, 34 (2007) 3229–3242.

[140] Kacem I., Hammadi S., Borne P., Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic, Mathematics and Computers in Simulation, 60 (2002) 245–276.

[141]Brandimarte P., Routing and scheduling in a flexible job shop by tabu search, Annals of Operations Research, 41 (1993) 157-183.

[142] Xing, L. N., Chen, Y. W., & Yang, K. W., An efficient search method for multi-objective flexible job shop scheduling problems. Journal of Intelligent Manufacturing, 20 (2009) 283–293.

[143] Seyed Habib A. Rahmati, Proposing a Pareto-based Multi-Objective Evolutionary Algorithm to Flexible Job Shop Scheduling Problem, World Academy of Science, Engineering and Technology, 61 (2012) 1160-1165.

[144]Rabiee M., Zandieh M., Ramezani P., Bi-objective partial flexible job shop scheduling problems: NSGA-II, NRGA, MOGA and PAES approaches, International Journal of Production Research, 50 (2012)7327-7342.

[145]Zhao S.Z., Suganthan P. N., Zhang Q., Decomposition-based multi-objective evolutionary algorithm with an ensemble of neighborhood sizes, IEEE Transactions on Evolutionary Computation, 16 (2012) 442-446.

[146]Wang X., Gao L., Zhang G., Shao X., A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem, International Journal of Advanced Manufacturing Technology, 51 (2010) 757–767.

[147] Lei D.M., A Genetic algorithm for flexible job shop scheduling with fuzzy processing time, International Journal of Production Research, 48 (2010) 2995–3013.

Appendix

A. The benchmark instances used in this thesis

In the first line there are (at least) 2 numbers: the first is the number of jobs and the second the number of machines (the 3rd is not necessary; it is the average number of machines per operation)
Every row represents one job: the first number is the number of operations of that job, the second number (let's say k>=1) is the number of machines that can process the first operation; then according to k, there are k pairs of numbers (machine, processing time) that specify which are the machines and the processing times; then the data for the second operation and so on...

Example:
6 6 1
6 1 3 1 1 1 3 1 2 6 1 4 7 1 6 3 1 5 6
6 1 2 8 1 3 5 1 5 10 1 6 10 1 1 10 1 4 4
6 1 3 5 1 4 4 1 6 8 1 1 9 1 2 1 1 5 7
6 1 2 5 1 1 5 1 3 5 1 4 3 1 5 8 1 6 9
6 1 3 9 1 2 3 1 5 5 1 6 4 1 1 3 1 4 1
6 1 2 3 1 4 3 1 6 9 1 1 10 1 5 4 1 3 1
first row = 6 jobs and 6 machines 1 machine per operation
second row: job 1 has 6 operations, the first operation can be processed by 1 machine that is machine 3 with processing time 1.

Kacem01:

4,5,0,
3,5,1,2,2,5,3,4,4,1,5,2,5,1,5,2,4,3,5,4,7,5,5,5,1,4,2,5,3,5,4,4,5,5,
3,5,1,2,2,5,3,4,4,7,5,8,5,1,5,2,6,3,9,4,8,5,5,5,1,4,2,5,3,4,4,54,5,5,
3,5,1,9,2,8,3,6,4,7,5,9,5,1,6,2,1,3,2,4,5,5,4,5,1,4,2,5,3,2,4,1,5,5,
2,5,1,1,2,5,3,2,4,4,5,12,5,1,5,2,1,3,2,4,1,5,2

Macem02:

8,8,0,
3,7,1,5,2,3,3,5,4,3,5,3,7,10,8,9,7,1,10,3,5,4,8,5,3,6,9,7,9,8,6,6,2,10,4,5,5,6,6,2,7,4,8,5,
4,6,1,5,2,7,3,3,4,9,5,8,7,9,7,2,8,3,5,4,2,5,6,6,7,7,10,8,9,6,2,10,4,5,5,6,6,4,7,1,8,7,6,1,10,2,8,3,9,4,6,5,4,6,7,
3,6,1,10,4,7,5,6,6,5,7,2,8,4,6,2,10,3,6,4,4,5,8,6,9,7,10,6,1,1,2,4,3,5,4,6,6,10,8,4,
3,8,1,3,2,1,3,6,4,5,5,9,6,7,7,8,8,4,8,1,12,2,11,3,7,4,8,5,10,6,5,7,6,8,9,8,1,4,2,6,3,2,4,10,5,3,6,9,7,5,8,7,
4,6,1,3,2,6,3,7,4,8,5,9,7,10,6,1,10,3,7,4,4,5,9,6,8,7,6,6,2,9,3,8,4,7,5,4,6,2,7,7,7,1,11,2,9,4,6,5,7,6,5,7,3,8,6,
3,7,1,6,2,7,3,1,4,4,5,6,6,9,8,10,7,1,11,3,9,4,9,5,9,6,7,7,6,8,4,6,1,10,2,5,3,9,4,10,5,11,7,10,
3,6,1,5,2,4,3,2,4,6,5,7,7,10,6,2,9,4,9,5,11,6,9,7,10,8,5,6,2,8,3,9,4,3,5,8,6,6,8,10,
4,6,1,2,2,8,3,5,4,9,6,4,8,10,6,1,7,2,4,3,7,4,8,5,9,7,10,7,1,9,2,9,4,8,5,5,6,6,7,7,8,1,6,1,9,3,3,4,7,5,1,6,5,7,8

Kacem03:

10,7,0,
3,7,1,1,2,4,3,6,4,9,5,3,6,5,7,2,7,1,8,2,9,3,5,4,4,5,1,6,1,7,3,7,1,4,2,8,3,10,4,4,5,11,6,4,7,3,
2,7,1,6,2,9,3,8,4,6,5,5,6,10,7,3,7,1,2,2,10,3,4,4,5,5,9,6,8,7,4,
3,7,1,15,2,4,3,8,4,4,5,8,6,7,7,1,7,1,9,2,6,3,1,4,10,5,7,6,1,7,6,7,1,11,2,2,3,7,4,5,5,2,6,3,7,14,
3,7,1,2,2,8,3,5,4,8,5,9,6,4,7,3,7,1,5,2,3,3,8,4,1,5,9,6,3,7,6,7,1,1,2,2,3,6,4,4,5,1,6,7,7,2,
3,7,1,7,2,1,3,8,4,5,5,4,6,3,7,9,7,1,2,2,4,3,5,4,10,5,6,6,4,7,9,7,1,5,2,1,3,7,4,1,5,6,6,6,7,2,
3,7,1,8,2,7,3,4,4,56,5,9,6,8,7,4,7,1,5,2,14,3,1,4,9,5,6,6,5,7,8,7,1,3,2,5,3,2,4,5,5,4,6,5,7,7,
3,7,1,5,2,6,3,3,4,6,5,5,6,15,7,2,7,1,6,2,5,3,4,4,9,5,5,6,4,7,3,7,1,9,2,8,3,2,4,8,5,6,6,1,7,7,
3,7,1,6,2,1,3,4,4,1,5,10,6,4,7,3,7,1,11,2,13,3,9,4,8,5,9,6,10,7,8,7,1,4,2,2,3,7,4,8,5,3,6,10,7,7,
3,7,1,12,2,5,3,4,4,5,5,4,6,5,7,5,7,1,4,2,2,3,15,4,99,5,4,6,7,7,3,7,1,9,2,5,3,11,4,2,5,5,6,4,7,2,
3,7,1,9,2,4,3,13,4,10,5,7,6,6,7,8,7,1,4,2,3,3,25,4,3,5,8,6,1,7,2,7,1,1,2,2,3,6,4,11,5,13,6,3,7,5

Kacem04:

10,10,0,
,10,1,1,2,4,3,6,4,9,5,3,6,5,7,2,8,8,9,9,10,5,10,1,4,2,1,3,1,4,3,5,4,6,8,7,10,8,4,9,11,10,4,10,1,3,2,2,3,5,4,1,5,5,6,6,7,9,8,5,9,
10,10,3,
,10,1,2,2,10,3,4,4,5,5,9,6,8,7,4,8,15,9,8,10,4,10,1,4,2,8,3,7,4,1,5,9,6,6,7,1,8,10,9,7,10,1,10,1,6,2,11,3,2,4,7,5,5,6,3,7,5,8,1
4,9,9,10,2,
,10,1,8,2,5,3,8,4,9,5,4,6,3,7,5,8,3,9,8,10,1,10,1,9,2,3,3,6,4,1,5,2,6,6,7,4,8,1,9,7,10,2,10,1,7,2,1,3,8,4,5,5,4,6,9,7,1,8,2,9,3,1
0,4,
,10,1,5,2,10,3,6,4,4,5,9,6,5,7,1,8,7,9,1,10,6,10,1,4,2,2,3,3,4,8,5,7,6,4,7,6,8,9,9,8,10,4,10,1,7,2,3,3,12,4,1,5,6,6,5,7,8,8,3,9,
5,10,2,
,10,1,7,2,10,3,4,4,5,5,6,6,3,7,5,8,15,9,2,10,6,10,1,5,2,6,3,3,4,9,5,8,6,2,7,8,8,6,9,1,10,7,10,1,6,2,1,3,4,4,1,5,10,6,4,7,3,8,11,
9,13,10,9,
,10,1,8,2,9,3,10,4,8,5,4,6,2,7,7,8,8,9,3,10,10,10,1,7,2,3,3,12,4,5,5,4,6,3,7,6,8,9,9,2,10,15,10,1,4,2,7,3,3,4,6,5,3,6,4,7,1,8,5,
9,1,10,11,
,10,1,1,2,7,3,8,4,3,5,4,6,9,7,4,8,13,9,10,10,7,10,1,3,2,8,3,1,4,2,5,3,6,6,7,11,8,2,9,13,10,3,10,1,5,2,4,3,2,4,1,5,2,6,1,7,8,8,1
4,9,5,10,7,
,10,1,5,2,7,3,11,4,3,5,2,6,9,7,8,8,5,9,12,10,8,10,1,8,2,3,3,10,4,7,5,5,6,13,7,4,8,6,9,8,10,4,10,1,6,2,2,3,13,4,5,5,4,6,3,7,5,8,
7,9,9,10,5 ,
,10,1,3,2,9,3,1,4,3,5,8,6,1,7,6,8,7,9,5,10,4,10,1,4,2,6,3,2,4,5,5,7,6,3,7,1,8,9,9,6,10,7,10,1,8,2,5,3,4,4,8,5,6,6,1,7,2,8,3,9,10,
10,12 ,
,10,1,4,2,3,3,1,4,6,5,7,6,1,7,2,8,6,9,20,10,6,10,1,3,2,1,3,8,4,1,5,9,6,4,7,1,8,4,9,17,10,15,10,1,9,2,2,3,4,4,2,5,3,6,5,7,2,8,4,9
,10,10,23

Kacem05:

15,10,0,

4,10,1,1,2,4,3,6,4,9,5,3,6,5,7,2,8,8,9,9,10,4,10,1,1,2,1,3,3,4,4,5,8,6,10,7,4,8,11,9,4,10,3,10,1,2,2,5,3,1,4,5,5,6,6,9,7,5,8,10,
9,3,10,2,10,1,10,2,4,3,5,4,9,5,8,6,4,7,15,8,8,9,4,10,4,
4,10,1,4,2,8,3,7,4,1,5,9,6,6,7,1,8,10,9,7,10,1,10,1,6,2,11,3,2,4,7,5,5,6,3,7,5,8,14,9,9,10,2,10,1,8,2,5,3,8,4,9,5,4,6,3,7,5,8,3,
9,8,10,1,10,1,9,2,3,3,6,4,1,5,2,6,6,7,4,8,1,9,7,10,2,
4,10,1,7,2,1,3,8,4,5,5,4,6,9,7,1,8,2,9,3,10,4,10,1,5,2,10,3,6,4,4,5,9,6,5,7,1,8,7,9,1,10,6,10,1,4,2,2,3,3,4,8,5,7,6,4,7,6,8,9,9,8
,10,4,10,1,7,2,3,3,12,4,1,5,6,6,5,7,8,8,3,9,5,10,2,
4,10,1,6,2,2,3,5,4,4,5,1,6,2,7,3,8,6,9,5,10,4,10,1,8,2,5,3,7,4,4,5,1,6,2,7,36,8,5,9,8,10,5,10,1,9,2,6,3,2,4,4,5,5,6,1,7,3,8,6,9,5
,10,2,10,1,11,2,4,3,5,4,6,5,2,6,7,7,5,8,4,9,2,10,1,
4,10,1,6,2,9,3,2,4,3,5,5,6,8,7,7,8,4,9,1,10,2,10,1,5,2,4,3,6,4,3,5,5,6,2,7,28,8,7,9,4,10,5,10,1,6,2,2,3,4,4,3,5,6,6,5,7,2,8,4,9,7
,10,9,10,1,6,2,5,3,4,4,2,5,3,6,2,7,5,8,4,9,7,10,2,
2,10,1,4,2,1,3,3,4,2,5,6,6,9,7,8,8,5,9,4,10,2,10,1,1,2,3,3,6,4,5,5,4,6,7,7,5,8,4,9,6,10,5,
2,10,1,1,2,4,3,2,4,5,5,3,6,6,7,9,8,8,9,5,10,4,10,1,2,2,1,3,4,4,5,5,2,6,3,7,5,8,4,9,2,10,5,
4,10,1,2,2,3,3,6,4,2,5,5,6,4,7,1,8,5,9,8,10,7,10,1,4,2,5,3,6,4,2,5,3,6,5,7,4,8,1,9,2,10,5,10,1,3,2,5,3,4,4,2,5,5,6,49,7,8,8,5,9,4
,10,5,10,1,1,2,2,3,36,4,5,5,2,6,3,7,6,8,4,9,11,10,2,
4,10,1,6,2,3,3,2,4,22,5,44,6,11,7,10,8,23,9,5,10,1,10,1,2,2,3,3,2,4,12,5,15,6,10,7,12,8,14,9,18,10,16,10,1,20,2,17,3,12,4,5,
5,9,6,6,7,4,8,7,9,5,10,6,10,1,9,2,8,3,7,4,4,5,5,6,8,7,7,8,6,9,56,10,2,
4,10,1,5,2,8,3,7,4,4,5,56,6,3,7,2,8,5,9,4,10,1,10,1,2,2,5,3,6,4,9,5,8,6,5,7,4,8,2,9,5,10,4,10,1,6,2,3,3,2,4,5,5,4,6,7,7,4,8,5,9,2
,10,1,10,1,3,2,2,3,5,4,6,5,5,6,8,7,7,8,4,9,5,10,2,
4,10,1,1,2,2,3,3,4,6,5,5,6,2,7,1,8,4,9,2,10,1,10,1,2,2,3,3,6,4,3,5,2,6,1,7,4,8,10,9,12,10,1,10,1,3,2,6,3,2,4,5,5,8,6,4,7,6,8,3,9,
2,10,5,10,1,4,2,1,3,45,4,6,5,2,6,4,7,1,8,25,9,2,10,4 ,
4,10,1,9,2,8,3,5,4,6,5,3,6,6,7,5,8,2,9,4,10,2,10,1,5,2,8,3,9,4,5,5,4,6,75,7,63,8,6,9,5,10,21,10,1,12,2,5,3,4,4,6,5,3,6,2,7,5,8,4
,9,2,10,5,10,1,8,2,7,3,9,4,5,5,6,6,3,7,2,8,5,9,8,10,4 ,
4,10,1,4,2,2,3,5,4,6,5,8,6,5,7,6,8,4,9,6,10,2,10,1,3,2,5,3,4,4,7,5,5,6,8,7,6,8,6,9,3,10,2,10,1,5,2,4,3,5,4,8,5,5,6,4,7,6,8,5,9,4,
10,2,10,1,3,2,2,3,5,4,6,5,5,6,4,7,8,8,5,9,6,10,4,
4,10,1,2,2,3,3,5,4,4,5,6,6,5,7,4,8,85,9,4,10,5,10,1,6,2,2,3,4,4,5,5,8,6,6,7,5,8,4,9,2,10,6,10,1,3,2,25,3,4,4,8,5,5,6,6,7,3,8,2,9,
5,10,4,10,1,8,2,5,3,6,4,4,5,2,6,3,7,6,8,8,9,5,10,4,
4,10,1,2,2,5,3,6,4,8,5,5,6,6,7,3,8,2,9,5,10,4,10,1,5,2,6,3,2,4,5,5,4,6,2,7,5,8,3,9,2,10,5,10,1,4,2,5,3,2,4,3,5,5,6,2,7,8,8,4,9,7,
10,5,10,1,6,2,2,3,11,4,14,5,2,6,3,7,6,8,5,9,4,10,8 ,

MK01:

10 6 2
6 2 1 5 3 4 3 5 3 3 5 2 1 2 3 4 6 2 3 6 5 2 6 1 1 1 3 1 3 6 6 3 6 4 3
5 1 2 6 1 3 1 1 1 2 2 2 6 4 6 3 6 5 2 6 1 1

5 1 2 6 2 3 4 6 2 3 6 5 2 6 1 1 3 3 4 2 6 6 6 2 1 1 5 5
5 3 6 5 2 6 1 1 1 2 6 1 3 1 3 5 3 3 5 2 1 2 3 4 6 2
6 3 5 3 3 5 2 1 3 6 5 2 6 1 1 1 2 6 2 1 5 3 4 2 2 6 4 6 3 3 4 2 6 6 6
6 2 3 4 6 2 1 1 2 3 3 4 2 6 6 6 1 2 6 3 6 5 2 6 1 1 2 1 3 4 2
5 1 6 1 2 1 3 4 2 3 3 4 2 6 6 6 3 2 6 5 1 1 6 1 3 1
5 2 3 4 6 2 3 3 4 2 6 6 6 3 6 5 2 6 1 1 1 2 6 2 2 6 4 6
6 1 6 1 2 1 1 5 5 3 6 6 3 6 4 3 1 1 2 3 3 4 2 6 6 6 2 2 6 4 6
6 2 3 4 6 2 3 3 4 2 6 6 6 3 5 3 3 5 2 1 1 6 1 2 2 6 4 6 2 1 3 4 2

MK02:

10 6 4
6 6 3 3 4 5 1 3 6 6 2 2 5 3 2 6 5 3 4 6 1 1 5 6 3 3 4 3 2 6 6 5 1 2 6 2 6 3 5 6 3 3 2 2 1 5 4
6 5 6 1 5 6 1 3 2 4 4 2 2 6 3 5 6 1 5 2 2 2 4 3 3 3 3 2 2 1 5 4 6 3 3 4 5 1 3 6 6 2 2 5 3
6 6 1 1 5 6 3 3 4 3 2 6 6 5 6 5 3 4 6 2 4 6 6 3 6 1 2 3 3 2 2 1 5 4 5 3 5 1 4 2 3 6 3 5 2 6 4 1 1 5 2 4 5 5 3 3 6 3 5 6 3 1 4 4 6 3 6 5 3
6 5 3 5 1 4 2 3 6 3 5 2 5 6 1 5 6 1 3 2 4 4 2 1 2 6 6 1 1 5 6 3 3 4 3 2 6 6 5 5 1 4 4 5 2 3 6 3 5 4 6 4 1 1 5 2 4 5 5 3 3 6 3
6 6 5 3 4 6 2 4 6 6 3 6 1 2 5 1 4 4 5 2 3 6 3 5 4 1 4 3 5 6 3 1 4 4 6 3 6 5 3 5 6 1 5 6 1 3 2 4 4 2 2 2 4 3 3
6 5 6 3 1 4 4 6 3 6 5 3 2 6 5 3 4 5 3 5 1 4 2 3 6 3 5 2 6 5 3 4 6 2 4 6 6 3 6 1 2 1 2 6 5 6 1 5 6 1 3 2 4 4 2
5 6 4 1 1 5 2 4 5 5 3 3 6 3 1 5 2 6 5 3 4 6 2 4 6 6 3 6 1 2 6 3 3 4 5 1 3 6 6 2 2 5 3 5 6 3 1 4 4 6 3 6 5 3
6 2 2 4 3 3 5 3 5 1 4 2 3 6 3 5 2 6 5 3 4 6 2 4 6 6 3 6 1 2 5 6 3 1 4 4 6 3 6 5 3 5 1 4 4 5 2 3 6 3 5 4 5 6 1 5 6 1 3 2 4 4 2
5 1 2 6 2 6 5 3 4 5 6 1 5 6 1 3 2 4 4 2 5 1 4 4 5 2 3 6 3 5 4 2 2 4 3 3
6 1 4 3 6 5 3 4 6 2 4 6 6 3 6 1 2 5 6 3 1 4 4 6 3 6 5 3 6 4 1 1 5 2 4 5 5 3 3 6 3 2 6 3 5 6 5 6 1 5 6 1 3 2 4 4 2

MK03:

15 8 3
10 4 7 15 8 11 4 5 5 19 2 3 18 4 5 4 8 18 7 3 6 11 3 16 4 5 7 2 1 7 2 3 19 2 5 6 6 3 3 4 5 5 2 8 18 1 5 2 1 1 17 5 5 10 2 10 1 12 8 5 3 14 3 7 15 6 2 8 19
10 4 8 18 7 3 6 11 3 16 1 1 17 2 2 1 4 13 5 5 10 2 10 1 12 8 5 3 14 5 4 11 1 9 2 18 6 18 3 13 2 6 15 7 13 4 7 15 8 11 4 5 5 19 4 5 7 2 1 7 2 3 19 4 4 11 1 7 6 13 8 3 3 7 15 6 2 8 19
10 2 3 3 5 5 4 5 7 2 1 7 2 3 19 2 3 18 4 5 2 5 6 6 3 4 4 11 1 7 6 13 8 3 3 7 15 6 2 8 19 5 4 11 1 9 2 18 6 18 3 13 3 4 5 5 2 8 18 1 1 17 2 2 1 4 13
10 2 3 18 4 5 2 3 3 5 5 5 4 11 1 9 2 18 6 18 3 13 4 4 11 1 7 6 13 8 3 2 6 15 7 13 4 5 7 2 1 7 2 3 19 1 5 2 4 8 18 7 3 6 11 3 16 1 1 17 2 5 6 6 3
10 2 6 15 7 13 3 7 15 6 2 8 19 1 5 2 4 7 15 8 11 4 5 5 19 5 4 11 1 9 2 18 6 18 3 13 4 5 7 2 1 7 2 3 19 3 4 5 5 2 8 18 2 5 6 6 3 2 3 3 5 5 5 5 10 2 10 1 12 8 5 3 14
10 2 2 1 4 13 2 6 15 7 13 2 3 18 4 5 4 8 18 7 3 6 11 3 16 5 4 11 1 9 2 18 6 18 3 13 5 5 10 2 10 1 12 8 5 3 14 4 4 11 1 7 6 13 8 3 4 7 15 8 11 4 5 5 19 2 5 6 6 3 2 3 3 5 5
10 5 5 10 2 10 1 12 8 5 3 14 4 4 11 1 7 6 13 8 3 2 2 1 4 13 1 1 17 2 6 15 7 13 4 5 7 2 1 7 2 3 19 1 5 2 5 4 11 1 9 2 18 6 18 3 13 2 3 18 4 5 3 7 15 6 2 8 19
10 3 7 15 6 2 8 19 1 1 17 4 7 15 8 11 4 5 5 19 2 6 15 7 13 5 5 10 2 10 1 12 8 5 3 14 4 4 11 1 7 6 13 8 3 5 4 11 1 9 2 18 6 18 3 13 2 2 1 4 13 2 3 18 4 5 2 3 3 5 5
10 1 1 17 5 5 10 2 10 1 12 8 5 3 14 4 8 18 7 3 6 11 3 16 3 7 15 6 2 8 19 2 6 15 7 13 4 4 11 1 7 6 13 8 3 1 5 2 2 2 1 4 13 5 4 11 1 9 2 18 6 18 3 13 4 7 15 8 11 4 5 5 19
10 1 1 17 2 6 15 7 13 3 4 5 5 2 8 18 5 4 11 1 9 2 18 6 18 3 13 4 4 11 1 7 6 13 8 3 2 3 18 4 5 2 5 6 6 3 3 7 15 6 2 8 19 4 8 18 7 3 6 11 3 16 5 5 10 2 10 1 12 8 5 3 14
10 2 2 1 4 13 3 7 15 6 2 8 19 4 8 18 7 3 6 11 3 16 2 3 18 4 5 2 5 6 6 3 1 1 17 2 3 3 5 5 3 4 5 5 2 8 18 5 5 10 2 10 1 12 8 5 3 14 5 4 11 1 9 2 18 6 18 3 13
10 4 4 11 1 7 6 13 8 3 3 4 5 5 2 8 18 4 8 18 7 3 6 11 3 16 1 1 17 5 4 11 1 9 2 18 6 18 3 13 3 7 15 6 2 8 19 1 5 2 2 3 3 5 5 4 7 15 8 11 4 5 5 19 2 2 1 4 13
10 5 5 10 2 10 1 12 8 5 3 14 1 5 2 2 3 18 4 5 4 5 7 2 1 7 2 3 19 2 6 15 7 13 4 8 18 7 3 6 11 3 16 4 7 15 8 11 4 5 5 19 5 4 11 1 9 2 18 6 18 3 13 2 5 6 6 3 4 4 11 1 7 6 13 8 3
10 4 8 18 7 3 6 11 3 16 3 4 5 5 2 8 18 2 2 1 4 13 4 5 7 2 1 7 2 3 19 2 5 6 6 3 2 3 18 4 5 2 6 15 7 13 1 5 2 5 4 11 1 9 2 18 6 18 3 13 1 1 17
10 5 5 10 2 10 1 12 8 5 3 14 2 5 6 6 3 2 6 15 7 13 4 7 15 8 11 4 5 5 19 4 8 18 7 3 6 11 3 16 1 1 17 5 4 11 1 9 2 18 6 18 3 13 3 4 5 5 2 8 18 2 3 18 4 5 4 5 7 2 1 7 2 3 19

MK04:

15 8 2
8 1 1 6 2 1 6 7 9 2 6 7 3 1 2 4 2 7 5 3 1 8 3 9 8 9 3 2 3 4 8 3 2 2 5 5 6 7 2 6 1 4 7

7 16126147116267313234832162172
6 16132348323327144242752173724431
5 17211621679267312455 7
7 17221679244313183989217373234832245 57
9 16224431332714426147245573183989217371612167 9
5 25567217372614716226731
6 24557255673234832162161216 79
9 11621679244313183989242752614717221737323483 2
5 25567116172245572167 9
4 31839891163234832242 75
6 24275161116217373183989172
4 16226731261472556 7
3 2556716124275
6 24557172318398932348323327144116

MK05:

15 4 2
6 2352721848216251372452624515
5 1372162514624526218 26
8 2473923527245152184821625146218262493 6
7 2451524739218481482182624526146
6 2371524627247391382352721826
9 1462452613823715246271482182621848245 15
5 1382473921625246271 37
8 2371513824739245151371482493621625
9 2352714824526216251462184921848218261 37
9 2182621848218492493621625138137146245 26
7 218262184821625137146138249 36
6 1481372473921625138218 48
7 1482493621848246272462721826237 15
7 2162523715218482182624515246271 46
7 1382184924936137245262182621625

MK06:

10 15 3
15 4286372952971257414912710421182375385851388253810935611625251991 5
7462106122795624872521584218373102894537537933945811
15 513882538109574149127104356116252158421837248722106123102894527 95
63753793375385839458112971221182428637295525199157 46
15 2118227956210612248725215842183739458112971237537935741491271044 2
863729551388253810931028945525199157463561162375385 8
15 3561162525199157465138825381095215842183724872210612375385829712 3
753793394581142863729521182574149127104279563102894 5
15 3102894521182394581129712375385852158421837356116237537934286372
952106125741491271042795652519915746513882538109248 72
15 3753858513882538109279563561162525199157462487229712521584218375
741491271044286372952118237537932106123945811310289 45
15 3561162310289453753858513882538109211822971252158421837375379357 4
1491271043945811210612428637295279562487252519915746
15 5741491271043753793375385821182356116252519915746310289453945811 2
971242863729551388253810924872210612521584218 37 27956
15 4286372953945811375385857414912710452158421837248722971231028945 5
1388253810921061252519915746375379327956211823561162
15 2118242863729531028945375385837537932106122795639458115741491271 0
4525199157465138825381093561162521584218372487229712

5 2 1 3 5 13 5 3 8 5 2 4 5 1 7 2 7 2 2 4 1 15 3 1 8 5 1 2 5 3 1 3 5 13 3 2
5 5 2 18 5 1 4 19 1 9 3 3 1 4 18 2 4 11 3 9 1 2 4 3 5 12 3 14 4 19
5 2 2 4 1 15 4 4 10 3 10 2 17 5 8 4 5 18 3 13 2 2 1 5 5 4 10 5 15 1 2 3 9 2 16 2 3 15 1 6
5 3 1 3 5 13 3 2 2 3 18 1 15 5 2 18 5 1 4 19 1 9 3 3 3 5 12 3 14 4 19 1 4 5
5 5 3 8 5 2 4 5 1 7 2 7 2 3 18 1 15 2 1 15 5 7 2 2 7 1 17 2 2 4 1 15
5 1 4 5 2 1 15 5 7 2 2 4 1 15 3 1 3 5 13 3 2 4 4 6 2 17 3 15 5 7
5 4 4 6 2 17 3 15 5 7 3 3 18 1 2 4 15 4 2 14 4 14 3 19 5 15 1 2 4 2 2 7 1 17
5 5 2 18 5 1 4 19 1 9 3 3 4 4 6 2 17 3 15 5 7 3 1 8 5 1 2 5 4 2 14 4 14 3 19 5 15 2 1 17 5 15
5 2 1 15 5 7 4 4 10 3 10 2 17 5 8 2 3 15 1 6 1 4 5 5 3 16 5 17 4 10 2 10 1 7
5 1 4 18 3 1 8 5 1 2 5 5 3 8 5 2 4 5 1 7 2 7 2 1 15 5 7 2 1 17 5 15
5 3 5 12 3 14 4 19 4 4 10 3 10 2 17 5 8 2 3 15 1 6 5 3 8 5 2 4 5 1 7 2 7 5 3 16 5 17 4 10 2 10 1 7
5 2 1 17 5 15 1 4 18 4 2 17 5 19 4 5 3 12 3 3 18 1 2 4 15 3 1 8 5 1 2 5
5 2 5 1 3 5 3 3 18 1 2 4 15 4 4 10 3 10 2 17 5 8 2 3 18 1 15 5 3 8 5 2 4 5 1 7 2 7
5 5 3 8 5 2 4 5 1 7 2 7 2 5 1 3 5 3 5 12 3 14 4 19 5 3 16 5 17 4 10 2 10 1 7 2 1 17 5 15
5 5 4 10 5 15 1 2 3 9 2 16 2 4 11 3 9 1 2 4 2 1 15 5 7 1 4 5
5 5 3 8 5 2 4 5 1 7 2 7 4 2 14 4 14 3 19 5 15 3 3 18 1 2 4 15 2 3 15 1 6 5 2 18 5 1 4 19 1 9 3 3
5 1 2 4 3 1 8 5 1 2 5 2 5 1 3 5 2 3 18 1 15 2 1 15 5 7
5 3 1 3 5 13 3 2 4 4 6 2 17 3 15 5 7 4 5 18 3 13 2 2 1 5 1 4 18 2 1 3 5 13
5 1 4 5 2 2 4 1 15 1 4 18 2 1 15 5 7 5 4 10 5 15 1 2 3 9 2 16

MK08:

20 10 2
10 2 7 18 4 5 2 5 7 7 7 1 3 19 1 7 14 2 4 5 10 12 1 1 10 1 10 18 2 7 10 8 19 2 3 11 8 9 2 3 5 8 12
12 1 2 5 2 7 18 4 5 2 3 5 8 12 1 1 10 1 10 19 2 3 15 4 19 1 7 14 1 5 9 2 5 14 9 5 1 1 19 2 7 10 8 19 1 1 16
14 2 5 14 9 5 1 1 19 1 1 10 1 3 19 2 7 18 4 5 2 4 5 10 12 2 3 5 8 12 1 10 10 1 5 9 1 1 7 2 7 10 8 19 1 1 10 1 10 19 1 10 18
10 1 10 10 2 5 7 7 7 1 7 14 1 1 10 1 10 18 2 3 15 7 13 2 10 14 5 7 2 3 11 8 9 1 9 11 1 5 9
12 1 5 9 2 5 14 9 5 2 7 18 4 5 2 3 11 8 9 1 1 10 1 9 11 1 1 7 1 7 14 2 4 5 10 12 2 3 15 4 19 1 8 18 1 10 19
10 2 3 15 7 13 1 3 19 1 5 9 1 10 19 2 3 5 8 12 2 7 18 4 5 2 8 14 10 9 2 4 5 10 12 1 10 18 1 1 7
12 1 1 10 1 10 18 1 1 7 1 5 9 2 8 14 10 9 2 7 10 8 19 2 3 15 4 19 2 10 14 5 7 1 8 18 1 10 19 1 1 19 1 1 10
11 1 1 10 1 7 14 1 1 10 2 3 15 4 19 2 5 14 9 5 2 7 18 4 5 1 3 19 1 1 19 2 4 5 10 12 1 5 9 1 10 19
14 2 7 10 8 19 2 8 14 10 9 1 1 19 1 10 19 2 10 14 5 7 1 2 5 2 4 5 10 12 2 5 7 7 7 1 1 16 1 1 7 1 9 11 1 3 19 1 1 10 1 10 18
11 1 10 19 2 10 14 5 7 1 8 18 2 3 11 8 9 1 1 7 1 1 10 2 5 14 9 5 2 3 15 4 19 1 10 18 1 3 19 1 1 19
11 2 5 14 9 5 1 1 10 1 8 18 2 3 15 4 19 2 7 10 8 19 2 3 5 8 12 2 3 11 8 9 2 8 14 10 9 1 10 10 1 9 11 1 3 19
10 1 10 19 2 3 11 8 9 2 5 7 7 7 1 1 16 1 7 14 2 7 18 4 5 2 4 5 10 12 1 1 10 1 8 18 2 5 14 9 5
11 2 10 14 5 7 1 10 19 2 7 10 8 19 2 3 15 4 19 1 1 19 1 8 18 2 8 14 10 9 2 3 11 8 9 1 10 18 2 5 14 9 5 1 2 5
11 1 1 10 2 5 7 7 7 1 1 10 1 9 11 1 7 14 2 3 15 7 13 2 8 14 10 9 1 1 16 2 3 5 8 12 2 5 14 9 5 1 2 5
11 2 5 14 9 5 2 5 7 7 7 1 7 14 1 10 10 2 7 10 8 19 2 3 15 4 19 2 7 18 4 5 1 1 7 2 3 11 8 9 1 1 19 1 8 18
11 1 2 5 2 7 10 8 19 1 10 10 1 9 11 1 8 18 2 10 14 5 7 2 5 14 9 5 1 1 10 1 1 19 2 3 15 7 13 2 8 14 10 9
13 1 10 10 2 5 14 9 5 1 5 9 1 10 19 1 1 10 2 3 5 8 12 1 2 5 2 10 14 5 7 1 1 10 2 8 14 10 9 2 3 15 7 13 1 1 16 1 7 14
11 2 3 15 7 13 1 2 5 1 10 19 1 3 19 1 8 18 1 1 7 1 5 9 1 7 14 2 7 18 4 5 1 1 10 2 5 14 9 5
10 2 7 10 8 19 1 2 5 2 3 11 8 9 1 9 11 2 4 5 10 12 1 10 18 2 7 18 4 5 2 8 14 10 9 2 3 5 8 12 1 10 19
10 1 10 18 1 10 10 1 7 14 1 9 11 2 3 15 7 13 1 2 5 2 8 14 10 9 2 3 5 8 12 1 5 9 1 1 16

MK09:

20 10 3
12 2 2 10 1 11 1 8 17 1 8 14 1 1 10 2 2 16 10 18 2 9 6 2 12 4 7 9 4 11 3 10 1 16 2 5 19 1 7 1 9 11 1 4 16 1 2 5 5 7 9 9 9 4 6 8 14 6 16
13 1 8 17 2 5 6 4 11 2 2 10 1 11 2 5 9 8 8 2 2 16 3 11 4 1 8 5 14 10 15 6 12 4 6 10 8 15 7 5 2 8 2 5 19 1 7 4 7 9 4 11 3 10 1 16 1 1 10 4 1 16 3 11 7 17 4 7 1 4 16 4 3 11 5 8 7 11 9 17
11 4 6 10 8 15 7 5 2 8 2 5 9 8 8 2 2 16 10 18 2 2 10 1 11 5 7 9 9 9 4 6 8 14 6 16 1 4 16 2 5 19 1 7 1 1 10 2 5 6 4 11 2 2 16 3 11 1 3 14
11 4 1 8 5 14 10 15 6 12 2 5 19 1 7 4 4 11 8 16 9 15 1 6 1 8 14 1 4 16 1 8 17 4 1 16 3 11 7 17 4 7 4 10 6 8 13 5 5 2 8 1 3 14 4 7 9 4 11 3 10 1 16 1 1 10
14 1 8 17 1 4 16 1 5 9 4 10 6 8 13 5 5 2 8 4 1 16 3 11 7 17 4 7 2 2 16 10 18 4 6 10 8 15 7 5 2 8 1 8 14 2 5 6 4 11 4 2 5 7 13 10 10 5 11 5 7 9 9 9 4 6 8 14 6 16 2 5 9 8 8 4 1 8 5 14 10 15 6 12 2 5 19 1 7
11 4 2 5 7 13 10 10 5 11 2 2 16 10 18 1 1 10 1 3 14 1 5 9 5 7 9 9 9 4 6 8 14 6 16 1 8 17 1 8 14 1 2 5 4 6 10 8 15 7 5 2 8 4 4 11 8 16 9 15 1 6
14 1 8 14 1 8 17 2 5 9 8 8 1 4 16 1 1 10 4 2 5 7 13 10 10 5 11 1 2 5 2 5 6 4 11 5 7 9 9 9 4 6 8 14 6 16 4 4 11 8 16 9 15 1 6 5 2 8 1 19 8 13 6 14 10 18 4 6 10 8 15 7 5 2 8 4 1 16 3 11 7 17 4 7 2 2 16 10 18

13 1 1 10 4 10 6 8 13 5 5 2 8 1 5 9 4 7 9 4 11 3 10 1 16 1 9 11 4 2 5 7 13 10 10 5 11 4 6 10 8 15 7 5 2 8 1 2 5 5 2 8 1 19 8 13 6 14 10 18 5 7 9 9 9 4 6 8 14 6 16 2 2 10 1 11 4 1 16 3 11 7 17 4 7 2 5 6 4 11
11 1 8 17 1 2 5 1 1 10 1 4 16 2 5 6 4 11 4 7 9 4 11 3 10 1 16 5 2 8 1 19 8 13 6 14 10 18 1 9 11 2 9 6 2 12 2 2 10 1 11 2 5 9 8 8
12 1 4 16 4 4 11 8 16 9 15 1 6 1 3 14 4 2 5 7 13 10 10 5 11 1 9 11 5 7 9 9 9 4 6 8 14 6 16 2 5 6 4 11 4 1 16 3 11 7 17 4 7 2 2 10 1 11 2 2 16 3 11 4 1 8 5 14 10 15 6 12 1 1 10
10 1 9 11 1 5 9 5 2 8 1 19 8 13 6 14 10 18 1 4 16 4 4 11 8 16 9 15 1 6 2 5 9 8 8 4 7 9 4 11 3 10 1 16 1 3 14 1 1 10 4 1 16 3 11 7 17 4 7
11 4 10 6 8 13 5 5 2 8 4 4 11 8 16 9 15 1 6 1 4 16 2 9 6 2 12 4 6 10 8 15 7 5 2 8 4 7 9 4 11 3 10 1 16 1 2 5 1 8 14 5 7 9 9 9 4 6 8 14 6 16 2 5 6 4 11 2 2 16 10 18
11 1 2 5 1 3 14 2 9 6 2 12 1 5 9 4 2 5 7 13 10 10 5 11 4 1 16 3 11 7 17 4 7 2 2 10 1 11 1 8 17 2 5 19 1 7 1 1 10 4 7 9 4 11 3 10 1 16
10 4 3 11 5 8 7 11 9 17 1 1 10 2 2 16 10 18 2 2 10 1 11 4 6 10 8 15 7 5 2 8 4 4 11 8 16 9 15 1 6 1 4 16 4 1 16 3 11 7 17 4 7 4 7 9 4 11 3 10 1 16 2 2 16 3 11
12 1 1 10 4 4 11 8 16 9 15 1 6 4 2 5 7 13 10 10 5 11 5 2 8 1 19 8 13 6 14 10 18 2 5 6 4 11 2 9 6 2 12 1 2 5 4 10 6 8 13 5 5 2 8 1 4 16 2 2 16 3 11 2 2 10 1 11 4 6 10 8 15 7 5 2 8
14 1 8 17 4 4 11 8 16 9 15 1 6 1 3 14 2 9 6 2 12 1 8 14 4 6 10 8 15 7 5 2 8 4 7 9 4 11 3 10 1 16 4 2 5 7 13 10 10 5 11 4 1 8 5 14 10 15 6 12 2 2 10 1 11 1 4 16 4 3 11 5 8 7 11 9 17 2 5 19 1 7 4 10 6 8 13 5 5 2 8
13 5 2 8 1 19 8 13 6 14 10 18 1 9 11 4 7 9 4 11 3 10 1 16 1 8 17 4 10 6 8 13 5 5 2 8 2 5 6 4 11 1 1 10 4 6 10 8 15 7 5 2 8 2 2 10 1 11 2 2 16 10 18 4 1 16 3 11 7 17 4 7 1 3 14 2 5 19 1 7
11 5 2 8 1 19 8 13 6 14 10 18 5 7 9 9 9 4 6 8 14 6 16 2 5 6 4 11 4 10 6 8 13 5 5 2 8 1 3 14 4 3 11 5 8 7 11 9 17 1 9 11 2 2 10 1 11 4 2 5 7 13 10 10 5 11 1 8 14 4 1 8 5 14 10 15 6 12
13 1 3 14 2 2 10 1 11 4 7 9 4 11 3 10 1 16 2 2 16 10 18 2 2 16 3 11 4 4 11 8 16 9 15 1 6 4 1 16 3 11 7 17 4 7 4 2 5 7 13 10 10 5 11 4 10 6 8 13 5 5 2 8 2 5 9 8 8 1 2 5 4 6 10 8 15 7 5 2 8 1 5 9
13 4 1 16 3 11 7 17 4 7 4 2 5 7 13 10 10 5 11 4 6 10 8 15 7 5 2 8 1 3 14 2 5 6 4 11 4 4 11 8 16 9 15 1 6 1 5 9 1 1 10 1 8 17 2 9 6 2 12 5 2 8 1 19 8 13 6 14 10 18 2 2 16 3 11 2 2 16 10 18

MK10:

20 15 3
12 2 6 5 2 5 2 7 11 6 11 1 2 5 4 8 10 3 18 4 10 9 7 2 7 9 1 7 4 1 8 7 14 9 12 4 7 3 4 13 8 8 2 6 5 3 8 1 19 9 13 10 19 2 16 5 2 16 10 9 3 12 4 11 5 15 2 9 10 10 5 3 7 5 2 8 4 7 4 1 6 6 13 5 11 10 7
13 2 7 11 6 11 4 2 16 10 9 5 9 8 16 2 6 5 2 5 2 2 11 1 9 2 3 12 7 15 4 4 11 10 14 5 10 7 15 4 3 8 1 12 5 5 13 11 5 3 8 1 19 9 13 10 19 2 16 3 4 13 8 8 2 6 4 8 10 3 18 4 10 9 7 4 1 16 5 11 10 17 3 6 2 9 10 10 5 2 5 11 2 11
11 4 3 8 1 12 5 5 13 11 2 2 11 1 9 2 7 9 1 7 2 6 5 2 5 4 1 6 6 13 5 11 10 7 2 9 10 10 5 5 3 8 1 19 9 13 10 19 2 16 4 8 10 3 18 4 10 9 7 4 2 16 10 9 5 9 8 16 2 3 12 7 15 2 2 5 9 19
11 4 4 11 10 14 5 10 7 15 5 3 8 1 19 9 13 10 19 2 16 1 5 15 1 2 5 2 9 10 10 5 2 7 11 6 11 4 1 16 5 11 10 17 3 6 2 10 13 6 11 2 2 5 9 19 3 4 13 8 8 2 6 4 8 10 3 18 4 10 9 7
14 2 7 11 6 11 2 9 10 10 5 4 5 11 7 8 10 11 2 16 2 10 13 6 11 4 1 16 5 11 10 17 3 6 2 7 9 1 7 4 3 8 1 12 5 5 13 11 1 2 5 4 2 16 10 9 5 9 8 16 3 1 15 2 19 9 9 4 1 6 6 13 5 11 10 7 2 2 11 1 9 4 4 11 10 14 5 10 7 15 5 3 8 1 19 9 13 10 19 2 16
11 3 1 15 2 19 9 9 2 7 9 1 7 4 8 10 3 18 4 10 9 7 2 2 5 9 19 4 5 11 7 8 10 11 2 16 4 1 6 6 13 5 11 10 7 2 7 11 6 11 1 2 5 3 7 5 2 8 4 7 4 3 8 1 12 5 5 13 11 1 5 15
14 1 2 5 2 7 11 6 11 2 2 11 1 9 2 9 10 10 5 4 8 10 3 18 4 10 9 7 3 1 15 2 19 9 9 3 7 5 2 8 4 7 4 2 16 10 9 5 9 8 16 4 1 6 6 13 5 11 10 7 1 5 15 4 7 13 10 19 6 18 4 8 4 3 8 1 12 5 5 13 11 4 1 16 5 11 10 17 3 6 2 7 9 1 7
13 4 8 10 3 18 4 10 9 7 2 10 13 6 11 4 5 11 7 8 10 11 2 16 3 4 13 8 8 2 6 5 2 16 10 9 3 12 4 11 5 15 3 1 15 2 19 9 9 4 3 8 1 12 5 5 13 11 3 7 5 2 8 4 7 4 7 13 10 19 6 18 4 8 4 1 6 6 13 5 11 10 7 2 6 5 2 5 4 1 16 5 11 10 17 3 6 4 2 16 10 9 5 9 8 16
11 2 7 11 6 11 3 7 5 2 8 4 7 4 8 10 3 18 4 10 9 7 2 9 10 10 5 4 2 16 10 9 5 9 8 16 3 4 13 8 8 2 6 4 7 13 10 19 6 18 4 8 5 2 16 10 9 3 12 4 11 5 15 4 1 8 7 14 9 12 4 7 2 6 5 2 5 2 2 11 1 9
12 2 9 10 10 5 1 5 15 2 2 5 9 19 3 1 15 2 19 9 9 5 2 16 10 9 3 12 4 11 5 15 4 1 6 6 13 5 11 10 7 4 2 16 10 9 5 9 8 16 4 1 16 5 11 10 17 3 6 2 6 5 2 5 2 3 12 7 15 4 4 11 10 14 5 10 7 15 4 8 10 3 18 4 10 9 7
10 5 2 16 10 9 3 12 4 11 5 15 4 5 11 7 8 10 11 2 16 4 7 13 10 19 6 18 4 8 2 9 10 10 5 1 5 15 2 2 11 1 9 3 4 13 8 8 2 6 2 2 5 9 19 4 8 10 3 18 4 10 9 7 4 1 16 5 11 10 17 3 6
11 2 10 13 6 11 1 5 15 2 9 10 10 5 4 1 8 7 14 9 12 4 7 4 3 8 1 12 5 5 13 11 3 4 13 8 8 2 6 3 7 5 2 8 4 7 1 2 5 4 1 6 6 13 5 11 10 7 4 2 16 10 9 5 9 8 16 2 7 9 1 7
11 3 7 5 2 8 4 7 2 2 5 9 19 4 1 8 7 14 9 12 4 7 4 5 11 7 8 10 11 2 16 3 1 15 2 19 9 9 4 1 16 5 11 10 17 3 6 2 6 5 2 5 2 7 11 6 11 5 3 8 1 19 9 13 10 19 2 16 4 8 10 3 18 4 10 9 7 3 4 13 8 8 2 6
10 2 5 11 2 11 4 8 10 3 18 4 10 9 7 2 7 9 1 7 2 6 5 2 5 4 3 8 1 12 5 5 13 11 1 5 15 2 9 10 10 5 4 1 16 5 11 10 17 3 6 3 4 13 8 8 2 6 2 3 12 7 15
12 4 8 10 3 18 4 10 9 7 1 5 15 3 1 15 2 19 9 9 4 7 13 10 19 6 18 4 8 4 2 16 10 9 5 9 8 16 4 1 8 7 14 9 12 4 7 3 7 5 2 8 4 7 2 10 13 6 11 2 9 10 10 5 2 3 12 7 15 2 6 5 2 5 4 3 8 1 12 5 5 13 11
14 2 7 11 6 11 1 5 15 2 2 5 9 19 4 1 8 7 14 9 12 4 7 1 2 5 4 3 8 1 12 5 5 13 11 3 4 13 8 8 2 6 3 1 15 2 19 9 9 4 4 11 10 14 5 10 7 15 2 6 5 2 5 2 9 10 10 5 2 5 11 2 11 5 3 8 1 19 9 13 10 19 2 16 2 10 13 6 11

13 4 7 13 10 19 6 18 4 8 5 2 16 10 9 3 12 4 11 5 15 3 4 13 8 8 2 6 2 7 11 6 11 2 10 13 6 11 4 2 16 10 9 5 9 8 16 4 8 10 3 18 4 10 9 7 4 3 8 1 12 5 5 13 11 2 6 5 2 5 2 7 9 1 7 4 1 16 5 11 10 17 3 6 2 2 5 9 19 5 3 8 1 19 9 13 10 19 2 16
11 4 7 13 10 19 6 18 4 8 4 1 6 6 13 5 11 10 7 4 2 16 10 9 5 9 8 16 2 10 13 6 11 2 2 5 9 19 2 5 11 2 11 5 2 16 10 9 3 12 4 11 5 15 2 6 5 2 5 3 1 15 2 19 9 9 1 2 5 4 4 11 10 14 5 10 7 15
13 2 2 5 9 19 2 6 5 2 5 3 4 13 8 8 2 6 2 7 9 1 7 2 3 12 7 15 1 5 15 4 1 16 5 11 10 17 3 6 3 1 15 2 19 9 9 2 10 13 6 11 2 2 11 1 9 3 7 5 2 8 4 7 4 3 8 1 12 5 5 13 11 4 5 11 7 8 10 11 2 16
13 4 1 16 5 11 10 17 3 6 3 1 15 2 19 9 9 4 3 8 1 12 5 5 13 11 2 2 5 9 19 4 2 16 10 9 5 9 8 16 1 5 15 4 5 11 7 8 10 11 2 16 4 8 10 3 18 4 10 9 7 2 7 11 6 11 4 1 8 7 14 9 12 4 7 4 7 13 10 19 6 18 4 8 2 3 12 7 15 2 7 9 1 7

Case 1 fuzzy processing time

10 10 2
4 10 1 5 8 11 2 4 7 9 3 10 13 17 4 4 6 8 5 6 9 11 6 5 7 10 7 6 9 12 8 4 6 9 9 8 10 13 10 5 8 11 10 1 6 9 12 2 4 7 10 3 3 6 9 4 3 5 8 5 6 7 9 6 5 6 8 7 9 13 16 8 7 10 12 9 4 7 10 10 5 7 10 10 1 9 11 14 2 3 5 7 3 5 7 10 4 3 5 7 5 4 7 9 6 5 8 10 7 5 7 10 8 11 15 18 9 8 10 13 10 6 8 10 10 1 5 8 11 2 9 12 15 3 8 11 15 4 6 9 11 5 7 10 13 6 13 15 18 7 15 19 22 8 7 9 13 9 9 13 17 10 7 9 13
4 10 1 10 14 17 2 4 7 10 3 4 8 11 4 5 6 9 5 6 9 11 6 5 8 11 7 5 8 10 8 7 10 12 9 7 9 11 10 5 8 10 10 1 9 11 15 2 5 8 9 3 6 9 10 4 7 10 12 5 5 7 9 6 5 8 11 7 7 9 12 8 5 7 9 9 8 11 13 10 9 12 15 10 1 5 8 9 2 4 7 9 3 6 8 11 4 7 8 10 5 7 9 11 6 4 8 10 7 5 7 10 8 6 8 12 9 7 8 10 10 8 9 10 10 1 7 8 10 2 9 11 14 3 8 10 13 4 11 14 17 5 13 17 20 6 7 10 12 7 8 11 12 8 6 9 11 9 5 8 12 10 6 10 13
4 10 1 3 4 5 2 4 5 6 3 2 3 6 4 6 7 9 5 7 8 10 6 7 9 10 7 4 5 7 8 4 5 6 9 5 7 8 10 6 8 9 10 1 3 5 6 2 7 9 12 3 6 9 11 4 7 8 11 5 8 10 13 6 5 6 8 7 7 10 13 8 7 9 12 9 6 9 11 10 5 9 12 10 1 10 14 17 2 5 7 10 3 10 13 17 4 9 13 17 5 8 11 15 6 6 9 12 7 5 8 11 8 6 9 12 9 10 12 14 10 7 9 13 10 1 4 7 10 2 3 5 9 3 5 9 12 4 6 8 12 5 9 11 14 6 5 9 12 7 6 10 13 8 19 24 28 9 5 8 10 10 7 10 12
4 10 1 3 5 6 2 4 7 10 3 5 8 10 4 5 7 10 5 6 9 11 6 7 9 11 7 4 7 10 8 3 5 8 9 4 7 9 10 10 11 13 10 1 3 4 5 2 4 7 8 3 7 9 12 4 5 8 10 5 6 8 11 6 3 5 8 7 4 7 8 8 5 8 9 9 11 13 16 10 5 7 9 10 1 2 4 6 2 7 9 12 3 4 5 7 4 5 8 10 5 3 5 8 6 4 5 7 7 9 12 15 8 7 9 13 9 6 8 11 10 8 11 15 10 1 5 8 11 2 9 12 14 3 8 11 13 4 6 9 12 5 5 8 11 6 7 10 13 7 6 9 11 8 5 8 11 9 7 9 12 10 5 7 10
4 10 1 3 6 8 2 4 5 7 3 8 9 11 4 7 10 14 5 4 6 9 6 3 6 8 7 5 8 10 8 9 12 14 9 5 6 8 10 7 9 13 10 1 1 3 4 2 5 6 8 3 7 9 10 4 3 5 8 5 5 8 10 6 5 7 9 7 7 9 12 8 8 10 13 9 4 6 9 10 5 6 8 10 1 8 11 14 2 7 10 12 3 6 7 8 4 5 8 12 5 4 7 10 6 6 9 11 7 8 11 15 8 6 9 13 9 6 8 9 10 5 8 12 10 1 8 10 13 2 7 9 12 3 8 10 12 4 6 9 12 5 11 14 18 6 5 8 10 7 4 7 10 8 6 8 11 9 8 10 13 10 5 8 9
4 10 1 8 9 10 2 5 9 12 3 2 3 5 4 7 9 10 5 8 11 15 6 4 6 9 7 3 5 8 8 7 8 10 9 8 9 10 10 4 5 7 10 1 6 9 12 2 7 10 12 3 8 11 13 4 5 7 10 5 8 11 13 6 9 12 14 7 6 8 10 8 5 7 9 9 5 7 9 10 5 8 9 10 1 2 3 4 2 4 7 8 3 5 8 10 4 3 5 6 5 4 7 8 6 6 9 11 7 7 10 12 8 5 8 10 9 6 8 11 10 4 5 7 10 1 3 4 5 2 10 13 17 3 6 8 11 4 7 10 13 5 4 7 8 6 5 8 10 7 3 6 8 8 3 4 5 9 10 14 17 10 3 5 7
4 10 1 2 4 6 2 3 5 8 3 4 6 8 4 8 10 13 5 4 6 9 6 3 5 8 7 7 8 10 8 6 8 11 9 1 2 4 10 5 6 7 10 1 9 11 14 2 6 8 9 3 7 9 10 4 8 12 15 5 9 13 17 6 5 9 13 7 6 8 12 8 5 8 11 9 7 9 10 10 7 10 12 10 1 5 8 10 2 4 7 8 3 10 12 15 4 6 9 11 5 6 9 11 6 5 7 11 7 6 9 12 8 7 10 13 9 6 8 10 10 15 19 23 10 1 4 7 10 2 5 8 10 3 6 9 12 4 4 7 9 5 5 8 11 6 4 7 9 7 9 12 16 8 8 11 15 9 5 7 10 10 4 7 10
4 10 1 9 12 15 2 6 8 11 3 4 7 10 4 5 8 11 5 10 13 15 6 9 13 16 7 8 11 15 8 5 8 10 9 6 8 11 10 4 7 10 10 1 5 6 8 2 2 3 5 3 3 5 8 4 6 8 11 5 7 10 13 6 4 7 8 7 8 11 14 8 6 9 13 9 3 5 8 10 15 19 24 10 1 5 8 10 2 9 13 16 3 7 10 14 4 6 10 13 5 7 10 12 6 8 11 14 7 8 11 14 8 7 9 13 9 10 13 15 10 7 9 12 10 1 3 4 5 2 8 11 13 3 5 7 10 4 7 9 11 5 8 9 11 6 5 7 10 7 10 12 15 8 3 5 6 9 5 6 8 10 5 8 10
4 10 1 7 9 11 2 10 14 17 3 9 12 16 4 8 10 12 5 7 9 11 6 8 11 14 7 10 13 16 8 8 11 15 9 7 10 14 10 5 7 9 10 1 4 6 7 2 5 8 9 3 7 8 10 4 4 7 9 5 35 39 44 6 4 7 9 7 7 10 13 8 8 11 14 9 7 9 13 10 10 12 14 10 1 8 10 13 2 7 8 9 3 6 8 11 4 9 12 14 5 5 6 8 6 7 9 12 7 8 11 15 8 6 8 11 9 6 8 9 10 7 10 12 10 1 2 4 5 2 7 10 13 3 8 10 12 4 6 9 12 5 3 5 8 6 6 8 11 7 6 7 9 8 13 17 20 9 6 7 9 10 2 4 5
4 10 1 3 4 6 2 5 7 9 3 5 7 9 4 8 12 15 5 7 10 12 6 7 9 12 7 6 9 12 8 7 8 10 9 19 23 26 10 4 5 8 10 1 9 12 17 2 6 8 10 3 5 8 11 4 4 7 9 5 5 8 11 6 6 9 12 7 7 10 13 8 6 7 9 9 4 5 6 10 10 13 17 10 1 6 8 9 2 7 8 10 3 8 10 11 4 9 11 12 5 10 13 15 6 6 8 9 7 11 15 18 8 10 15 19 9 7 8 10 10 9 12 14 10 1 10 14 17 2 5 8 10 3 9 12 13 4 6 8 9 5 7 9 10 6 5 6 8 7 8 11 13 8 5 6 8 9 7 8 11 10 9 10 12

Case 2 fuzzy processing time

10 10 2
4 10 1 7 10 14 2 6 9 11 3 10 13 17 4 7 9 12 5 8 11 15 6 5 8 11 7 8 11 15 8 9 12 16 9 9 12 16 10 8 12 16 10 1 16 20 25 2 14 19 23 3 13 17 21 4 12 15 19 5 16 19 23 6 15 16 19 7 9 15 19 8 10 15 19 9 14 18 22 10 14 18 22 10 1 10 16 17 2 8 10 13 3 10 12 16 4 8 11 13 5 9 12 14 6 10 13 15 7 9 12 14 8 15 20 23 9 13 15 18 10 10 13 15 10 1 8 12 15 2 13 16 19 3 12 15 19 4 10 13 14 5 11 14 17 6 16 19 23 7 18 22 26 8 10 13 14 9 12 17 21 10 11 13 17
4 10 1 11 15 18 2 5 8 10 3 5 9 13 4 6 7 10 5 7 10 12 6 6 9 12 7 6 9 12 8 8 11 13 9 8 10 12 10 6 9 11 10 1 10 12 14 2 5 8 9 3 5 9 10 4 7 10 13 5 5 7 9 6 6 8 11 7 7 10 13 8 5 7 9 9 8 11 14 10 8 12 14 10 1 8 9 10 2 10 13 14 3 10 12 15 4 11 13 16 5 10 11 13 6 8 12 15 7 8 11 13 8 6 9 12 9 6 8 12 10 8 9 10 10 1 6 9 11 2 9 11 14 3 8 10 13 4 11 14 17 5 7 10 12 6 8 11 12 7 13 17 20 8 5 8 12 9 6 10 13 10 7 8 10

4 10 1 6 7 9 2 4 5 6 3 6 9 12 4 7 8 10 5 6 7 9 6 8 10 13 7 5 7 10 8 9 11 13 9 9 12 14 10 4 5 6 10 1 9 12 15 2 6 8 12 3 8 10 13
4 7 8 11 5 8 10 13 6 5 6 8 7 7 10 13 8 5 9 12 9 6 9 11 10 5 9 12 10 1 5 6 8 2 8 10 13 3 10 14 19 4 7 11 14 5 6 8 11 6 6 9 13
7 9 13 17 8 8 10 13 9 7 11 14 10 6 9 12 10 1 10 14 18 2 11 15 20 3 14 19 24 4 13 17 20 5 9 15 21 6 9 14 18 7 10 14 18 8 15
19 26 9 23 28 33 10 8 12 17
4 10 1 9 12 16 2 9 13 17 3 11 15 20 4 8 13 19 5 12 17 24 6 10 15 19 7 10 15 19 8 13 16 20 9 10 16 21 10 12 16 23 10 1 11
15 20 2 10 13 17 3 13 18 23 4 11 17 23 5 11 16 23 6 13 18 24 7 11 15 20 8 10 12 16 9 11 17 24 10 14 19 23 10 1 15 21 26
2 10 11 17 3 9 13 17 4 12 18 23 5 10 11 17 6 11 16 20 7 13 17 21 8 11 15 20 9 13 19 24 10 10 15 19 10 1 3 4 5 2 6 7 10 3 8
9 11 4 9 11 14 5 5 8 11 6 8 11 16 7 7 10 15 8 8 12 16 9 6 7 10 10 9 12 16
4 10 1 8 11 15 2 7 10 12 3 7 9 11 4 6 9 12 5 10 13 17 6 7 10 12 7 9 11 15 8 7 9 12 9 11 14 18 10 5 8 11 10 1 6 8 9 2 12 15
19 3 7 10 13 4 6 10 12 5 7 9 12 6 10 13 18 7 7 11 15 8 6 10 12 9 8 11 15 10 8 12 17 10 1 5 6 8 2 6 9 11 3 9 10 13 4 7 9 10 5
5 8 11 6 11 14 18 7 8 10 13 8 9 12 16 9 7 8 10 10 7 8 10 10 1 4 7 9 2 9 10 13 3 5 8 12 4 7 10 14 5 6 8 11 6 7 9 12 7 6 8 11 8
9 10 12 9 5 9 12 10 6 8 11
4 10 1 10 13 17 2 6 8 9 3 7 9 11 4 8 10 13 5 8 10 13 6 8 11 15 7 6 10 13 8 7 9 12 9 9 11 13 10 7 9 11 10 1 10 14 19 2 11 15
20 3 12 16 22 4 9 13 17 5 8 13 17 6 10 13 18 7 11 15 21 8 12 19 25 9 8 13 17 10 10 14 17 10 1 7 10 12 2 12 17 23 3 12 19
25 4 10 15 20 5 9 16 27 6 11 15 19 7 13 18 24 8 11 16 23 9 10 15 20 10 10 15 19 10 1 3 4 5 2 2 3 5 3 7 8 10 4 5 8 10 5 3 5
8 6 6 8 10 7 7 10 11 8 6 8 10 9 4 7 9 10 11 14 18
4 10 1 8 11 13 2 7 9 11 3 9 13 17 4 10 13 17 5 10 14 19 6 7 10 13 7 9 13 17 8 8 12 16 9 9 11 15 10 8 10 14 10 1 6 9 11 2 10
13 18 3 7 8 10 4 8 9 11 5 8 9 11 6 11 14 18 7 10 13 16 8 7 10 13 9 10 12 14 10 7 9 12 10 1 5 6 8 2 4 5 7 3 6 8 11 4 8 10 13
5 7 10 12 6 5 8 10 7 8 10 12 8 9 11 15 9 5 8 10 10 11 15 20 10 1 11 14 18 2 10 13 17 3 8 12 17 4 7 10 14 5 9 14 20 6 8 12
17 7 10 15 21 8 9 14 19 9 10 14 19 10 8 10 13
4 10 1 7 10 12 2 8 11 15 3 9 14 18 4 10 15 19 5 11 14 18 6 9 15 21 7 8 12 17 8 7 10 12 9 9 13 18 10 8 12 17 10 1 10 14 19
2 9 12 15 3 12 17 21 4 13 18 23 5 10 13 18 6 11 15 20 7 11 16 19 8 11 16 19 9 10 13 18 10 12 16 21 10 1 4 6 8 2 8 11 15 3
7 11 14 4 8 10 13 5 6 8 10 6 12 17 23 7 7 10 12 8 10 13 17 9 9 14 18 10 6 8 10 10 1 7 8 10 2 4 5 7 3 3 5 8 4 7 9 11 5 6 8 10
6 5 7 9 7 6 8 11 8 7 10 12 9 7 10 12 10 8 10 13
4 10 1 5 8 10 2 10 13 18 3 4 7 9 4 8 10 13 5 6 9 12 6 5 7 10 7 9 12 16 8 7 9 13 9 6 8 12 10 5 7 10 10 1 2 3 5 2 4 6 7 3 8 10
11 4 7 9 10 5 5 6 8 6 6 7 9 7 9 12 14 8 8 11 13 9 4 6 7 10 5 7 10 10 1 10 14 17 2 6 9 11 3 7 11 14 4 5 9 12 5 5 8 11 6 8 10 14
7 12 16 20 8 6 8 11 9 7 10 13 10 10 13 15 10 1 9 11 15 2 8 11 14 3 7 10 12 4 10 13 17 5 8 10 13 6 11 16 21 7 7 9 11 8 13 17
22 9 10 14 17 10 8 9 12
4 10 1 5 8 11 2 7 9 13 3 6 9 11 4 10 13 17 5 7 10 13 6 8 11 15 7 9 12 16 8 9 14 17 9 10 14 18 10 7 11 14 10 1 13 17 22 2 7
10 12 3 8 10 13 4 6 10 13 5 6 9 11 6 9 13 16 7 7 9 10 8 8 11 15 9 7 10 12 10 9 13 17 10 1 5 8 10 2 11 15 19 3 7 9 11 4 6 8
11 5 8 10 13 6 9 11 15 7 7 9 11 8 12 17 21 9 5 9 12 10 10 13 17 10 1 3 5 8 2 4 5 7 3 9 10 13 4 5 8 10 5 6 8 12 6 7 9 12 7 8
11 14 8 10 15 19 9 4 7 9 10 11 16 20

Case 3 fuzzy processing time

10 10 2
4 10 1 3 4 6 2 7 9 12 3 5 7 10 4 8 10 13 5 9 11 14 6 5 8 11 7 10 14 18 8 6 9 12 9 7 9 10 10 8 11 15 10 1 5 7 9 2 8 10 13 3 7
8 10 4 6 8 9 5 8 10 11 6 9 11 14 7 6 7 10 8 5 7 10 9 7 9 10 10 10 14 18 10 1 7 9 11 2 8 10 13 3 7 8 11 4 6 8 9 5 5 7 8 6 11 14
17 7 5 8 11 8 7 9 12 9 6 8 11 10 5 7 8 10 1 8 10 13 2 5 8 10 3 9 13 17 4 6 8 11 5 6 9 12 6 7 10 13 7 5 9 12 8 6 9 12 9 12 17
21 10 8 12 16
6 10 1 3 4 5 2 6 8 9 3 9 10 13 4 5 7 8 5 6 9 11 6 7 10 12 7 10 13 15 8 5 7 10 9 10 13 17 10 6 9 12 10 1 10 13 17 2 8 11 14 3
7 10 12 4 6 9 11 5 6 9 11 6 9 12 16 7 8 11 15 8 7 11 14 9 6 10 13 10 15 19 24 10 1 4 5 7 2 8 10 13 3 6 8 10 4 5 8 9 5 6 9 11
6 6 8 9 7 7 10 12 8 11 14 18 9 6 9 12 10 9 11 14 10 1 6 9 11 2 5 7 8 3 4 6 7 4 3 5 8 5 6 8 9 6 7 10 12 7 8 10 13 8 5 8 10 9 6
9 11 10 7 9 12 10 1 5 6 7 2 9 10 14 3 8 10 13 4 7 9 10 5 4 5 7 6 6 8 10 7 5 8 10 8 6 8 9 9 9 11 15 10 3 5 8 10 1 8 11 15 2 7
10 13 3 6 9 11 4 5 8 10 5 7 9 12 6 6 8 11 7 8 10 13 8 10 13 17 9 5 9 13 10 7 8 10
4 10 1 7 9 12 2 6 7 9 3 4 6 9 4 8 11 14 5 9 13 15 6 5 7 10 7 6 9 12 8 7 8 11 9 9 11 13 10 5 6 7 10 1 5 7 9 2 7 9 12 3 4 7 9 4 8
9 12 5 7 8 10 6 6 9 10 7 5 6 8 8 3 5 8 9 8 10 11 10 6 8 9 10 1 10 14 18 2 9 13 17 3 12 16 21 4 8 11 15 5 7 11 14 6 15 19 24
7 11 15 19 8 16 23 28 9 9 12 16 10 8 11 14 10 1 6 9 11 2 5 8 10 3 8 9 11 4 7 10 13 5 9 11 15 6 10 13 17 7 11 14 17 8 8 10
13 9 11 15 19 10 6 10 12
6 10 1 8 11 14 2 7 9 12 3 6 8 11 4 5 8 11 5 9 11 15 6 7 10 13 7 6 10 13 8 9 12 15 9 5 7 10 10 11 16 20 10 1 4 5 7 2 3 4 6 3 5
8 10 4 4 6 9 5 2 3 5 6 7 9 11 7 8 9 12 8 5 7 10 9 8 10 11 10 7 8 9 10 1 8 10 14 2 9 12 16 3 10 14 18 4 7 10 13 5 6 9 12 6 5 8
10 7 6 8 11 8 7 11 14 9 8 9 10 10 6 9 12 10 1 6 8 10 2 5 7 9 3 7 10 12 4 8 10 13 5 9 11 15 6 6 9 11 7 11 15 18 8 10 13 18 9
7 9 12 10 5 8 11 10 1 13 17 21 2 8 11 14 3 7 11 14 4 9 12 15 5 10 14 18 6 8 10 13 7 7 10 13 8 14 18 23 9 8 11 15 10 9 11 15
10 1 7 10 12 2 6 8 11 3 5 8 10 4 8 10 13 5 9 12 17 6 7 9 13 7 11 15 19 8 8 11 15 9 6 9 12 10 13 17 20
5 10 1 9 12 16 2 7 10 13 3 6 9 12 4 8 11 14 5 7 9 10 6 8 10 11 7 13 15 16 8 10 13 15 9 9 12 16 10 8 10 11 10 1 3 4 6 2 7 8
10 3 8 10 11 4 9 11 15 5 5 7 8 6 6 8 10 7 7 10 12 8 5 8 10 9 9 12 14 10 10 13 18 10 1 7 9 12 2 6 9 11 3 8 11 15 4 9 12 16 5
9 13 17 6 10 14 16 7 7 10 13 8 8 10 13 9 11 15 19 10 6 9 11 10 1 6 9 11 2 5 8 11 3 4 8 11 4 7 11 14 5 8 11 15 6 7 9 12 7 9
12 16 8 10 14 18 9 7 10 13 10 8 11 14 10 1 2 4 6 2 1 2 4 3 4 6 8 4 7 8 9 5 8 10 12 6 5 8 10 7 3 5 8 8 4 6 9 9 7 9 11 10 6 9 11
5 10 1 7 10 13 2 5 8 11 3 8 10 13 4 6 8 11 5 7 9 11 6 8 11 14 7 9 10 12 8 6 8 10 9 7 10 13 10 10 13 16 10 1 6 8 9 2 5 6 7 3 4
6 8 4 3 5 8 5 8 10 13 6 4 5 6 7 6 8 10 8 7 9 11 9 5 8 9 10 6 9 11 10 1 9 11 14 2 10 14 18 3 8 11 15 4 7 10 13 5 8 10 13 6 9 12
15 7 7 9 11 8 6 9 12 9 10 15 20 10 8 11 15 10 1 4 7 9 2 5 8 10 3 6 9 12 4 7 10 12 5 8 11 15 6 9 10 13 7 6 8 11 8 10 14 17 9
8 10 14 10 5 7 10 10 1 8 10 13 2 7 9 11 3 7 10 13 4 8 11 15 5 6 9 12 6 5 7 9 7 6 8 10 8 10 13 15 9 8 10 13 10 11 14 17

5 10 1 6 8 9 2 7 10 12 3 8 11 13 4 9 13 17 5 5 8 11 6 8 11 14 7 7 9 12 8 6 9 13 9 5 9 12 10 6 10 13 10 1 4 5 7 2 5 6 7 3 6 7 9
4 7 9 11 5 8 10 11 6 9 11 13 7 7 10 12 8 5 7 10 9 4 6 8 10 8 11 13 10 1 8 11 14 2 7 8 10 3 9 11 12 4 10 13 14 5 6 9 10 6 7 9
10 7 8 10 13 8 11 14 18 9 6 10 13 10 13 17 21 10 1 11 13 17 2 5 8 11 3 4 7 10 4 6 8 11 5 7 10 12 6 8 10 13 7 9 11 14 8 10
14 17 9 8 11 15 10 6 9 11 10 1 13 17 21 2 7 10 13 3 8 11 14 4 9 11 13 5 10 14 17 6 14 19 22 7 6 9 12 8 5 8 12 9 7 11 14 10
8 11 15
5 10 1 6 8 11 2 5 8 12 3 7 11 15 4 8 10 13 5 9 12 16 6 10 14 18 7 8 11 15 8 6 9 12 9 7 10 13 10 8 9 11 10 1 15 19 24 2 11 15
19 3 9 10 13 4 12 15 18 5 8 12 17 6 7 11 14 7 17 21 26 8 8 11 15 9 10 14 17 10 9 12 14 10 1 7 10 12 2 6 9 11 3 8 10 13 4 9
12 14 5 7 11 14 6 10 13 17 7 7 10 12 8 11 15 19 9 9 11 13 10 8 10 13 10 1 8 10 13 2 10 14 17 3 6 9 12 4 5 8 12 5 7 10 13 6
5 8 12 7 12 15 19 8 6 9 11 9 7 11 14 10 9 12 15 10 1 5 6 7 2 6 8 11 3 7 10 12 4 5 8 12 5 9 11 14 6 10 14 18 7 6 9 12 8 7 11
14 9 8 11 15 10 5 7 10
5 10 1 10 14 17 2 9 12 15 3 8 11 15 4 7 11 14 5 6 9 12 6 7 10 13 7 12 16 20 8 13 18 23 9 7 9 13 10 8 10 13 10 1 7 9 11 2 5
8 10 3 7 10 13 4 9 12 16 5 6 9 12 6 8 11 14 7 10 14 16 8 9 13 17 9 6 10 13 10 7 9 13 10 1 9 13 17 2 6 9 12 3 10 14 18 4 8 11
15 5 7 10 13 6 6 9 11 7 12 17 21 8 8 10 13 9 9 11 14 10 11 14 17 10 1 4 5 7 2 5 7 9 3 6 8 9 4 7 9 10 5 8 10 13 6 4 7 9 7 5 6
8 8 2 3 5 9 6 9 11 10 7 10 12 10 1 8 10 13 2 9 11 14 3 7 9 11 4 6 9 13 5 10 13 17 6 5 7 10 7 8 11 15 8 9 12 16 9 6 9 12 10 7
10 12
5 10 1 3 5 8 2 7 9 11 3 4 5 7 4 5 8 11 5 6 9 12 6 8 10 13 7 6 8 11 8 10 14 17 9 7 11 14 10 8 10 11 10 1 8 11 15 2 9 13 17 3 7
11 14 4 6 10 13 5 5 9 12 6 7 10 13 7 6 9 13 8 10 14 17 9 8 10 13 10 15 19 24 10 1 7 9 11 2 13 17 22 3 7 9 13 4 8 10 13 5 9
11 14 6 8 11 15 7 5 8 12 8 6 9 13 9 10 12 15 10 7 10 12 10 1 6 9 12 2 5 8 10 3 8 10 13 4 9 11 15 5 7 9 12 6 8 11 13 7 5 9 12
8 7 11 14 9 10 14 17 10 11 15 19 10 1 4 5 7 2 7 8 9 3 6 8 10 4 3 5 8 5 4 6 9 6 5 7 10 7 6 9 12 8 7 10 12 9 5 8 11 10 8 10 13

Case 4 fuzzy processing time

10 10 2
4 10 1 9 13 17 2 5 6 8 3 3 4 6 4 6 8 9 5 2 3 5 6 4 5 7 7 7 9 11 8 6 8 11 9 5 8 10 10 5 7 9 10 1 5 7 10 2 8 10 13 3 6 9 11 4 7 10
12 5 5 6 8 6 4 6 9 7 7 9 11 8 3 5 8 9 4 7 9 10 2 3 5 10 1 1 2 4 2 4 5 7 3 6 8 10 4 7 8 10 5 6 8 9 6 9 10 12 7 3 5 8 8 5 6 8 9 7 9
11 10 6 8 11 10 1 3 4 5 2 5 8 11 3 7 9 10 4 4 6 9 5 6 8 11 6 4 7 9 7 6 9 12 8 5 8 10 9 3 5 8 10 4 6 7
6 10 1 5 8 10 2 6 9 11 3 7 10 12 4 4 7 9 5 7 8 10 6 8 9 11 7 4 6 9 8 9 11 14 9 6 9 13 10 5 7 10 10 1 4 6 9 2 3 4 6 3 5 7 9 4 6
8 11 5 7 10 12 6 5 8 11 7 4 5 6 8 3 5 7 9 2 4 7 10 5 7 10 10 1 1 2 4 2 4 7 9 3 6 8 11 4 5 6 8 5 4 6 8 6 2 3 5 7 4 6 8 8 5 7 9 9 4
6 10 10 3 5 8 10 1 7 8 11 2 4 7 10 3 5 8 12 4 6 9 12 5 7 8 10 6 5 7 10 7 10 13 17 8 8 10 13 9 5 7 9 10 11 14 17 10 1 8 10 14
2 5 7 11 3 6 8 12 4 7 9 11 5 5 9 12 6 4 7 9 7 5 6 9 8 3 5 8 9 4 7 10 10 6 9 12 10 1 6 9 11 2 4 7 11 3 5 8 12 4 4 6 8 5 7 8 11 6
8 9 11 7 9 11 13 8 7 9 12 9 6 9 11 10 6 7 10
4 10 1 3 4 6 2 5 7 8 3 6 8 9 4 7 9 11 5 8 10 11 6 6 8 11 7 4 5 7 8 6 9 12 9 5 9 12 10 10 13 17 10 1 7 8 10 2 6 9 11 3 6 8 12 4
5 7 10 5 8 10 12 6 7 10 13 7 6 9 12 8 5 8 12 9 4 7 10 10 7 8 10 10 1 5 7 10 2 3 5 8 3 2 3 5 4 4 7 8 5 5 8 10 6 7 8 10 7 4 6 9 8
7 9 12 9 8 10 13 10 9 11 15 10 1 8 11 15 2 7 8 10 3 6 9 11 4 5 9 12 5 4 8 10 6 3 5 9 7 5 8 12 8 3 4 7 9 6 7 9 10 7 10 11
6 10 1 6 9 10 2 8 10 11 3 5 7 8 4 6 7 9 5 7 8 10 6 9 10 12 7 4 6 7 8 5 8 9 9 7 10 11 10 4 6 9 10 1 2 3 5 2 4 7 8 3 6 8 9 4 7 9
11 5 1 2 4 6 3 5 7 7 4 6 9 8 8 10 11 9 5 7 8 10 6 8 10 10 1 3 5 8 2 2 3 5 3 7 8 9 4 5 6 8 5 4 6 8 6 5 8 10 7 6 9 11 8 3 4 6 9 2 4
5 10 6 7 10 10 1 6 8 9 2 7 10 11 3 6 9 11 4 5 8 10 5 7 8 11 6 5 8 12 7 9 11 15 8 14 17 21 9 5 9 12 10 5 7 10 10 1 4 6 9 2 9 12
16 3 7 9 10 4 6 8 11 5 5 9 12 6 7 8 11 7 4 7 10 8 8 11 15 9 6 8 11 10 4 5 7 10 1 9 10 14 2 5 8 12 3 7 11 15 4 6 9 12 5 5 9 12
6 7 10 13 7 8 10 14 8 4 8 11 9 11 14 17 10 9 11 14
5 10 1 5 7 10 2 8 10 13 3 9 11 15 4 6 10 13 5 7 10 12 6 8 11 15 7 7 9 12 8 6 8 10 9 5 8 11 10 4 7 9 10 1 4 7 9 2 7 8 10 3 5 8
11 4 6 9 10 5 5 8 10 6 7 8 10 7 4 6 9 8 7 9 12 9 8 10 13 10 9 11 15 10 1 8 11 14 2 6 8 11 3 6 9 12 4 7 10 12 5 5 8 11 6 9 11
15 7 7 11 14 8 6 9 12 9 10 14 17 10 7 9 13 10 1 6 8 11 2 9 10 11 3 8 10 13 4 7 11 14 5 7 8 10 6 12 15 19 7 10 14 17 8 7 10
11 9 5 9 12 10 6 9 12 10 1 3 5 8 2 5 6 8 3 1 2 3 4 2 4 7 5 4 5 8 6 6 8 9 7 2 3 6 8 7 8 10 9 5 7 10 10 4 6 9
5 10 1 6 9 12 2 7 10 13 3 9 12 16 4 5 9 12 5 7 8 11 6 5 7 10 7 8 11 15 8 4 7 10 9 8 9 11 10 5 6 8 10 1 10 14 17 2 8 11 15 3 6
9 13 4 8 10 13 5 7 11 13 6 10 12 15 7 13 18 24 8 7 9 11 9 6 10 13 10 8 10 13 10 1 3 4 7 2 6 8 9 3 4 5 8 4 7 8 11 5 5 7 10 6 2
4 7 7 6 7 10 8 7 8 11 9 4 6 9 10 7 10 12 10 1 5 8 11 2 7 9 12 3 8 11 14 4 9 11 15 5 5 7 10 6 6 10 13 7 7 10 11 8 11 14 18 9 6
9 12 10 5 9 13 10 1 4 7 9 2 6 7 10 3 8 10 11 4 9 11 14 5 6 7 9 6 5 7 9 7 10 15 18 8 8 11 15 9 7 11 14 10 7 10 12
5 10 1 8 10 14 2 6 9 12 3 7 10 12 4 5 9 12 5 7 10 13 6 5 8 10 7 4 7 10 8 6 9 13 9 8 10 14 10 6 9 11 10 1 5 6 8 2 4 7 9 3 6 10
13 4 7 10 12 5 4 6 8 6 5 8 10 7 9 10 14 8 7 9 12 9 5 9 13 10 7 8 11 10 1 10 13 17 2 8 11 15 3 7 10 13 4 6 9 12 5 5 9 12 6 4 7
9 7 5 8 11 8 5 8 10 9 7 11 14 10 6 10 13 10 1 2 3 4 2 7 9 11 3 4 5 8 4 5 7 10 5 3 5 9 6 6 9 11 7 4 7 9 8 8 11 15 9 4 8 11 10 5
8 11 10 1 5 8 10 2 10 13 16 3 7 8 10 4 4 5 8 5 6 9 11 6 3 6 9 7 5 8 10 8 7 10 12 9 4 7 9 10 9 12 17
5 10 1 4 8 11 2 9 10 14 3 3 6 8 4 7 8 10 5 5 6 9 6 6 10 13 7 10 13 18 8 7 10 13 9 6 9 12 10 4 7 9 10 1 8 10 14 2 7 9 11 3 5 8
11 4 10 12 16 5 4 7 10 6 7 10 13 7 4 6 9 8 4 5 7 9 9 10 13 10 12 15 19 10 1 5 6 8 2 9 11 15 3 4 7 10 4 10 12 18 5 3 6 9 6 8
10 13 7 4 7 9 8 7 10 12 9 11 13 17 10 6 9 11 10 1 2 3 4 2 4 5 7 3 5 7 9 4 1 2 4 5 3 5 8 6 6 8 11 7 7 10 12 8 3 5 9 9 6 7 10 10
4 6 10 10 1 8 10 13 2 10 12 16 3 7 10 13 4 6 10 13 5 5 9 12 6 4 7 10 7 7 9 11 8 11 14 19 9 5 7 10 10 6 9 13
5 10 1 6 8 10 2 7 9 11 3 4 6 8 4 5 7 9 5 6 7 8 6 4 5 6 7 8 10 12 8 9 11 13 9 7 8 10 10 5 9 12 10 1 5 6 8 2 2 3 5 3 3 5 8 4 6 8
11 5 7 9 12 6 4 5 7 7 6 7 10 8 4 6 9 9 3 5 7 10 7 8 10 10 1 7 9 13 2 6 10 13 3 5 9 12 4 6 8 11 5 5 8 12 6 4 7 11 7 8 11 15 8 3
7 10 9 5 7 11 10 8 10 14 10 1 9 10 12 2 11 14 17 3 7 9 11 4 6 8 11 5 5 7 10 6 4 7 9 7 3 5 8 8 7 8 11 9 6 9 11 10 5 7 11 10 1
3 5 8 2 10 13 17 3 4 7 9 4 6 8 10 5 9 12 17 6 8 10 13 7 5 7 10 8 10 11 14 9 6 10 13 10 7 8 9
5 10 1 11 15 19 2 10 13 17 3 9 10 13 4 8 10 13 5 7 10 11 6 6 7 9 7 5 7 10 8 7 11 14 9 9 12 16 10 6 7 10 10 1 8 11 14 2 6 9
13 3 5 8 11 4 9 10 13 5 5 7 10 6 4 7 10 7 6 8 11 8 11 14 18 9 6 8 9 10 9 12 16 10 1 5 7 10 2 4 5 7 3 6 8 11 4 7 10 12 5 6 9 13

6 5 6 8 7 3 5 8 8 6 7 10 9 8 11 15 10 5 7 11 10 1 4 5 7 2 5 7 10 3 6 8 11 4 3 6 9 5 7 9 12 6 4 6 9 7 8 9 11 8 7 9 11 9 5 8 10 10
2 3 4 10 1 7 9 12 2 6 8 10 3 5 7 10 4 4 7 9 5 10 12 15 6 6 7 9 7 5 6 8 8 7 10 12 9 4 5 7 10 13 15 18

Case 5 fuzzy processing time

15 10 2
5 10 1 3 4 7 2 2 4 7 3 5 7 10 4 6 8 11 5 8 10 13 6 1 2 4 7 7 9 12 8 6 7 9 9 11 14 18 10 4 7 9 10 1 8 10 14 2 7 9 12 3 5 8 11 4
6 9 12 5 4 5 7 6 3 5 8 7 7 10 12 8 4 7 10 9 5 6 8 10 2 3 5 10 1 5 8 11 2 4 8 10 3 8 11 15 4 7 11 14 5 5 7 11 6 6 8 12 7 9 12 16
8 7 9 10 9 6 7 10 10 10 11 15 10 1 4 5 8 2 2 3 5 3 3 5 7 4 6 8 9 5 7 10 12 6 8 11 15 7 5 7 10 8 5 6 8 9 7 8 10 10 4 6 9 10 1 7
9 13 2 5 8 9 3 4 7 10 4 6 9 11 5 3 6 9 6 7 10 13 7 4 5 6 8 12 15 20 9 8 11 15 10 6 8 12
6 10 1 3 4 5 2 4 7 8 3 6 9 11 4 7 10 12 5 5 7 10 6 3 5 8 7 2 4 6 8 7 8 9 9 4 7 9 10 7 9 10 10 1 7 9 13 2 6 8 12 3 7 11 13 4 5 9
12 5 6 10 13 6 8 12 14 7 5 7 11 8 3 6 10 9 5 8 13 10 6 8 13 10 1 6 7 10 2 7 10 12 3 5 8 10 4 4 7 9 5 3 6 8 6 5 6 9 7 2 4 7 8 4
8 11 9 6 10 13 10 4 7 11 10 1 2 3 5 2 1 2 4 3 3 5 7 4 4 5 8 5 5 7 8 6 6 9 10 7 4 7 10 8 6 8 11 9 7 9 13 10 6 9 12 10 1 10 12 15
2 9 10 14 3 7 11 14 4 6 9 12 5 5 8 12 6 4 8 10 7 8 10 13 8 6 7 10 9 5 9 12 10 7 9 13 10 1 8 9 11 2 5 6 7 3 7 8 10 4 4 6 8 5 4
7 8 6 10 11 14 7 12 14 17 8 6 8 11 9 5 7 9 10 6 9 12
5 10 1 6 8 11 2 9 12 16 3 7 9 11 4 4 5 7 5 8 11 15 6 5 8 11 7 4 7 10 8 8 9 11 9 6 8 11 10 6 7 9 10 1 5 7 10 2 6 9 12 3 7 10 13
4 8 11 15 5 9 13 17 6 10 14 18 7 5 8 11 8 4 7 10 9 6 9 11 10 7 8 10 10 1 4 5 8 2 7 9 11 3 3 5 8 4 6 9 12 5 4 6 10 6 5 8 12 7 6
10 13 8 7 11 14 9 6 8 12 10 4 6 7 10 1 7 10 13 2 6 8 12 3 5 7 10 4 4 6 7 5 6 10 13 6 7 8 10 7 9 12 16 8 5 6 8 9 4 7 9 10 6 9
12 10 1 10 13 16 2 4 7 9 3 10 12 13 4 5 6 8 5 6 8 10 6 7 8 12 7 10 12 15 8 5 6 8 9 3 5 8 10 10 14 18
6 10 1 3 5 6 2 8 10 13 3 5 8 11 4 9 12 16 5 5 7 10 6 7 11 13 7 9 13 15 8 8 10 13 9 2 4 7 10 4 7 10 10 1 9 12 15 2 10 15 18 3
7 11 14 4 5 8 9 5 4 7 8 6 5 7 10 7 6 10 13 8 8 9 13 9 7 10 12 10 8 11 14 10 1 5 8 9 2 6 8 11 3 8 11 15 4 7 10 14 5 4 7 10 6 3
5 8 7 6 8 10 8 10 14 17 9 6 9 12 10 7 8 12 10 1 4 7 10 2 6 10 13 3 7 11 12 4 9 12 15 5 5 8 11 6 4 7 11 7 8 10 13 8 6 7 8 9 5
8 9 10 7 8 11 10 1 11 15 19 2 8 11 15 3 7 11 13 4 6 10 12 5 10 14 18 6 9 11 14 7 7 11 14 8 5 8 12 9 7 12 15 10 6 9 13 10 1
5 9 12 2 4 7 8 3 3 5 8 4 5 6 8 5 7 8 9 6 8 9 11 7 6 10 13 8 4 6 9 9 5 7 10 10 10 13 18
5 10 1 10 11 14 2 9 12 14 3 7 10 14 4 7 11 12 5 4 5 8 6 8 10 13 7 7 11 14 8 6 9 10 9 10 14 18 10 7 10 13 10 1 1 3 4 2 3 4 5
3 5 6 7 4 4 6 9 5 4 7 10 6 3 5 8 7 6 7 9 8 3 4 7 9 2 3 5 10 7 8 10 10 1 7 10 12 2 6 7 10 3 5 8 9 4 7 8 10 5 9 11 15 6 4 7 10 7 6
8 10 8 4 7 9 9 6 8 11 10 4 8 10 10 1 5 7 8 2 2 3 4 3 6 7 9 4 3 4 6 5 7 9 10 6 6 8 10 7 4 7 10 8 4 5 7 9 7 10 12 10 1 2 4 10 1 5
7 9 2 6 8 11 3 4 7 10 4 5 6 9 5 3 4 5 6 7 8 10 7 4 5 7 8 4 6 9 9 8 10 14 10 5 8 11
5 10 1 11 15 19 2 6 8 12 3 7 11 13 4 5 9 12 5 8 11 15 6 7 10 12 7 6 10 11 8 6 9 13 9 7 11 15 10 9 13 17 10 1 8 10 14 2 7 9
12 3 5 9 12 4 6 8 9 5 7 8 11 6 4 8 10 7 6 7 11 8 9 12 16 9 4 5 8 10 3 4 6 10 1 12 17 21 2 4 7 10 3 5 9 12 4 6 9 13 5 5 7 10 6
6 10 12 7 9 12 14 8 7 11 12 9 8 9 12 10 7 11 14 10 1 7 8 10 2 6 10 13 3 9 13 17 4 8 11 15 5 10 14 18 6 11 17 20 7 7 11 12 8
5 9 12 9 4 7 10 10 6 8 11 10 1 3 5 8 2 2 3 5 3 1 2 4 4 5 7 9 5 6 8 10 6 4 6 9 7 7 8 9 8 5 7 11 9 6 7 10 10 4 5 8
5 10 1 7 11 12 2 8 12 14 3 10 14 16 4 5 8 12 5 6 9 12 6 6 8 13 7 5 7 10 8 8 10 13 9 4 7 9 10 6 9 12 10 1 12 15 20 2 9 13 18
3 7 9 12 4 8 11 15 5 6 10 13 6 6 9 12 7 9 12 17 8 8 11 15 9 7 10 12 10 11 13 17 10 1 4 5 8 2 7 8 11 3 5 6 8 4 6 7 10 5 5 6 9
6 4 7 9 7 5 6 9 8 7 9 11 9 4 7 10 10 6 9 12 10 1 7 11 13 2 6 8 10 3 5 7 10 4 9 10 13 5 10 14 17 6 11 15 19 7 6 9 12 8 5 8 12
9 5 7 10 10 7 10 12 10 1 6 8 9 2 7 10 12 3 5 7 8 4 6 10 11 5 7 8 11 6 4 7 9 7 8 9 12 8 7 10 12 9 11 15 19 10 5 8 12
5 10 1 10 14 15 2 11 17 20 3 8 10 13 4 7 12 14 5 9 11 15 6 7 11 12 7 6 9 12 8 8 11 15 9 7 11 14 10 5 7 9 10 1 9 13 17 2 8 12
16 3 10 13 15 4 8 9 12 5 7 10 12 6 10 12 13 7 7 11 14 8 6 10 13 9 5 9 12 10 9 12 16 10 1 1 2 4 2 4 5 7 3 3 5 8 4 4 7 8 5 5 7
9 6 6 8 11 7 7 9 12 8 5 8 11 9 5 7 10 10 10 14 16 10 1 7 8 10 2 6 7 8 3 5 7 8 4 4 5 6 5 3 4 5 6 7 10 12 7 6 8 11 8 5 7 9 9 8 10
13 10 7 9 12 10 1 8 10 12 2 9 10 15 3 7 10 13 4 9 12 16 5 8 11 15 6 7 12 14 7 6 10 13 8 8 10 13 9 9 11 15 10 5 8 11
5 10 1 6 8 11 2 8 10 14 3 9 11 15 4 8 10 13 5 7 8 10 6 6 10 13 7 7 11 13 8 8 12 16 9 9 13 17 10 6 9 12 10 1 5 6 7 2 3 4 5 3 4
5 6 4 7 8 9 5 8 10 11 6 9 12 14 7 6 8 11 8 5 7 10 9 4 6 9 10 3 5 8 10 1 7 10 13 2 6 8 11 3 5 8 10 4 8 10 13 5 5 9 11 6 6 9 12
7 7 11 13 8 6 8 12 9 7 8 10 10 6 9 13 10 1 11 15 20 2 10 13 15 3 9 10 14 4 8 9 12 5 8 11 14 6 7 10 13 7 7 11 14 8 6 10 13 9
6 8 12 10 7 11 14 10 1 6 8 11 2 5 6 8 3 4 7 10 4 7 10 11 5 4 5 8 6 6 10 11 7 8 11 14 8 5 7 10 9 9 12 16 10 6 7 10
5 10 1 4 5 6 2 1 3 4 3 3 4 7 4 6 7 9 5 5 7 10 6 4 7 9 7 3 5 8 8 3 6 9 9 4 7 10 10 7 10 13 10 1 8 11 14 2 5 7 10 3 6 9 12 4 7 10
13 5 6 8 10 6 7 11 13 7 8 10 14 8 9 12 15 9 5 8 11 10 7 9 11 10 1 10 14 17 2 12 15 19 3 13 18 21 4 10 15 18 5 9 12 17 6 8
11 15 7 7 11 14 8 14 19 24 9 11 15 19 10 10 13 18 10 1 7 10 12 2 5 7 10 3 4 7 9 4 5 6 9 5 4 7 10 6 7 8 11 7 6 9 12 8 5 8 12
9 4 7 10 10 7 8 10 10 1 7 9 12 2 6 7 9 3 5 7 10 4 8 10 13 5 5 8 11 6 4 7 9 7 6 8 11 8 7 11 12 9 8 11 13 10 8 9 11
6 10 1 5 7 10 2 3 5 8 3 5 7 8 4 6 7 9 5 3 5 8 6 2 4 7 7 4 6 9 8 6 8 10 9 7 9 11 10 7 8 10 10 1 10 14 17 2 12 16 19 3 13 17 20
4 9 12 15 5 8 11 15 6 7 11 14 7 6 9 12 8 8 10 14 9 7 10 12 10 7 10 12 10 1 8 11 15 2 6 9 12 3 7 9 11 4 10 14 18 5 6 10 13 6
7 10 12 7 6 9 11 8 5 8 12 9 6 8 11 10 5 9 11 10 1 10 12 15 2 5 7 10 3 6 8 11 4 7 10 12 5 7 11 13 6 8 10 12 7 9 11 14 8 9 12
15 9 6 9 12 10 11 13 19 10 1 7 8 9 2 3 4 5 3 3 5 8 4 2 3 5 5 1 2 4 6 1 3 5 7 3 6 9 8 4 7 10 9 5 8 10 10 5 7 9 10 1 2 4 6 2 4 6 8
3 6 8 10 4 8 10 13 5 7 10 11 6 6 8 10 7 5 8 11 8 4 7 10 9 5 8 11 10 3 5 8
6 10 1 8 9 11 2 7 8 10 3 6 7 9 4 9 10 12 5 3 5 7 6 4 6 9 7 5 7 10 8 6 9 11 9 5 6 8 10 7 9 12 10 1 10 13 17 2 13 17 19 3 9 11
14 4 8 11 14 5 7 11 13 6 6 10 13 7 8 11 15 8 7 10 14 9 7 9 12 10 6 8 10 10 1 7 8 10 2 3 5 7 3 4 7 8 4 5 8 10 5 6 9 11 6 5 7 9
7 7 8 10 8 5 8 11 9 5 6 8 10 5 7 10 10 1 1 2 3 2 3 4 6 3 4 5 7 4 5 6 8 5 4 7 9 6 5 8 10 7 6 8 11 8 5 7 10 9 4 7 8 10 3 8 8 10 1
6 8 10 2 7 9 12 3 5 8 11 4 4 7 10 5 4 6 9 6 3 5 9 7 2 4 6 8 7 8 10 9 5 8 11 10 8 9 12 10 1 8 10 13 2 7 8 10 3 6 8 10 4 7 9 11 5
5 8 12 6 7 11 13 7 5 8 10 8 4 7 10 9 8 11 13 10 7 9 13
6 10 1 7 9 12 2 5 8 11 3 4 7 10 4 8 10 14 5 6 10 13 6 5 9 12 7 7 10 13 8 7 9 12 9 6 7 10 10 6 9 11 10 1 10 15 19 2 9 12 16 3
8 11 15 4 7 10 13 5 6 10 12 6 5 7 10 7 7 9 12 8 8 10 13 9 7 10 13 10 6 8 9 10 1 7 9 12 2 6 7 10 3 5 7 8 4 4 7 9 5 5 8 11 6 7
10 12 7 6 9 12 8 9 12 15 9 10 11 14 10 7 11 14 10 1 6 7 9 2 5 6 8 3 4 5 7 4 5 7 10 5 3 5 8 6 6 9 11 7 7 10 12 8 5 8 11 9 4 7 9

10 4 5 6 10 1 4 5 7 2 3 5 6 3 2 3 4 4 3 5 7 5 1 2 4 6 3 5 6 7 4 7 9 8 5 8 10 9 6 9 12 10 5 7 10 10 1 7 9 13 2 8 10 14 3 7 10 12 4 9 12 15 5 6 9 12 6 5 8 12 7 7 11 13 8 6 8 10 9 7 8 11 10 6 9 11
5 10 1 10 15 20 2 11 15 21 3 13 17 22 4 9 13 17 5 8 11 15 6 10 14 17 7 10 13 15 8 9 12 16 9 7 11 14 10 10 11 15 10 1 6 8 10 2 5 7 9 3 4 7 10 4 7 10 12 5 5 8 11 6 5 8 12 7 6 9 12 8 7 10 12 9 5 8 11 10 7 8 11 10 1 8 10 14 2 9 12 14 3 6 10 13 4 7 10 11 5 6 9 11 6 5 8 10 7 8 12 16 8 7 11 13 9 10 14 17 10 9 11 13 10 1 3 4 5 2 6 8 10 3 5 6 8 4 4 6 9 5 7 10 12 6 8 11 14 7 6 10 12 8 5 7 10 9 3 5 8 10 6 9 12 10 1 8 11 15 2 7 10 12 3 9 12 17 4 6 9 13 5 7 11 12 6 7 11 14 7 10 13 15 8 8 11 15 9 7 9 13 10 4 6 9
5 10 1 5 6 7 2 4 6 7 3 4 5 8 4 3 5 8 5 7 8 11 6 3 4 6 7 5 8 11 8 6 8 9 9 5 7 10 10 4 6 9 10 1 7 10 13 2 6 8 10 3 5 8 9 4 8 11 15 5 6 8 11 6 5 9 11 7 7 10 12 8 5 8 12 9 6 9 10 10 7 11 14 10 1 13 17 22 2 10 15 19 3 8 10 14 4 9 11 16 5 7 11 13 6 8 11 13 7 9 12 17 8 6 10 13 9 8 8 15 10 9 9 17 10 1 8 9 13 2 7 8 11 3 6 7 11 4 5 6 9 5 4 7 10 6 5 8 12 7 6 9 13 8 7 10 11 9 6 9 10 10 7 11 12 10 1 7 11 13 2 8 9 10 3 6 9 12 4 8 12 15 5 9 10 12 6 5 8 12 7 9 12 16 8 8 11 15 9 7 11 13 10 6 10 13

Case 6 fuzzy processing time

15 10 2
5 10 1 1 2 4 2 2 3 5 3 3 4 5 4 3 4 6 5 9 10 14 6 4 5 7 7 2 4 7 8 4 7 9 9 3 5 8 10 2 4 6 10 1 4 5 8 2 3 4 7 3 2 3 5 4 1 3 4 5 4 6 9 6 3 4 8 7 5 6 8 8 4 7 9 9 2 3 5 10 4 6 9 10 1 5 7 9 2 6 7 10 3 4 6 8 4 3 6 8 5 2 4 6 6 5 6 9 7 4 7 10 8 5 6 7 9 6 7 10 10 3 5 8 10 1 3 4 6 2 1 2 4 3 2 4 6 4 4 5 8 5 5 6 9 6 6 7 10 7 7 8 9 8 5 7 10 9 6 8 11 10 4 7 10 10 1 5 8 9 2 6 7 9 3 4 5 7 4 3 5 8 5 2 4 7 6 4 7 9 7 5 8 11 8 4 5 9 9 3 6 9 10 6 7 10
6 10 1 7 8 10 2 4 5 8 3 3 5 7 4 5 7 10 5 4 6 9 6 6 8 10 7 6 9 11 8 5 8 12 9 4 7 10 10 6 8 11 10 1 4 6 9 2 2 4 7 3 3 5 8 4 4 6 9 5 5 8 11 6 6 8 12 7 4 8 10 8 5 9 12 9 5 6 7 10 6 7 8 10 1 3 6 9 2 4 7 10 3 5 8 11 4 6 9 12 5 7 10 13 6 5 7 9 7 4 6 8 8 3 5 7 9 5 9 11 10 6 10 12 10 1 7 10 12 2 5 8 10 3 4 7 9 4 3 6 8 5 4 8 10 6 6 10 12 7 5 6 10 8 3 7 8 9 5 9 10 10 5 6 10 10 1 4 5 7 2 3 4 6 3 2 3 5 4 6 7 9 5 5 6 8 6 7 8 10 7 8 9 11 8 6 8 9 9 5 7 8 10 4 6 7 10 1 6 9 12 2 7 10 13 3 8 11 14 4 6 8 10 5 5 7 9 6 7 9 11 7 4 6 8 8 5 8 11 9 7 8 11 10 5 6 10
5 10 1 3 5 7 2 5 7 9 3 6 8 11 4 7 9 12 5 4 6 9 6 6 10 11 7 7 11 12 8 8 12 13 9 6 7 11 10 5 6 10 10 1 7 9 12 2 6 7 10 3 5 6 9 4 7 8 11 5 8 10 12 6 6 9 11 7 5 8 12 8 4 8 10 9 3 5 9 10 6 8 9 10 1 6 9 10 2 5 7 8 3 4 6 7 4 7 9 10 5 5 8 11 6 7 10 13 7 4 8 10 8 7 11 12 9 8 10 14 10 5 7 10 10 1 8 10 11 2 4 6 7 3 6 8 9 4 5 8 11 5 5 9 11 6 4 6 9 7 6 8 11 8 9 11 16 9 6 9 12 10 7 9 13 10 1 5 6 9 2 4 5 8 3 3 6 8 4 2 3 5 5 1 2 4 6 4 6 7 7 5 8 9 8 6 9 10 9 4 8 9 10 5 8 11
6 10 1 5 8 11 2 6 9 12 3 8 11 14 4 9 12 16 5 6 10 13 6 5 8 12 7 7 11 15 8 4 7 11 9 6 9 11 10 4 6 7 10 1 6 8 9 2 7 9 10 3 8 9 11 4 6 7 10 5 5 8 9 6 4 7 8 7 3 6 9 8 2 4 6 9 5 7 10 10 5 8 11 10 1 3 4 7 2 5 7 10 3 3 6 9 4 4 6 8 5 1 3 5 6 7 9 11 7 6 9 12 8 5 8 12 9 4 7 11 10 8 11 15 10 1 7 10 13 2 8 12 16 3 9 13 17 4 7 11 14 5 6 8 10 6 5 9 10 7 6 7 11 8 7 8 12 9 8 9 13 10 6 9 13 10 1 2 3 5 2 3 5 7 3 5 7 8 4 7 8 9 5 6 9 12 6 7 10 12 7 3 5 8 8 4 6 9 9 5 8 12 10 4 7 10 10 1 6 7 10 2 5 8 9 3 3 6 7 4 4 7 8 5 5 9 11 6 4 5 6 7 8 10 15 8 7 11 13 9 4 6 9 10 5 8 12
5 10 1 10 14 16 2 9 11 14 3 6 10 13 4 7 11 13 5 6 9 13 6 6 10 13 7 7 10 11 8 8 10 14 9 9 11 12 10 11 14 18 10 1 6 9 10 2 7 8 11 3 5 8 9 4 4 7 8 5 7 10 11 6 8 9 12 7 7 8 10 8 6 10 13 9 5 9 11 10 7 10 11 10 1 4 5 7 2 3 4 6 3 5 6 8 4 6 7 9 5 7 8 9 6 4 6 9 7 5 8 11 8 2 3 5 9 6 9 10 10 4 7 10 10 1 1 2 4 2 3 4 6 3 2 3 5 4 4 5 7 5 5 6 8 6 6 7 9 7 4 7 10 8 3 6 9 9 5 7 9 10 1 3 5 10 1 6 8 11 2 5 7 10 3 4 6 9 4 7 9 12 5 8 10 13 6 8 12 14 7 6 10 13 8 5 9 10 9 4 7 9 10 7 10 13
5 10 1 9 12 15 2 7 10 13 3 6 9 12 4 5 8 11 5 7 10 11 6 8 11 14 7 7 11 14 8 6 9 13 9 5 9 12 10 7 11 12 10 1 5 6 8 2 6 7 9 3 4 6 9 4 5 7 10 5 6 9 11 6 5 8 11 7 4 7 10 8 7 11 12 9 5 8 12 10 4 7 11 10 1 8 9 12 2 6 7 10 3 6 9 12 4 5 8 12 5 11 14 18 6 4 8 11 7 7 10 12 8 5 8 12 9 6 9 10 10 7 10 13 10 1 2 3 5 2 3 4 5 3 4 6 7 4 3 6 9 5 5 6 8 6 4 6 9 7 3 5 8 8 6 7 9 9 5 6 9 10 4 7 10 10 1 4 7 9 2 5 8 10 3 6 8 10 4 7 9 11 5 5 9 12 6 5 7 11 7 7 10 13 8 4 6 9 9 5 8 12 10 6 7 10
5 10 1 5 7 9 2 3 5 7 3 4 6 9 4 5 8 11 5 4 7 10 6 3 6 9 7 6 9 11 8 7 10 13 9 5 8 10 10 4 7 11 10 1 9 12 15 2 8 10 13 3 7 10 11 4 6 9 10 5 7 11 14 6 8 11 16 7 5 8 12 8 7 10 13 9 11 14 19 10 6 10 13 10 1 11 14 19 2 5 8 12 3 7 11 13 4 6 8 12 5 7 10 12 6 8 10 13 7 9 12 13 8 7 10 12 9 6 9 12 10 8 11 15 10 1 6 9 12 2 8 11 14 3 7 11 13 4 7 9 11 5 9 13 17 6 7 10 12 7 6 10 13 8 5 9 12 9 7 10 12 10 7 9 11 10 1 7 9 12 2 5 7 9 3 8 10 13 4 9 10 12 5 8 11 15 6 6 10 13 7 7 11 14 8 10 14 17 9 6 7 11 10 5 8 12
5 10 1 3 5 8 2 5 8 10 3 6 8 11 4 4 6 9 5 5 7 8 6 6 9 12 7 5 8 12 8 6 10 13 9 4 7 9 10 7 10 11 10 1 11 15 19 2 10 12 13 3 9 13 17 4 8 11 15 5 7 10 12 6 7 11 14 7 8 10 13 8 11 14 18 9 7 11 14 10 8 12 16 10 1 6 8 10 2 5 7 9 3 4 6 8 4 5 7 10 5 4 7 9 6 6 9 12 7 7 8 10 8 4 5 8 9 6 9 10 10 6 9 10 10 1 7 11 14 2 6 9 12 3 5 9 12 4 7 10 12 5 6 10 13 6 7 11 12 7 8 11 15 8 5 9 12 9 7 10 12 10 6 9 12 10 1 3 5 8 2 7 8 9 3 6 7 9 4 4 5 7 5 5 8 10 6 2 3 5 7 7 10 11 8 4 7 10 9 5 8 11 10 6 9 12
5 10 1 6 9 12 2 5 8 11 3 7 10 13 4 8 11 14 5 6 8 12 6 5 7 9 7 6 9 13 8 7 11 12 9 10 14 16 10 7 11 14 10 1 9 13 17 2 10 14 18 3 6 9 11 4 7 10 12 5 6 8 10 6 7 11 14 7 11 15 19 8 8 10 14 9 7 11 12 10 8 12 16 10 1 8 11 14 2 7 10 12 3 8 12 16 4 7 8 9 5 6 10 13 6 11 14 18 7 8 10 13 8 7 8 10 9 6 10 12 10 7 11 12 10 1 5 8 11 2 6 9 11 3 7 10 12 4 8 11 14 5 9 13 17 6 6 10 12 7 5 9 12 8 7 11 12 9 8 10 13 10 6 10 13 10 1 6 7 9 2 5 6 8 3 4 6 7 4 5 8 10 5 6 9 11 6 5 8 11 7 5 7 10 8 4 7 9 9 7 11 14 10 5 7 10
5 10 1 3 5 8 2 1 3 4 3 3 6 9 4 2 4 7 5 4 7 9 6 5 6 9 7 6 7 8 8 5 8 11 9 6 8 10 10 2 4 5 10 1 7 8 10 2 6 7 9 3 5 6 8 4 4 5 7 5 3 4 6 6 8 9 11 7 6 8 10 8 5 7 9 9 4 6 8 10 3 6 9 10 1 6 9 10 2 7 10 11 3 5 8 9 4 6 8 11 5 5 7 10 6 4 6 9 7 3 5 8 8 7 9 12 9 8 10 13 10 9 11 14 10 1 8 10 14 2 7 9 13 3 6 8 11 4 5 7 10 5 5 8 10 6 6 8 12 7 7 9 13 8 5 8 12 9 7 10 13 10 8 11 15 10 1 4 7 9 2 3 6 8 3 5 8 10 4 6 9 11 5 7 10 12 6 5 8 12 7 4 7 10 8 6 9 12 9 8 11 14 10 6 8 12
6 10 1 5 8 11 2 4 7 10 3 6 9 12 4 7 10 13 5 6 10 13 6 5 9 12 7 7 11 14 8 4 8 11 9 5 6 10 10 6 7 11 10 1 3 4 5 2 5 6 7 3 4 5 6 4 3 5 8 5 1 3 4 6 4 6 7 7 5 8 9 8 6 7 9 9 5 7 9 10 4 6 9 10 1 7 9 11 2 6 8 10 3 5 8 11 4 6 7 9 5 7 10 14 6 8 9 11 7 6 8 11 8 5 8 12 9 6 9 13 10 5 8 11 10 1 10 13 16 2 11 14 18 3 13 18 21 4 7 11 14 5 6 9 13 6 8 12 15 7 9 13 17 8 8 10 13 9 9 10 11 10 8

13 17 10 1 7 9 12 2 6 8 11 3 5 7 10 4 7 9 12 5 6 8 11 6 8 9 10 7 7 11 14 8 6 10 12 9 5 8 11 10 7 8 11 10 1 8 10 12 2 7 9 11 3
6 8 10 4 5 7 9 5 6 9 12 6 7 10 13 7 8 12 15 8 7 8 9 9 8 12 16 10 6 9 13
6 10 1 6 7 9 2 5 7 8 3 5 8 10 4 4 6 9 5 5 8 12 6 7 9 11 7 5 9 12 8 6 10 13 9 4 7 10 10 7 11 13 10 1 5 6 8 2 6 8 10 3 7 8 11 4 6
9 11 5 4 7 10 6 5 8 12 7 7 10 12 8 6 9 12 9 6 8 10 10 7 9 11 10 1 9 11 14 2 7 10 12 3 6 9 13 4 5 8 12 5 7 8 9 6 8 10 13 7 5 9
11 8 4 7 11 9 6 9 13 10 5 9 12 10 1 3 4 7 2 2 3 5 3 6 7 9 4 5 7 10 5 4 6 9 6 3 5 8 7 6 8 11 8 5 8 9 9 7 10 12 10 8 11 14 10 1 7
8 10 2 5 6 8 3 5 7 10 4 6 8 11 5 4 7 10 6 8 11 15 7 6 8 11 8 5 7 8 9 6 9 12 10 4 6 9 10 1 4 6 9 2 7 8 10 3 6 7 11 4 5 7 9 5 4 7
10 6 6 9 11 7 7 11 12 8 5 6 9 9 7 8 11 10 8 10 13
6 10 1 7 9 11 2 6 8 10 3 8 10 12 4 9 11 13 5 5 7 11 6 8 11 15 7 7 11 12 8 9 10 11 9 6 9 12 10 11 14 18 10 1 12 17 21 2 11 14
16 3 10 14 17 4 8 11 15 5 9 13 17 6 7 11 12 7 9 11 14 8 8 12 16 9 7 10 12 10 8 11 15 10 1 8 11 15 2 7 10 14 3 6 9 13 4 7 11
13 5 6 9 12 6 5 8 11 7 9 13 17 8 8 10 14 9 7 11 15 10 7 9 13 10 1 7 8 10 2 6 7 9 3 5 7 8 4 5 8 10 5 4 5 9 6 6 8 12 7 5 7 10 8
3 5 8 9 4 6 9 10 6 7 10 10 1 1 2 3 2 2 3 4 3 4 5 6 4 5 6 8 5 4 7 9 6 6 8 10 7 3 6 9 8 2 4 6 9 5 7 10 10 4 7 8 10 1 6 9 12 2 7 10
13 3 8 11 14 4 9 13 17 5 6 8 12 6 5 9 12 7 7 8 9 8 6 7 10 9 7 11 12 10 5 6 8
5 10 1 9 12 15 2 4 5 8 3 5 8 11 4 6 8 12 5 7 10 13 6 4 7 9 7 5 8 12 8 7 10 11 9 7 11 14 10 8 12 16 10 1 8 10 13 2 6 8 11 3 5
7 10 4 4 6 9 5 6 9 12 6 7 8 10 7 7 11 14 8 6 7 9 9 7 11 12 10 6 9 13 10 1 5 8 10 2 4 7 9 3 3 5 8 4 6 8 11 5 8 11 15 6 6 9 13 7
5 7 10 8 4 6 9 9 7 8 11 10 8 12 16 10 1 7 8 9 2 9 10 11 3 6 8 10 4 7 11 13 5 8 9 10 6 6 7 9 7 5 8 11 8 5 7 10 9 6 9 12 10 7 11
12 10 1 5 7 11 2 4 6 9 3 8 10 14 4 7 8 10 5 6 9 11 6 5 8 12 7 4 7 10 8 9 11 15 9 5 8 11 10 4 7 9
5 10 1 2 3 5 2 3 4 6 3 4 7 9 4 5 7 10 5 3 6 9 6 7 8 10 7 6 8 12 8 5 9 11 9 6 10 12 10 5 8 10 10 1 7 9 12 2 8 11 14 3 6 8 10 4 9
12 15 5 10 14 17 6 7 11 14 7 6 9 13 8 5 8 12 9 8 11 15 10 6 7 10 10 1 5 8 11 2 6 8 9 3 5 6 8 4 7 8 10 5 8 9 11 6 6 7 10 7 7 11
12 8 8 10 13 9 6 8 11 10 7 9 12 10 1 4 5 7 2 7 8 10 3 6 7 9 4 3 4 6 5 8 9 11 6 5 6 8 7 7 9 11 8 5 8 11 9 6 9 13 10 4 8 11 10 1
6 9 12 2 5 8 11 3 7 10 11 4 8 10 13 5 4 6 9 6 5 7 10 7 7 10 13 8 6 8 12 9 5 7 9 10 7 8 11

B. 8 real instances with most probable processing time

Instance 1:

5,4,2,
4,1,2,6,4,3,9,1,2,4,10,2,8,2,3,6,1,3,3,4,4,3,10,1,10,
5,3,3,1,4,2,2,7,4,4,2,1,7,3,5,2,5,2,4,3,1,8,1,1,9,1,4,6,
4,1,3,5,4,3,7,4,7,1,8,2,8,2,2,4,1,9,4,4,7,1,5,2,1,3,7,
5,1,1,2,4,4,1,1,9,2,7,3,5,3,4,6,2,4,3,9,3,3,5,1,9,4,2,3,3,6,2,3,1,7,
5,3,1,3,2,2,4,5,4,4,10,3,9,1,2,2,7,4,1,4,2,3,4,9,3,4,4,2,10,3,2,1,7,4,9,1,2,3,

Instance2:

8,8,2,
8,3,8,2,6,2,1,9,6,5,6,2,3,6,6,4,8,7,5,3,1,5,1,5,4,8,6,1,2,5,5,8,8,7,8,3,2,6,3,8,6,4,10,5,3,1,4,2,9,2,7,3,5,7,6,6,2,3,4,2,2,5,4,4,5,
7,5,7,1,10,6,2,8,2,5,4,2,5,4,10,7,2,8,5,3,7,2,6,7,2,1,1,9,3,9,4,6,8,1,
8,2,3,8,6,7,4,3,6,6,10,7,3,5,2,2,2,1,7,10,4,1,1,8,6,6,5,5,6,7,8,1,4,8,7,1,5,4,6,4,3,2,2,8,2,5,1,8,8,1,7,8,3,6,10,4,10,1,5,
8,3,6,7,7,10,8,8,3,3,9,2,7,8,9,2,4,5,6,2,2,4,1,3,3,2,3,5,4,9,5,5,9,2,8,6,3,3,8,4,8,3,1,7,3,3,8,9,7,1,7,8,9,3,7,5,2,6,10,2,10,4,3,
8,8,6,9,8,7,3,8,4,7,7,5,1,3,2,4,5,4,6,8,4,3,9,1,6,5,2,4,9,7,6,8,6,5,8,5,2,8,5,9,7,3,3,9,1,7,4,6,2,6,10,8,6,4,6,4,7,9,4,5,5,6,3,1,9,
5,8,7,4,8,1,9,5,3,2,5,3,8,6,6,4,4,7,10,8,4,7,6,1,7,6,3,4,2,1,8,2,4,10,1,4,
8,5,6,5,7,10,4,6,5,2,8,10,7,1,3,3,7,2,9,8,8,4,4,6,5,7,8,2,2,2,3,6,1,5,10,6,6,4,7,4,2,4,3,7,8,5,5,6,4,7,10,8,7,1,5,5,10,2,8,3,3,4,
5,5,4,2,4,3,10,4,3,8,9,
8,3,1,2,7,5,4,3,8,1,2,3,2,5,8,4,2,8,9,6,6,7,2,2,7,5,6,10,2,2,4,2,8,4,1,1,3,6,8,8,2,3,6,5,8,7,5,4,6,6,1,9,4,7,1,4,2,8,8,4,1,10,6,3,
7,8,4,3,3,8,2,6,5,5,4,8,6,7,7,2,10,6,9,
8,8,5,4,1,5,6,3,3,7,4,2,7,3,2,9,8,6,3,1,6,7,6,6,7,8,8,1,3,4,7,8,4,5,2,8,5,4,6,9,1,8,7,7,1,6,10,4,3,2,6,5,1,8,9,1,7,2,5,2,8,2,1,7,5,
2,1,5,4,7,7,3,8,7,10,2,7,1,9,5,10,4,3,8,9,
8,8,6,7,7,2,1,3,5,8,8,2,2,6,3,2,4,3,3,4,6,2,2,5,7,3,1,10,2,5,4,9,4,6,10,2,6,7,7,5,7,5,8,1,5,2,7,2,1,9,6,7,1,7,3,7,2,7,5,6,3,3,8,9,
4,4,1,4,7,8,4,8,3,7,3,6,1,5,2,

Instance 3:

10,6,2,
8,3,2,6,3,1,5,4,6,5,7,2,7,3,4,4,10,6,9,1,3,2,4,5,5,10,6,4,8,5,2,2,7,3,3,1,7,6,2,2,2,7,5,7,5,4,9,5,10,3,3,2,9,1,3,5,2,4,6,8,4,9,1,9
,5,8,3,1,10,5,9,3,8,
10,3,5,8,3,4,1,7,2,5,2,3,2,6,5,2,2,4,4,10,1,3,3,8,6,10,3,1,2,2,6,3,1,1,5,3,5,2,4,5,7,6,7,3,3,4,6,4,3,7,4,3,1,5,6,2,2,5,6,6,10,3,6,
3,1,4,4,10,1,6,10,
7,3,6,6,5,3,3,9,2,3,9,6,9,1,4,10,2,1,8,6,4,2,3,2,2,2,3,3,3,5,10,1,5,2,4,3,6,10,
8,1,6,3,1,5,3,5,1,5,3,7,2,10,4,10,5,7,2,6,9,2,3,4,2,5,3,10,4,5,6,9,4,6,2,4,9,5,5,3,6,5,3,10,1,1,4,1,6,10,5,7,5,2,7,5,5,6,5,3,1,4,
10,

9,6,3,2,6,9,4,10,2,10,5,4,1,5,4,5,5,1,7,4,5,2,10,3,2,7,5,10,4,3,5,4,1,1,2,2,7,3,2,6,7,5,4,1,2,10,1,8,6,1,3,2,1,5,6,1,2,4,2,2,9,4,
6,5,1,6,4,1,5,1,6,9,2,5,
8,5,4,3,2,7,5,10,3,6,6,3,3,1,6,3,5,5,3,6,6,1,1,3,4,9,3,4,2,10,5,8,2,6,6,5,1,2,5,10,2,1,6,6,8,5,1,4,7,1,4,3,9,2,6,4,6,4,5,6,1,1,4,3
,6,1,3,5,10,3,3,4,2,2,4,6,5,
6,5,5,6,3,7,4,6,1,8,6,8,4,1,2,3,6,2,3,5,10,1,1,3,4,3,8,4,8,5,5,2,7,2,6,8,1,4,6,2,5,1,4,5,4,3,4,4,8,6,6,
8,5,3,10,4,2,1,2,5,9,2,10,2,1,9,5,7,1,1,5,2,3,3,6,10,3,5,5,2,1,6,8,1,6,8,1,3,6,5,3,4,6,1,4,9,2,10,5,8,
9,1,4,5,5,3,6,1,9,6,2,2,10,4,6,1,6,5,6,2,9,1,10,6,9,5,1,4,2,3,2,5,1,6,3,1,2,10,4,8,6,10,3,1,8,4,3,3,5,1,4,10,1,5,4,1,6,2,
8,4,1,2,5,2,2,10,6,5,5,4,6,1,5,3,9,6,3,2,10,2,3,9,5,1,2,3,2,1,7,2,4,9,3,3,4,3,9,5,2,6,2,2,6,1,4,4,6,3,4,1,5,2,8,6,7,5,6,4,4,

Instance 4:

10,10,2,
10,6,8,2,9,9,4,1,3,2,6,4,5,7,2,6,8,3,8,10,2,10,9,8,5,7,7,6,8,1,4,4,6,6,10,2,3,3,1,3,3,4,1,8,2,2,2,4,10,3,1,9,6,2,7,8,7,6,2,2,6,9,
4,8,9,4,9,1,2,7,2,5,8,8,1,8,3,7,9,8,10,9,7,5,5,1,4,7,1,10,1,6,4,2,2,8,6,10,7,7,9,2,8,10,3,9,6,10,10,2,1,3,5,6,2,2,4,2,8,5,7,4,8,1
,6,8,8,3,8,2,9,6,7,10,10,
10,9,10,4,9,6,4,8,2,2,6,2,8,3,1,6,3,1,7,3,2,7,9,4,5,8,6,1,5,10,3,6,9,5,10,7,8,4,7,10,4,3,7,9,1,6,6,8,6,1,8,10,5,4,2,2,5,9,2,3,7,7
,3,3,6,10,8,6,9,4,1,1,4,1,5,9,3,3,8,5,4,7,6,1,4,5,3,3,7,5,2,6,2,4,4,1,5,6,2,3,9,2,10,6,2,4,10,2,3,9,2,5,8,1,3,8,3,3,8,7,4,10,10,
10,10,10,7,4,6,6,5,7,1,2,7,3,4,9,10,8,9,5,1,1,2,2,4,6,5,8,8,1,6,8,6,5,1,2,7,4,1,9,4,10,5,6,1,9,1,10,5,5,8,7,6,5,3,9,9,5,4,4,10,6,
2,3,4,7,5,4,4,8,9,6,2,1,2,2,4,8,10,1,10,6,10,7,2,10,9,9,6,4,10,6,8,7,4,8,3,4,7,7,1,5,2,8,5,1,8,7,4,2,1,6,1,4,1,1,1,9,2,10,8,3,8,4
,5,5,1,6,7,9,8,10,
10,9,8,2,4,3,2,10,6,8,7,1,3,8,10,10,1,2,9,9,8,2,7,1,6,7,9,10,2,4,3,6,3,5,10,9,10,6,3,5,10,8,6,7,7,7,1,3,5,3,3,10,5,9,2,8,6,9,5,1
,7,8,9,10,1,6,4,2,10,7,2,3,3,1,8,7,2,4,1,9,8,6,8,10,2,9,5,4,7,8,10,5,9,6,1,9,6,1,9,4,3,9,8,3,7,5,6,
10,4,5,9,4,1,6,5,1,5,8,1,5,7,5,3,1,10,2,2,6,6,5,4,5,9,8,3,10,10,8,7,5,7,1,6,10,4,6,5,4,7,3,6,2,8,10,2,8,6,2,8,3,7,10,3,2,9,1,4,1
0,1,2,5,6,10,8,9,5,5,9,5,8,7,7,4,2,9,10,6,1,7,6,9,3,3,4,1,3,8,10,3,8,6,7,6,3,4,9,7,1,5,6,4,10,4,8,8,9,1,6,8,10,9,1,6,9,4,4,7,9,2,
4,3,3,10,6,
10,3,1,4,5,10,8,4,2,5,1,4,6,6,2,3,8,6,10,10,1,6,4,8,7,2,10,7,3,1,10,3,7,4,4,6,8,9,3,8,1,5,7,2,3,10,2,1,9,10,1,9,10,10,10,6,2,9,
7,6,9,3,4,10,8,7,6,10,5,3,3,10,1,5,10,5,6,4,8,9,10,1,9,10,6,3,6,7,6,8,5,2,7,6,2,2,2,2,10,7,5,9,7,7,4,4,2,1,5,8,9,
10,5,4,10,1,1,8,1,6,5,3,5,1,10,6,1,8,4,9,4,8,5,6,1,4,10,2,7,5,2,4,6,7,9,4,3,6,6,6,8,3,3,10,10,7,5,2,8,5,7,1,7,1,8,1,1,2,9,4,4,3,6
,7,1,5,5,9,9,10,3,5,5,7,1,1,8,7,10,6,6,10,4,1,3,8,1,6,1,7,5,7,9,5,7,10,1,9,5,10,6,10,8,8,4,4,
10,5,8,8,5,3,10,1,7,3,3,8,8,1,6,8,1,10,2,7,5,6,3,5,2,4,9,9,7,5,5,2,9,7,2,5,4,6,8,2,7,1,6,3,2,10,3,4,6,6,2,5,8,7,1,9,9,9,8,6,10,2,
7,1,5,10,6,2,2,8,1,10,3,3,5,4,9,9,7,8,4,10,3,6,6,8,6,6,7,10,8,5,4,3,2,1,10,8,3,7,5,5,1,9,1,7,8,5,10,5,9,2,1,6,4,5,6,7,7,3,9,5,3,1
,8,4,6,10,10,7,10,3,6,9,3,2,3,6,2,
10,10,4,8,10,1,7,10,2,9,6,2,5,10,8,8,1,5,9,5,3,6,5,7,4,1,5,8,1,2,2,6,5,3,7,2,4,8,8,3,6,2,2,7,3,1,6,9,2,3,3,5,5,6,8,9,4,6,6,10,9,1
0,2,5,10,1,10,5,10,7,9,9,10,1,4,6,2,10,9,8,6,3,9,4,2,2,5,5,1,5,2,5,3,5,4,9,9,3,3,8,3,9,5,2,5,1,6,10,1,3,9,
10,7,6,4,9,6,10,1,4,2,8,3,5,10,7,6,9,2,8,8,6,7,9,3,3,10,6,9,6,4,9,6,5,5,7,6,4,4,3,7,8,8,7,2,10,7,6,10,9,4,10,3,8,5,4,7,9,6,2,8,3,
10,4,1,1,9,9,3,3,7,1,7,8,10,9,8,5,4,7,2,6,9,10,6,4,1,9,5,9,3,2,7,6,5,6,5,9,4,10,4,2,10,1,1,5,2,6,9,9,8,5,7,2,5,9,4,5,8,9,1,7,8,3,
6,2,1,7,9,1,

Instance 5:

15,8,2,
10,6,3,8,8,10,6,6,4,10,2,6,7,10,6,8,3,7,6,5,2,1,4,6,4,3,8,7,5,3,1,7,3,10,8,8,4,7,7,1,6,1,7,5,10,2,1,3,8,1,7,7,6,8,3,6,1,3,2,2,3,9
,4,3,2,2,8,1,1,6,8,10,2,1,6,8,5,6,4,3,3,5,1,8,7,4,2,1,4,5,5,9,7,9,1,3,6,
14,8,5,4,2,4,3,1,1,7,4,4,6,9,7,7,8,5,6,7,4,6,3,8,8,5,7,4,6,3,10,5,6,3,1,10,2,2,5,5,8,9,7,7,6,4,6,1,8,6,6,8,5,3,5,2,1,1,8,7,7,4,10,
6,4,5,7,2,9,8,3,3,10,1,3,6,2,3,7,10,6,8,1,8,4,9,5,4,2,7,6,3,3,6,5,8,8,4,2,1,7,4,6,5,4,4,7,5,7,3,5,1,10,6,6,7,8,8,8,4,9,4,2,6,4,3,6
,5,5,3,3,2,5,8,5,4,1,4,8,3,4,5,6,9,5,4,7,6,9,1,4,4,8,2,8,3,5,5,8,8,4,
13,1,3,2,1,8,8,6,3,9,4,8,2,10,8,8,7,2,5,8,7,4,5,1,8,6,10,3,7,5,7,2,4,7,3,6,4,4,2,5,8,2,6,4,3,9,1,1,5,7,2,3,2,5,2,2,3,8,3,1,6,4,3,7
,3,1,1,5,5,3,8,10,2,10,7,2,1,2,9,6,5,6,2,5,6,7,3,5,4,5,8,4,8,5,5,2,8,4,7,8,1,1,5,3,6,6,6,7,7,5,5,1,6,5,1,7,7,3,4,1,
12,8,4,1,8,3,1,7,7,10,6,2,3,10,2,3,5,6,3,2,10,7,9,6,4,8,6,1,5,8,2,3,1,7,8,10,7,3,3,8,4,5,1,6,2,4,6,6,5,3,1,6,2,3,5,4,1,1,6,6,7,7,
7,3,4,1,5,9,8,7,10,1,5,2,8,5,4,4,4,6,10,3,3,8,9,6,4,6,3,3,6,10,2,2,7,6,8,4,8,1,5,6,4,5,8,8,9,2,3,3,3,7,3,4,4,6,8,10,5,5,6,2,4,4,7,
6,3,4,1,1,7,
12,4,7,2,3,8,8,1,1,9,4,4,4,8,9,6,3,3,4,3,5,1,2,4,4,7,5,3,2,8,3,2,7,4,1,6,8,2,6,1,2,2,2,6,2,2,7,6,6,3,8,10,1,4,7,10,2,6,5,6,5,8,6,1
,6,3,10,6,10,7,9,7,4,7,6,8,1,5,5,2,8,2,7,9,2,6,4,8,3,7,3,4,9,1,10,3,3,7,4,10,7,5,6,6,5,3,8,4,7,8,9,1,3,5,2,
15,7,5,1,7,7,3,10,2,10,6,2,4,9,1,1,3,4,3,7,3,3,5,6,8,4,3,7,7,8,2,1,4,5,6,6,5,5,6,6,6,1,2,7,8,2,3,6,2,2,1,6,8,7,4,8,6,4,7,1,7,8,7,7
,8,5,5,2,4,6,4,3,3,4,1,4,3,10,5,2,4,3,7,5,1,6,5,8,4,7,8,7,3,1,7,8,1,9,2,6,5,10,6,5,6,8,5,1,5,3,2,5,10,6,8,4,8,6,7,9,1,8,6,1,4,4,2,
5,3,10,7,8,9,6,1,1,10,5,6,3,8,2,4,7,4,6,3,2,2,1,7,9,5,3,1,1,8,7,1,6,8,4,2,7,3,2,8,7,1,9,
15,1,7,1,5,3,9,6,7,5,1,2,1,7,8,3,6,7,4,2,3,7,6,5,6,8,6,3,9,6,8,2,4,7,3,5,7,5,2,7,6,9,5,5,8,9,3,6,10,5,4,7,7,2,1,10,6,6,8,2,2,7,7,6
,2,4,3,1,8,5,4,8,7,3,2,5,2,5,3,9,1,6,6,7,5,8,2,3,6,1,9,4,4,10,5,7,6,8,8,6,5,5,9,6,9,3,4,8,6,7,8,7,7,10,3,9,6,3,8,4,2,9,1,5,4,6,6,1,
7,7,7,5,7,3,3,6,8,8,7,2,4,4,7,3,
8,2,4,2,6,3,4,2,2,4,8,8,4,5,7,1,7,8,7,7,6,5,6,6,8,1,9,4,5,2,8,8,7,7,1,10,5,1,3,8,6,10,4,6,2,6,8,6,4,6,2,5,7,1,10,8,5,5,2,3,4,9,6,3
,1,10,8,3,5,4,1,5,6,6,1,8,3,3,10,

9,2,4,1,2,5,3,3,1,2,10,7,6,1,1,10,7,4,1,8,10,2,7,1,4,7,4,5,3,3,10,7,4,3,1,8,2,5,5,1,8,6,3,2,7,5,8,3,3,5,6,1,8,8,1,2,4,6,4,4,1,7,3,1,4,2,1,8,3,5,6,6,1,2,3,10,5,3,8,6,
12,4,5,9,8,8,4,4,3,1,2,1,10,5,4,4,6,2,3,8,8,8,7,10,2,6,2,3,10,3,2,4,5,1,7,9,1,5,10,7,4,4,3,9,2,5,5,10,6,9,7,3,1,1,3,7,10,3,2,5,4,4,2,2,8,5,5,3,4,10,4,1,8,8,10,2,9,3,4,4,6,5,7,10,8,4,1,9,4,7,10,1,1,3,3,5,4,
9,5,4,9,1,9,6,5,5,5,8,7,3,5,10,7,6,6,2,7,3,2,5,5,4,4,7,6,1,1,8,9,2,3,6,3,5,8,5,6,1,7,6,1,6,2,4,8,8,3,1,2,3,5,4,4,5,1,6,3,2,5,7,1,7,4,8,2,1,5,1,3,1,6,3,7,3,8,7,1,8,8,2,1,5,7,4,6,7,9,6,1,5,4,1,1,8,7,3,3,
8,6,6,1,3,10,2,9,1,3,7,7,5,4,5,3,3,4,5,2,8,1,10,8,3,8,7,10,3,1,4,10,1,9,6,8,5,2,8,1,2,4,1,6,6,4,1,4,2,9,8,6,6,6,8,5,8,4,2,2,7,1,8,3,4,6,5,8,10,7,1,8,7,8,2,9,1,9,5,6,4,5,8,6,3,10,6,4,3,8,10,7,3,4,1,
14,8,2,1,8,2,7,7,1,6,4,1,3,1,6,1,5,3,2,8,7,5,9,4,4,9,7,1,8,2,1,7,2,4,7,1,5,6,6,2,2,5,3,8,4,4,7,6,8,3,1,8,2,1,1,4,4,7,5,1,5,8,8,5,10,7,3,3,2,3,1,3,7,10,4,7,5,2,8,2,5,2,3,5,6,6,5,1,2,8,5,4,2,4,3,8,6,7,5,10,8,2,1,7,1,1,8,5,2,4,7,6,1,3,3,8,1,5,1,8,7,4,4,8,5,7,6,1,8,8,8,7,2,6,6,5,6,3,7,4,1,2,6,1,1,
10,2,7,5,5,3,6,5,6,8,4,7,8,2,5,6,5,3,8,3,4,3,5,10,2,3,4,2,8,8,9,5,1,4,10,8,5,8,8,8,7,4,6,4,2,4,3,5,4,7,1,5,4,3,4,5,6,6,4,1,6,1,6,6,1,3,6,7,5,9,1,2,8,7,7,1,3,3,6,6,2,7,7,1,3,7,2,2,6,6,10,3,6,5,5,4,7,
10,4,4,7,6,1,7,6,5,2,4,5,2,7,3,8,1,4,3,7,1,3,5,2,7,2,8,8,4,6,2,1,6,4,2,6,10,8,4,8,8,8,4,9,5,9,2,6,3,10,7,8,6,1,1,1,3,6,6,2,10,8,6,7,5,4,6,10,7,9,1,8,8,5,3,8,2,2,6,8,8,7,3,2,1,5,9,4,8,3,4,8,1,3,2,3,6,1,7,4,8,10,5,6,3,5,4,9,6,6,10,1,2,3,8,4,3,8,6,5,2,

Instance 6:

15,10,2,
13,6,8,7,1,2,5,6,3,3,2,4,7,10,10,10,10,1,5,8,4,9,10,6,9,5,8,7,8,4,6,2,3,3,10,7,6,3,3,7,10,1,5,6,2,6,1,3,8,8,5,9,9,3,5,8,10,4,4,6,9,3,10,8,9,4,3,7,6,7,10,5,10,8,5,9,3,2,10,4,10,6,8,2,2,8,4,7,9,9,3,2,10,9,3,7,7,4,4,10,1,3,6,10,5,2,2,3,5,9,8,3,3,10,3,2,7,6,4,10,7,5,9,7,8,7,6,7,5,7,1,10,3,8,10,6,4,8,2,8,1,3,3,5,6,2,9,10,1,5,3,8,7,10,
15,6,3,10,7,10,9,5,8,1,10,2,5,10,8,1,1,3,4,8,1,10,4,6,1,9,10,4,3,5,7,8,10,8,7,1,5,3,3,7,6,4,1,5,2,5,9,6,7,5,7,3,1,7,8,9,6,6,4,8,7,4,1,8,8,2,5,10,1,1,7,2,6,5,9,6,4,5,2,8,6,2,1,8,8,5,5,1,7,10,1,7,1,9,6,10,2,3,5,6,10,9,9,1,1,5,4,5,3,5,8,7,1,9,2,9,7,10,8,2,5,2,2,2,10,3,6,6,1,6,3,4,4,2,3,9,2,1,1,8,7,5,7,7,1,3,8,7,2,10,6,3,2,4,10,9,1,8,8,7,2,10,9,5,10,4,3,5,7,4,1,4,5,5,8,7,10,4,10,9,8,5,6,6,3,8,5,1,4,3,9,1,8,2,
10,6,9,6,5,1,1,5,10,2,4,5,8,3,2,6,5,1,7,10,3,8,9,2,4,8,10,7,2,4,1,5,8,1,7,9,6,2,5,4,4,4,1,1,6,9,8,6,4,3,2,5,7,2,1,4,3,10,7,5,2,3,6,1,10,4,2,3,8,9,5,6,2,7,6,10,1,7,9,10,5,10,7,2,4,10,6,9,5,1,8,2,9,
13,2,6,7,9,3,7,2,9,4,10,5,2,7,10,6,8,9,8,1,9,5,9,5,7,3,8,4,4,10,5,10,9,5,7,1,3,9,8,7,2,10,5,3,5,6,3,4,10,8,8,6,6,3,1,2,7,7,9,5,8,10,10,2,5,9,1,4,7,8,7,3,2,7,3,5,9,1,2,2,8,6,5,6,6,6,1,8,2,6,7,1,5,3,9,7,3,5,6,7,10,6,1,5,3,3,5,3,8,5,1,7,9,6,4,8,9,2,5,3,6,9,1,10,1,7,3,1,6,5,4,6,3,2,1,1,6,1,2,3,5,5,8,5,
10,9,2,4,5,2,9,6,7,5,3,10,10,8,6,5,8,3,4,2,7,5,9,2,5,8,9,3,4,1,7,6,9,4,6,7,6,8,10,2,1,10,2,9,9,5,5,4,4,4,7,3,5,4,1,6,7,7,4,9,7,5,5,8,4,5,3,10,8,2,5,3,6,1,1,8,1,9,8,3,2,3,8,9,4,10,8,3,6,1,8,8,9,9,4,2,9,7,8,10,7,4,3,7,5,10,4,8,1,7,3,5,6,7,7,2,10,4,8,5,8,3,8,8,4,9,8,6,4,4,5,10,1,1,6,
12,1,9,10,7,8,6,6,8,4,10,2,7,5,9,7,6,3,9,2,10,6,6,4,6,5,6,7,6,9,5,2,6,10,5,1,10,4,1,7,9,5,8,4,7,9,1,7,7,1,6,4,2,4,4,5,9,5,4,10,7,2,1,9,6,8,2,4,2,6,6,9,8,1,3,8,1,5,4,
13,4,9,3,1,9,4,3,7,10,7,9,4,8,10,7,8,2,2,10,2,6,6,1,2,3,5,9,2,9,7,7,3,10,9,6,6,2,8,5,6,8,9,8,3,10,4,2,10,3,3,7,6,5,8,8,6,1,5,5,3,4,9,3,8,2,9,8,8,7,2,8,5,7,10,7,6,9,3,1,4,1,9,8,5,4,2,10,2,7,1,8,1,6,5,10,6,6,8,5,3,5,7,5,1,5,5,1,2,5,9,8,4,3,10,2,9,2,3,8,4,6,10,7,2,1,7,9,1,10,7,4,9,5,5,7,9,6,5,10,6,2,7,10,8,4,3,9,4,7,
13,2,10,4,9,4,4,4,6,3,9,8,10,1,6,2,3,4,1,5,3,4,6,9,3,8,7,5,1,6,6,6,4,4,10,3,9,5,7,9,5,3,9,4,4,2,5,8,5,7,4,5,8,7,10,4,1,5,4,10,6,2,9,7,2,3,8,9,5,2,3,4,9,3,2,5,7,9,10,1,4,2,7,7,9,9,9,10,7,1,9,5,2,6,4,4,4,1,10,2,1,10,10,7,2,9,10,1,9,6,7,1,6,10,1,8,8,10,
10,4,10,1,8,2,3,3,5,3,5,5,3,8,8,3,8,10,3,9,4,4,4,8,8,5,6,7,9,3,1,8,1,2,2,6,10,3,1,4,2,2,1,4,6,7,2,10,4,9,4,6,8,10,3,3,5,9,7,8,4,5,9,7,2,3,3,10,6,
10,2,2,2,10,8,1,10,6,10,5,5,8,1,4,8,3,10,7,9,9,8,1,10,6,8,2,5,10,3,9,9,6,4,5,6,10,5,4,10,2,8,5,7,10,3,6,2,1,8,7,3,8,5,6,5,9,8,5,7,2,6,4,7,3,4,5,1,10,9,1,7,4,3,3,6,1,8,9,10,1,3,7,6,8,5,5,7,2,10,4,8,10,6,3,6,2,9,4,8,4,6,8,2,2,2,7,7,3,4,1,3,9,1,7,2,5,10,3,8,3,7,10,6,6,4,4,9,3,
13,10,8,7,2,2,1,5,9,4,3,5,5,3,6,5,10,8,4,6,7,5,2,9,8,6,6,9,9,4,6,4,8,6,10,10,1,9,5,3,3,10,7,9,2,4,6,4,8,5,5,3,5,7,9,10,8,6,8,7,7,9,5,6,8,1,1,5,6,10,4,2,2,8,6,10,8,9,4,1,8,7,2,4,6,6,10,2,5,7,7,2,2,2,9,8,6,5,9,5,7,8,8,4,2,9,3,1,4,8,7,4,2,6,2,9,5,4,9,10,3,4,2,5,7,9,3,1,3,10,2,9,7,10,2,4,6,5,5,8,10,8,4,5,8,10,7,4,3,4,9,7,1,1,
15,7,1,4,3,5,6,8,10,6,9,6,8,8,7,6,2,7,5,2,7,4,1,1,6,10,8,4,9,2,3,2,10,10,2,3,5,3,8,1,2,3,6,7,4,3,3,8,10,10,4,7,9,1,5,4,3,10,2,2,2,8,4,6,1,7,9,3,4,6,2,8,10,10,5,10,9,7,3,9,5,3,10,5,1,8,8,10,7,2,10,6,10,1,2,7,7,2,2,7,3,4,10,2,5,1,8,2,1,9,3,9,9,8,10,1,8,2,8,5,10,7,8,7,6,10,2,8,8,1,5,4,3,5,6,6,1,3,9,5,6,7,7,8,2,3,1,6,8,1,
14,1,1,1,3,10,3,8,9,4,6,9,1,3,5,6,2,2,4,7,6,5,3,2,8,7,10,7,7,1,2,2,7,1,7,7,3,6,9,4,5,4,7,1,2,4,6,1,4,3,5,1,6,9,4,4,9,2,9,7,5,5,9,8,1,2,4,5,7,10,3,2,6,6,5,10,7,8,9,2,5,7,7,4,9,1,4,4,8,5,10,6,4,8,5,4,1,3,1,1,7,10,5,9,2,5,5,9,4,7,9,7,2,1,10,5,4,1,10,7,9,10,9,9,2,5,10,4,1,3,8,5,9,3,5,9,2,4,4,3,3,
14,6,1,3,2,8,8,2,9,9,5,8,6,4,7,6,7,5,6,9,4,10,4,7,5,2,5,1,10,6,6,8,3,2,4,10,7,4,9,4,5,1,5,8,9,1,4,9,1,7,8,3,8,10,5,5,2,7,6,9,4,6,9,4,8,7,10,9,3,9,1,1,7,4,8,4,1,9,8,2,3,8,4,10,9,7,2,5,6,3,5,9,2,2,5,6,9,7,7,4,10,9,1,8,8,2,3,7,6,4,7,2,2,10,3,4,1,7,4,1,5,9,9,6,6,3,9,9,2,7,8,9,6,3,3,2,1,1,9,7,3,6,10,5,4,8,4,6,6,3,7,2,3,5,2,10,8,6,1,3,9,7,9,8,1,2,2,5,1,3,8,1,8,4,7,9,9,10,6,7,5,10,3,5,8,8,10,6,7,4,1,10,2,2,5,6,4,1,9,10,6,5,4,2,2,9,1,6,10,10,7,
10,5,5,4,8,4,9,8,7,8,2,4,2,10,9,3,1,2,10,5,1,10,5,8,1,1,7,5,3,7,8,3,8,4,3,9,7,5,9,1,6,10,9,10,10,1,2,6,1,7,8,2,9,8,4,4,7,3,10,5,4,7,7,4,6,3,1,1,4,5,5,5,3,10,10,9,9,5,2,7,9,8,10,6,4,2,7,4,5,3,6,10,3,9,1,6,9,2,8,7,1,10,7,8,2,10,10,8,8,9,10,5,2,10,8,4,4,2,1,7,9,3,4,6,9,

Instance 7:

20,10,2,
17,8,9,2,6,6,3,6,10,4,7,7,4,5,8,1,5,5,1,5,6,10,4,1,10,2,8,8,7,7,6,4,5,10,9,7,3,8,2,1,1,5,6,8,9,7,6,10,3,1,8,6,10,9,6,6,1,6,9,8,3,
3,5,1,6,6,7,10,3,2,4,6,2,9,1,6,9,2,7,6,8,9,4,1,2,7,10,3,1,2,2,9,8,8,9,10,7,2,5,10,1,9,3,9,4,3,10,9,2,1,5,8,8,2,10,10,4,7,5,6,5,1
0,6,3,5,6,8,9,3,9,4,10,9,8,7,10,10,5,2,2,1,7,1,1,7,10,1,10,9,9,2,9,8,5,4,3,3,7,10,2,5,5,7,5,6,4,5,4,7,5,5,8,4,2,9,10,5,2,10,5,3,
2,6,2,9,8,8,4,2,7,3,6,6,1,5,3,10,4,2,6,3,7,
19,7,6,7,9,5,5,8,10,8,1,5,7,10,8,2,1,7,6,2,7,8,4,1,2,2,1,6,9,2,7,10,9,9,7,7,6,3,10,5,2,10,1,8,9,2,10,4,3,8,7,3,9,8,10,9,1,7,5,3,
3,6,4,8,6,1,1,5,3,8,6,7,1,10,9,1,2,9,4,7,10,2,7,4,3,4,9,6,3,8,2,4,1,3,3,9,5,10,2,5,6,2,10,7,1,5,5,6,7,8,8,8,2,1,4,8,3,3,2,2,10,8,
1,6,7,9,2,1,9,9,6,2,1,5,5,8,2,10,3,9,4,2,7,10,5,4,9,6,3,9,10,7,2,4,1,8,6,10,2,1,5,3,2,2,10,3,10,8,2,2,6,4,6,3,5,9,3,8,4,4,5,2,8,6
,1,1,1,8,
13,7,2,8,4,10,3,10,6,9,10,8,8,8,5,4,1,9,2,8,3,9,9,10,8,9,4,1,7,3,6,5,5,7,10,2,3,3,6,5,1,10,7,8,7,5,1,10,8,3,5,2,4,3,10,1,3,2,2,5
,10,8,10,4,7,6,7,1,3,9,5,10,6,2,3,3,3,5,1,7,2,8,6,8,10,8,8,5,4,6,5,2,1,4,2,2,3,2,8,8,7,7,2,5,7,9,2,1,5,3,1,4,6,10,2,2,2,10,3,10,1
,1,5,5,1,3,10,10,6,3,4,8,8,9,1,4,7,2,4,1,3,3,
14,3,6,5,7,7,8,4,9,8,3,4,6,6,10,3,4,7,8,5,1,10,3,9,8,2,9,1,2,3,4,5,6,9,3,2,2,3,4,10,10,4,1,5,3,4,7,10,9,4,6,4,4,7,5,8,2,1,8,8,10,
2,3,9,9,6,3,7,3,4,1,3,9,8,9,5,8,10,10,1,10,3,10,4,1,6,8,7,8,4,10,1,8,7,3,10,9,6,8,5,9,9,8,3,6,7,5,10,1,8,8,9,9,10,6,3,4,4,3,1,10
,2,8,6,2,9,6,5,2,8,10,3,2,4,6,10,7,1,9,7,8,6,10,8,3,5,1,6,6,1,8,5,2,2,5,10,1,3,1,8,1,2,7,9,1,7,10,7,6,3,8,2,2,1,3,5,5,1,7,5,4,5,3,
10,8,3,7,8,3,
17,8,7,10,6,10,10,2,1,9,9,5,5,4,4,2,2,2,9,4,3,5,5,10,7,6,9,7,10,1,4,8,2,9,2,2,1,9,6,1,9,9,7,4,8,3,10,1,4,5,3,9,2,3,5,5,3,8,2,2,4,
10,6,7,9,1,1,10,8,8,7,6,5,9,10,4,6,9,9,7,7,10,5,5,3,4,7,8,5,3,4,1,3,6,10,9,10,10,5,8,7,4,8,1,1,7,6,10,9,3,4,4,2,6,10,4,3,1,4,9,3
,7,6,10,2,2,8,3,4,6,7,1,5,7,9,2,6,8,8,6,4,1,4,7,5,5,9,4,3,3,1,10,6,10,2,9,7,2,8,3,10,8,1,1,9,1,6,7,5,4,3,5,4,3,7,6,1,3,7,7,3,8,7,1
,7,10,9,4,3,2,2,7,4,2,2,3,8,10,3,6,8,10,4,3,5,8,10,4,7,2,2,10,8,3,3,4,8,5,2,9,8,8,1,1,1,6,1,7,9,6,7,4,5,3,8,6,6,7,4,1,9,3,
14,3,7,10,9,10,8,4,7,9,1,2,4,4,10,7,4,1,5,6,2,5,7,9,3,2,10,10,8,7,7,6,4,10,2,4,9,7,1,5,5,7,2,2,3,4,8,6,2,4,3,8,7,4,1,9,9,5,5,2,9,
7,6,2,8,8,4,9,2,6,8,3,4,10,6,4,3,5,8,3,7,3,6,2,9,8,2,8,7,2,10,5,5,2,1,2,2,2,6,8,3,3,2,9,2,1,10,2,7,6,1,6,6,7,5,4,1,9,4,2,7,3,8,5,2
,7,1,10,6,3,9,7,7,10,5,10,4,8,8,6,8,6,3,1,8,8,7,9,10,2,10,5,4,3,7,7,5,
20,4,7,9,6,1,10,2,9,1,4,7,8,10,4,4,7,8,8,7,2,9,9,2,6,4,1,7,8,6,5,9,7,9,8,3,9,7,4,4,2,6,6,2,10,5,6,10,3,9,3,6,8,9,2,10,5,2,7,8,6,9
,4,2,5,6,4,3,2,5,8,7,9,8,3,9,4,5,7,5,3,3,2,5,6,4,9,3,5,6,8,2,10,9,4,5,2,9,8,10,8,7,5,9,10,4,5,1,8,10,2,10,6,2,1,1,9,3,7,8,4,1,5,3,
6,10,2,9,1,2,8,6,7,4,9,4,10,4,3,5,2,10,2,6,6,8,9,4,1,7,9,7,8,5,7,8,2,1,2,8,3,9,3,4,9,9,4,10,8,10,7,1,10,6,6,7,5,1,2,4,5,8,10,3,8,
5,2,2,3,1,5,8,7,6,9,4,3,1,5,2,6,10,2,4,10,8,4,4,1,5,4,8,8,6,9,7,4,8,10,4,1,6,2,10,5,10,7,5,6,5,1,10,10,2,8,7,2,6,10,8,3,
11,1,9,8,1,9,4,5,9,6,1,7,4,5,7,5,10,6,8,2,6,8,5,4,5,6,4,7,10,10,6,1,7,3,8,1,10,6,5,7,8,5,3,4,9,9,10,6,9,10,4,6,9,10,3,9,7,4,6,9,
1,5,8,9,2,9,10,2,5,2,10,8,7,7,5,10,3,9,3,1,10,5,10,4,6,3,4,2,7,6,6,8,8,9,5,7,6,9,3,6,7,3,9,3,1,4,10,10,7,7,5,6,6,5,8,4,6,10,5,1,
10,8,3,3,6,1,1,3,9,2,
12,2,3,6,1,1,8,4,2,7,8,8,5,6,1,1,6,9,4,10,8,3,9,8,1,1,9,9,6,5,7,1,8,5,5,3,4,10,3,5,9,8,5,9,5,5,4,3,1,1,6,10,2,4,9,7,3,2,8,8,2,2,8,
3,9,9,1,2,6,5,5,8,4,6,3,7,7,8,4,10,3,2,2,7,5,9,3,4,4,6,1,10,9,5,6,5,8,8,10,7,7,7,5,3,4,2,3,5,2,9,1,9,9,1,4,9,1,6,9,8,3,7,10,3,8,4,
4,10,3,2,4,3,8,3,5,2,2,2,5,5,4,2,8,10,2,9,8,7,1,4,6,1,4,8,1,10,7,2,5,9,8,4,6,10,2,7,1,5,3,
12,7,2,4,9,6,4,8,7,10,6,6,3,10,5,5,4,3,7,10,8,2,2,6,2,2,4,8,1,6,2,10,2,7,1,4,1,3,4,7,2,9,7,5,5,5,10,9,7,3,7,2,4,8,3,8,8,9,9,5,5,2
,7,9,10,10,4,2,2,2,3,6,10,4,9,9,1,8,5,2,6,7,4,6,9,5,8,3,8,1,1,10,6,4,6,1,9,6,3,10,7,3,7,1,2,2,9,3,8,4,2,5,10,9,3,7,2,3,4,6,9,5,6,
4,7,7,8,6,10,1,1,4,9,10,4,2,7,3,8,
15,6,7,3,3,8,8,2,5,5,10,3,4,10,3,5,4,7,7,1,8,10,8,3,7,9,9,6,10,1,6,10,4,4,5,3,3,5,2,9,1,9,1,2,10,6,5,7,2,2,6,5,10,5,8,10,4,9,7,4
,1,5,5,8,3,3,3,2,3,1,8,10,9,6,6,3,9,3,3,10,7,6,2,4,5,9,6,1,10,8,4,3,9,2,5,6,4,9,6,7,4,2,6,4,5,3,10,10,1,10,7,1,3,5,6,4,8,10,3,8,6
,5,8,6,2,2,8,3,10,6,9,2,3,3,5,2,10,9,1,4,10,10,2,7,1,6,2,5,8,1,6,10,9,4,7,5,4,3,4,8,1,1,5,6,8,10,8,6,10,4,1,6,3,8,7,5,9,3,8,6,
15,7,2,7,1,2,8,8,7,1,10,5,5,2,9,10,5,5,6,10,8,8,3,2,4,4,5,4,7,6,4,1,8,7,2,9,4,6,6,5,1,7,3,1,10,4,6,3,9,4,8,3,3,9,8,3,7,1,6,8,6,4,
6,10,4,6,6,2,10,7,1,2,5,7,9,10,3,5,4,2,2,1,10,5,1,10,2,6,10,2,3,6,8,2,8,10,8,2,8,5,7,7,6,4,10,9,9,3,9,1,2,5,4,7,5,5,3,3,7,3,9,4,
10,3,3,5,6,8,4,9,1,4,8,1,3,10,10,6,9,2,8,7,4,3,7,4,8,9,1,5,6,9,8,6,8,8,4,3,10,5,7,10,8,6,5,6,9,1,10,6,3,3,6,4,8,7,
18,3,1,1,5,7,8,8,2,7,1,8,2,10,10,6,4,10,1,10,6,10,2,9,5,3,7,8,8,9,3,9,9,6,5,6,2,9,4,8,10,1,6,5,3,2,9,8,8,10,4,3,7,7,3,5,6,1,4,3,
8,4,9,4,10,5,7,4,10,8,6,9,5,7,6,6,4,5,9,2,6,3,5,10,10,3,8,1,9,7,3,4,9,5,4,6,8,9,4,2,7,10,5,3,10,1,5,1,4,2,7,6,10,7,10,9,5,3,5,10
,3,5,8,2,7,9,10,7,5,1,3,2,9,4,6,2,2,3,4,2,8,6,1,10,8,4,2,6,1,10,2,7,4,8,5,3,4,9,2,5,10,9,4,2,8,1,9,10,5,8,3,9,7,1,6,7,10,8,2,10,9
,5,2,8,3,10,7,2,10,6,4,3,1,4,2,7,10,9,3,4,9,1,7,10,1,5,4,1,5,4,7,10,1,8,2,9,2,5,5,
11,3,3,5,9,1,1,3,8,2,9,1,7,8,9,4,2,6,8,5,9,10,10,3,8,3,4,1,3,4,8,2,1,5,4,7,1,9,10,7,6,4,2,10,4,5,5,1,7,10,7,3,10,7,9,1,5,2,2,4,1
0,8,6,5,1,1,4,5,8,8,9,1,4,6,9,5,8,7,10,9,9,10,1,3,1,7,8,10,6,9,10,5,1,4,4,5,7,4,5,7,1,8,10,7,5,7,1,6,6,3,9,7,10,2,4,1,7,9,
20,6,9,3,1,4,4,10,8,2,2,2,10,9,5,2,1,9,1,8,7,5,8,10,6,1,2,5,1,1,2,4,4,3,5,3,1,8,2,6,4,6,1,9,1,1,9,4,5,2,8,9,2,4,10,2,9,9,10,1,6,4
,2,10,1,5,1,3,2,6,9,8,10,7,2,10,5,1,1,7,8,5,9,9,4,7,2,9,3,5,7,7,10,6,6,6,6,7,2,10,3,2,3,9,1,6,8,4,1,10,7,7,9,7,8,10,5,9,6,7,10,1
0,3,10,1,7,4,3,2,2,6,8,8,9,3,4,4,2,1,10,8,7,8,10,3,3,9,9,5,6,4,2,2,6,7,1,1,3,8,9,10,5,6,4,1,5,1,6,5,10,10,5,9,9,2,4,3,4,4,10,7,9,
4,3,7,4,10,5,6,6,6,7,9,8,6,8,8,9,2,4,4,2,1,6,7,7,5,3,3,3,9,6,5,6,1,2,8,8,7,10,7,3,5,7,8,1,2,6,5,6,7,10,6,1,3,10,10,4,7,10,10,4,5,
7,4,1,9,6,6,3,5,5,3,
20,5,3,5,8,1,6,1,4,4,10,8,4,9,4,4,1,1,6,3,2,6,8,6,3,1,6,8,4,4,1,6,5,1,1,5,1,5,7,10,6,7,5,4,9,3,10,5,5,9,4,6,8,2,4,4,2,10,2,9,3,8,
9,1,4,8,6,8,2,7,5,8,1,6,10,8,8,9,8,2,10,5,8,8,6,10,8,6,3,1,10,7,7,4,4,9,2,4,8,6,9,8,4,4,10,1,5,6,1,2,3,1,6,5,5,10,8,3,4,8,10,5,4,
6,6,10,6,8,10,4,2,5,1,6,3,4,5,1,4,1,9,4,8,10,7,8,3,10,7,2,7,5,6,6,9,5,8,10,4,3,10,1,6,6,3,3,8,5,5,9,5,3,4,8,8,7,7,2,8,4,6,10,5,9,
10,4,5,2,2,8,7,7,9,4,3,6,8,5,4,9,1,3,9,7,9,10,1,5,2,1,1,4,1,3,2,2,9,8,3,9,8,1,2,1,9,2,8,9,6,4,4,7,10,10,5,6,6,3,7,5,1,1,10,7,10,1
0,5,7,1,6,9,8,4,1,8,2,6,1,3,5,3,9,4,1,6,

15,8,2,5,1,10,9,6,8,10,3,2,5,4,10,7,4,9,1,9,1,2,7,10,4,5,2,7,4,1,2,9,6,5,3,6,10,8,9,4,8,6,7,3,4,10,5,1,1,5,8,8,1,9,3,5,1,2,1,6,2,
7,7,4,5,3,7,7,7,7,9,3,8,6,3,5,5,8,6,6,10,9,2,4,7,2,9,5,2,10,3,10,1,10,10,6,9,2,8,8,8,4,10,6,1,7,1,9,10,5,5,10,9,3,10,6,5,7,10,6,
6,10,7,7,9,8,2,7,4,9,1,10,8,5,8,6,9,7,5,9,9,4,10,9,1,4,3,10,8,10,4,5,2,10,4,7,4,5,4,9,2,6,10,3,1,10,8,8,
18,2,4,2,10,6,10,5,5,9,7,8,5,6,9,3,8,10,10,1,7,2,5,4,3,7,1,2,8,5,4,9,5,2,6,3,1,10,9,9,2,6,6,8,10,4,4,4,8,8,7,8,3,1,2,9,6,9,9,3,5,
1,6,8,2,3,2,4,8,6,10,5,2,8,9,7,3,4,10,3,4,2,1,3,8,8,2,9,7,1,4,3,10,6,8,3,6,9,3,10,9,8,2,6,6,2,1,3,2,10,9,8,5,10,8,7,9,4,6,10,1,9,
6,6,7,5,4,1,4,3,1,4,7,3,1,2,3,7,8,6,4,2,2,4,10,1,10,7,10,9,4,7,2,6,7,5,9,9,6,10,3,7,5,1,10,8,8,9,2,4,6,1,1,9,7,4,9,5,6,9,1,4,10,9
,5,7,4,8,3,8,2,10,8,2,6,4,5,8,4,7,5,1,3,3,6,5,7,9,7,
10,10,5,1,8,8,4,6,3,1,6,2,2,10,7,9,10,3,1,10,9,1,2,3,1,5,6,6,4,6,9,2,2,2,1,1,10,10,5,8,1,10,3,5,9,1,4,3,6,10,1,9,7,7,1,5,4,1,8,4
,8,4,7,10,8,8,7,1,5,6,1,5,9,7,3,2,10,2,10,4,9,6,3,6,2,8,5,1,8,
17,3,5,5,9,4,4,2,10,9,10,2,5,5,6,10,4,4,6,7,9,8,1,6,8,3,5,1,2,9,3,8,5,3,6,6,8,2,4,7,10,1,1,2,7,9,2,4,6,6,6,5,2,8,6,7,6,10,2,2,3,6
,4,6,5,4,8,5,6,9,1,3,10,5,7,3,10,7,2,6,2,2,9,9,5,4,9,1,5,6,4,9,9,9,3,2,7,2,5,9,6,6,5,5,7,9,3,7,4,2,4,8,4,4,8,4,3,2,2,3,5,7,4,1,9,1
0,1,3,4,8,6,5,8,1,5,4,9,9,1,8,7,3,2,9,1,4,10,9,10,9,9,1,3,10,7,7,6,5,8,2,5,6,4,7,2,3,10,9,6,8,10,5,4,4,1,2,2,3,8,7,7,10,8,6,4,1,8
,1,2,5,8,9,4,1,1,3,1,8,8,6,10,4,3,2,2,7,7,1,5,9,

Instance 8:

20,15,2,
20,1,11,4,12,10,9,5,8,8,5,13,1,1,6,3,8,15,2,4,6,7,4,6,10,14,2,11,9,6,7,2,10,1,8,3,3,8,12,3,5,6,12,3,9,2,9,1,6,11,8,15,7,12,8,9
,2,14,1,7,7,13,4,5,4,6,5,3,8,8,12,5,1,2,11,11,1,2,5,14,2,1,4,6,4,8,7,15,5,10,1,9,8,12,9,5,7,4,15,7,3,4,4,9,12,3,14,1,1,7,4,5,2,
12,6,11,2,10,7,3,6,9,5,14,7,8,7,2,3,4,9,6,5,13,3,15,3,10,10,8,2,3,12,3,15,6,7,3,8,3,11,3,5,7,14,2,6,3,4,7,9,9,1,5,13,3,3,1,4,5,
1,14,5,9,7,7,12,4,13,5,2,4,11,8,10,4,3,10,4,7,15,5,2,10,9,6,8,14,13,3,8,1,7,9,1,7,11,7,15,1,4,8,2,1,6,5,14,5,5,4,10,2,3,4,12,1
0,3,5,10,13,7,4,10,6,12,1,11,3,7,8,4,2,14,9,13,7,12,2,7,12,8,6,4,13,5,9,1,11,4,15,8,5,8,8,3,3,4,1,9,14,1,12,12,9,13,2,15,9,14
,8,11,2,3,7,9,4,8,6,6,3,10,6,5,2,4,6,9,8,5,9,10,7,7,5,10,11,2,13,6,1,10,2,5,3,8,2,3,9,8,5,2,8,4,7,1,
17,6,4,7,5,10,15,10,7,6,8,1,2,2,9,11,1,13,9,15,8,3,2,10,10,5,8,9,8,4,5,6,10,2,9,5,7,6,5,9,5,7,4,15,2,1,6,11,9,10,1,6,14,2,11,2
,12,9,15,9,3,8,9,4,13,3,7,1,4,9,14,2,5,10,6,13,5,8,7,5,10,15,4,7,7,14,5,11,1,1,2,12,2,4,8,3,8,6,7,13,11,4,8,8,12,8,3,6,10,9,9,
8,7,10,14,7,4,7,6,4,15,3,5,5,13,6,4,8,7,12,2,15,2,3,5,5,8,5,6,5,3,10,12,8,9,9,14,1,1,8,2,12,5,9,8,11,9,13,4,3,8,5,1,15,4,10,3,
6,3,4,9,7,3,2,6,13,3,10,6,5,4,6,12,7,11,7,7,8,9,1,8,3,15,4,13,1,2,5,14,7,5,1,5,3,3,10,10,8,8,7,4,6,7,13,2,10,15,9,3,1,10,2,5,7,
7,2,14,10,6,6,8,3,4,10,13,1,12,3,9,3,8,13,2,7,6,9,10,6,9,12,3,3,6,1,1,5,9,13,11,2,2,1,7,6,12,6,9,6,13,7,3,2,10,5,14,8,5,6,4,3,
15,7,6,3,15,11,1,5,1,15,5,4,9,14,5,6,2,9,5,10,3,8,4,13,6,1,6,7,4,3,3,2,4,12,3,13,2,9,14,10,12,3,15,7,10,5,11,4,1,6,8,3,13,5,6,
1,5,9,4,3,9,7,
15,6,5,9,14,6,10,1,6,7,11,1,12,4,3,11,8,10,10,13,5,5,12,6,2,9,7,1,10,10,11,3,7,9,3,6,8,14,9,7,6,3,3,1,7,15,7,3,10,10,11,2,4,5
,14,8,1,14,8,15,3,9,7,6,1,11,8,5,3,12,5,7,3,2,8,10,4,4,4,13,4,3,4,10,6,7,8,5,11,1,12,5,10,1,2,1,1,8,13,6,9,4,5,3,15,15,10,10,1
,12,8,9,4,13,4,3,3,5,1,8,6,6,10,2,9,11,2,14,8,4,9,1,5,7,10,5,9,5,3,2,13,3,1,10,8,4,11,9,8,14,6,3,1,12,10,8,8,2,5,6,10,15,9,10,
6,1,9,5,2,14,9,7,4,9,6,4,2,2,7,4,8,5,14,1,13,1,5,8,11,4,3,3,10,7,15,5,1,5,5,3,3,14,1,7,4,5,10,10,3,8,8,9,13,6,12,7,4,8,11,10,9,
6,10,8,15,5,1,8,9,10,2,9,12,6,3,3,6,5,5,5,11,8,9,2,15,9,7,7,10,9,
17,14,15,8,13,4,8,8,7,1,1,7,10,3,9,3,11,6,4,4,12,7,5,5,14,6,6,5,3,7,8,3,10,13,6,5,4,10,4,14,9,6,2,11,7,15,8,2,1,2,7,5,15,9,10,
6,2,3,8,15,8,14,2,2,8,7,4,12,9,4,2,13,10,8,6,11,10,5,4,10,9,1,8,12,7,7,10,10,5,7,12,3,11,5,6,5,9,6,8,3,13,1,15,2,3,9,4,9,5,13,
6,8,5,4,5,15,10,11,10,8,10,5,4,8,15,2,12,5,11,1,6,9,1,2,9,2,2,5,6,8,7,7,7,4,4,2,1,10,13,8,11,9,8,8,9,2,7,1,3,10,3,13,1,7,9,5,8,
2,8,14,2,6,3,10,9,10,10,3,1,10,2,8,7,6,2,15,1,10,9,5,3,4,14,8,5,4,6,8,13,2,5,9,9,15,8,4,10,7,1,11,9,15,12,3,1,7,14,9,7,1,13,4,
2,6,15,10,3,10,9,2,11,1,5,10,10,9,4,5,6,3,8,4,1,2,7,15,1,9,7,3,5,1,12,8,4,7,15,10,6,5,13,3,9,4,11,5,8,6,14,8,3,10,2,6,10,6,
15,3,7,2,14,2,11,4,2,15,1,14,9,6,15,9,14,10,7,10,9,10,10,8,5,5,4,7,6,14,8,13,5,6,3,12,1,6,5,7,8,2,10,10,14,5,9,6,13,5,11,9,1
2,8,7,2,3,2,6,6,5,8,3,4,8,7,2,1,8,11,6,12,10,6,12,6,13,6,14,1,11,2,1,7,8,4,4,1,7,6,6,8,5,7,2,1,10,1,3,15,6,13,4,10,1,7,9,8,7,6,
6,4,10,12,8,3,5,9,12,1,14,1,15,4,2,6,11,6,4,5,1,7,9,9,7,7,7,8,8,5,10,13,3,11,4,2,8,15,4,9,4,11,7,10,14,8,5,4,9,8,15,2,11,3,6,1
0,12,8,3,2,1,2,10,9,4,5,1,14,1,10,5,12,8,7,10,4,14,8,5,4,12,6,13,6,15,4,9,5,5,3,3,15,4,1,5,12,7,7,6,3,15,1,14,10,12,2,
20,1,13,10,3,9,8,6,10,13,2,7,15,5,9,5,7,1,8,3,12,7,2,9,1,9,3,2,9,3,8,7,10,9,5,1,14,4,8,5,10,4,11,4,15,1,7,3,3,6,12,4,8,2,3,10,
3,13,7,9,5,8,3,12,8,7,7,1,4,9,5,8,14,9,4,6,12,7,11,1,10,7,6,9,7,5,3,9,8,7,6,9,8,5,4,2,9,1,8,12,3,15,7,8,9,13,11,4,2,3,13,7,7,1,
10,3,8,5,3,6,1,4,4,6,14,7,12,2,15,7,5,6,1,12,10,14,3,10,4,3,15,5,13,7,6,6,7,1,11,8,8,5,10,7,12,8,5,4,14,5,1,1,2,8,8,12,6,9,3,2
,10,3,10,10,8,13,9,5,7,14,5,10,6,8,10,10,15,1,4,9,11,4,1,5,9,8,14,7,7,10,8,10,15,9,1,14,5,7,7,12,10,13,3,10,5,6,10,8,1,15,8,
2,1,4,8,3,9,5,4,11,9,1,3,14,10,5,5,8,15,6,7,1,12,3,2,7,9,7,8,1,1,8,11,2,13,6,3,6,6,1,14,10,10,9,10,14,10,13,3,11,6,8,10,6,5,5,
6,15,7,7,10,3,7,1,12,6,3,4,5,10,4,13,6,11,4,2,10,9,3,1,12,1,15,3,14,6,9,3,2,8,13,5,6,5,7,3,12,5,9,10,1,13,4,6,1,14,2,15,3,3,3,
2,8,12,7,7,4,9,5,8,1,
17,6,1,8,4,9,13,2,12,2,9,10,5,8,2,13,7,10,8,8,14,6,12,3,13,1,4,8,9,2,10,3,5,7,2,7,2,15,5,14,2,14,15,10,14,3,13,1,2,1,3,9,4,5,
6,2,12,6,7,3,1,5,9,2,8,6,5,9,11,6,5,15,9,3,5,4,5,12,9,10,9,13,3,5,14,9,1,3,12,2,2,5,11,7,13,6,8,4,7,3,9,6,6,10,4,4,5,4,13,9,7,1
4,1,3,2,11,3,6,10,15,5,12,7,10,8,2,6,13,7,5,8,4,1,1,7,3,5,3,10,7,6,1,14,8,3,2,8,6,8,4,3,11,1,10,4,14,5,9,6,1,5,15,1,5,4,7,10,3,
9,13,10,1,15,8,9,3,3,15,7,1,4,9,7,7,6,13,3,8,9,5,6,14,2,11,14,5,15,4,13,8,9,8,7,4,10,9,11,10,6,4,8,10,4,8,12,2,2,5,3,15,5,11,
5,2,10,4,12,3,9,7,13,10,8,6,14,7,4,7,7,8,2,5,1,9,7,1,6,8,10,4,9,15,10,11,1,3,5,12,6,14,6,8,14,8,8,2,9,10,5,8,10,7,4,10,11,1,1
2,7,2,7,1,4,7,1,15,2,13,10,
18,11,1,6,13,5,7,1,12,4,11,4,9,4,15,1,6,3,14,3,8,5,5,2,1,5,1,12,2,1,8,6,15,5,1,2,11,4,10,1,4,4,7,10,12,7,14,4,9,2,13,5,7,11,2,
1,8,10,2,15,4,5,7,7,5,9,5,10,1,1,4,2,14,9,15,1,11,9,3,9,6,5,13,5,8,7,7,6,12,7,7,14,1,11,6,9,3,1,1,15,10,13,5,8,1,12,3,5,1,10,5
,4,6,7,11,8,13,8,2,9,7,4,1,2,6,1,9,8,6,3,1,13,7,1,3,2,1,5,8,4,3,9,13,10,10,5,14,5,12,8,11,7,5,3,9,3,3,6,1,9,15,15,7,14,7,5,8,8,
10,9,6,2,8,3,6,6,6,10,3,4,8,7,7,13,4,12,3,11,6,1,9,1,2,10,11,6,9,8,8,14,5,12,6,9,3,7,10,11,9,3,5,1,7,13,8,10,1,14,10,5,14,6,5,
3,9,2,1,5,11,6,13,2,3,4,6,8,15,4,4,5,8,7,12,2,2,5,15,14,8,2,5,1,1,8,2,5,7,4,2,6,9,7,4,11,2,13,5,15,10,12,8,3,9,9,1,10,3,13,2,2,

15,2,8,10,3,4,5,7,6,9,7,7,13,10,11,9,14,5,12,10,4,7,9,3,1,6,3,14,14,10,15,7,1,3,2,10,10,8,6,1,12,3,9,4,4,8,13,9,11,4,7,1,3,7,
5,3,15,11,2,1,1,10,8,5,1,12,1,14,1,4,3,8,7,7,2,13,4,3,6,2,2,9,4,6,8,15,4,
19,7,1,5,14,9,11,3,4,9,7,5,5,8,9,4,6,11,5,12,3,7,1,14,8,9,4,5,4,6,8,7,5,9,2,9,6,7,12,2,13,10,15,2,9,11,1,4,6,14,3,9,3,8,3,5,6,1
2,2,7,7,13,3,10,9,3,10,6,10,1,6,15,4,13,6,7,7,5,12,8,11,5,3,10,9,2,14,6,4,2,2,1,13,2,5,9,15,2,10,8,2,14,7,12,7,10,6,1,5,2,12,
5,10,5,13,8,2,6,15,3,3,7,14,9,8,8,5,13,10,10,2,1,6,12,6,5,4,7,12,7,14,7,2,7,13,8,15,6,3,10,1,3,10,9,2,2,2,3,4,12,6,15,6,14,10
,5,8,10,5,11,8,13,8,5,1,8,2,9,13,2,12,8,11,10,10,11,6,9,6,6,7,7,4,3,8,14,1,13,8,10,8,12,10,4,3,15,7,2,15,7,1,2,3,6,5,5,8,6,9,6
,2,6,4,3,13,9,10,9,11,8,6,2,14,5,12,9,14,2,9,4,1,7,4,11,6,14,3,13,3,9,4,8,10,15,10,10,1,6,7,5,3,12,1,3,5,15,12,4,1,9,6,2,13,5,
2,1,4,2,10,5,15,8,14,4,11,2,5,1,9,2,3,6,8,1,7,4,1,1,1,10,1,8,15,5,4,6,13,3,11,3,9,1,14,10,5,1,12,8,6,4,2,4,3,10,3,4,7,7,15,7,1
3,6,2,3,
16,10,3,9,14,10,10,9,9,2,4,3,2,4,13,3,8,10,12,3,6,6,6,1,1,4,6,3,7,12,3,8,9,2,3,8,11,10,8,8,5,6,12,8,15,8,10,9,2,10,9,5,12,2,2,
3,3,1,3,12,3,9,6,4,8,5,1,15,2,13,8,8,8,6,3,11,1,2,5,5,13,8,2,1,10,3,6,15,8,8,1,2,5,10,12,2,10,9,4,5,11,9,14,7,13,2,7,7,6,2,3,2,
15,4,2,6,9,7,13,2,3,4,9,8,4,6,2,9,6,3,4,12,7,7,6,14,10,5,7,11,10,10,4,1,7,7,5,7,14,2,12,9,10,1,1,10,9,8,6,2,7,12,7,5,1,1,4,8,2,
15,10,7,1,10,9,8,11,7,14,8,15,10,7,6,3,4,10,8,12,9,1,10,11,7,8,6,10,14,4,11,3,9,2,10,5,15,2,13,10,2,1,8,2,5,4,3,1,6,6,5,4,8,1
5,6,8,14,10,5,5,13,1,8,6,4,1,3,6,12,7,9,6,7,7,1,8,10,7,11,9,15,10,2,1,8,4,2,15,8,10,8,2,4,11,3,1,3,13,7,3,7,3,7,3,9,3,15,3,
20,4,5,9,8,9,6,4,4,3,14,11,4,15,10,9,2,13,5,3,6,10,2,2,6,7,3,8,1,6,3,4,10,5,5,14,4,12,4,1,13,3,10,5,5,14,6,4,3,10,8,1,5,11,5,9
,4,2,2,6,6,8,5,11,11,7,12,10,8,4,13,2,15,10,6,9,9,4,10,3,5,8,14,3,4,7,7,13,7,6,9,15,2,5,10,2,9,10,5,9,7,5,9,5,14,8,5,4,7,2,4,2,
7,2,3,1,7,5,10,12,9,15,8,8,5,9,10,2,7,8,9,8,14,2,8,1,5,5,1,12,10,4,7,10,3,11,8,6,6,13,8,7,6,15,6,9,7,14,10,8,7,3,15,10,1,8,11,
3,13,8,3,1,5,3,5,15,9,12,5,4,8,13,1,2,4,14,6,7,2,9,7,5,9,6,4,2,1,5,5,8,8,11,1,7,3,3,6,8,5,14,8,15,8,5,7,13,7,4,9,4,1,10,8,4,6,8,
1,2,7,7,8,7,14,6,13,6,6,6,1,7,12,7,9,5,1,4,5,15,10,5,14,5,13,7,9,5,6,5,11,10,12,2,1,7,5,9,3,6,15,2,4,10,7,9,8,6,2,8,12,10,10,6
,8,3,5,2,2,11,5,8,7,13,2,12,7,7,9,9,7,5,4,4,10,
19,3,4,5,5,5,12,10,7,6,2,7,8,13,4,5,2,12,1,1,7,11,9,13,5,3,9,10,14,2,8,1,15,7,4,3,10,2,12,9,2,2,3,4,7,4,13,9,6,9,12,5,4,4,4,12
,3,14,1,1,3,2,5,9,8,13,1,6,2,7,4,11,9,3,2,3,5,9,2,8,8,6,15,3,5,6,7,15,4,9,6,1,2,14,8,11,4,13,5,7,5,8,10,5,2,4,10,10,2,12,1,2,2,
6,9,9,14,1,7,7,15,9,11,8,3,7,11,8,1,1,6,13,8,12,1,3,4,15,10,11,1,2,2,6,5,7,9,10,8,14,10,1,7,3,13,6,9,6,5,5,1,1,11,8,14,8,15,1,
4,7,6,1,12,1,3,6,8,1,6,15,9,8,4,2,6,6,8,4,8,10,9,7,7,6,5,8,3,9,14,7,13,2,11,3,6,4,13,6,2,13,3,8,2,5,4,11,3,10,4,1,8,3,8,2,4,15,
2,9,8,12,9,14,5,7,10,8,8,6,1,3,6,6,11,7,2,8,7,10,5,3,4,15,7,4,5,12,7,5,6,12,15,8,11,8,5,8,2,8,6,5,14,9,3,10,8,2,7,8,10,3,4,2,9,
5,8,6,6,14,8,13,9,15,6,11,10,9,8,7,1,10,9,12,9,6,15,4,1,8,12,7,11,10,7,10,14,3,10,4,5,10,6,2,8,8,13,7,5,6,10,10,6,12,9,15,8,
1,9,13,7,9,13,5,4,3,9,1,6,4,10,6,12,9,8,9,15,7,11,2,14,4,3,9,5,4,
17,13,3,5,14,3,5,9,9,9,1,2,7,6,6,6,15,10,13,1,8,7,12,4,4,4,11,3,3,2,6,12,4,13,1,3,1,10,14,6,12,5,5,1,5,15,1,7,9,5,6,10,7,4,9,3
,11,7,14,8,12,1,13,5,7,7,5,15,2,6,4,2,10,10,7,12,9,8,1,4,1,9,3,14,4,3,4,13,4,3,8,6,14,6,6,2,6,2,1,1,6,10,7,12,4,11,4,9,10,12,7
,4,14,3,15,2,12,1,11,8,8,4,9,4,4,10,10,9,1,7,2,10,3,5,13,12,8,9,4,1,1,2,2,8,1,3,1,15,3,6,10,14,10,5,10,10,5,4,3,13,5,15,10,9,
11,7,12,1,14,1,4,7,9,3,7,5,1,10,15,10,13,6,2,10,6,10,3,6,5,9,8,2,6,4,4,9,2,2,8,15,1,5,7,14,6,10,11,9,6,1,9,3,3,6,2,10,10,7,7,9
,1,3,13,3,14,2,13,12,10,8,9,11,3,10,9,7,3,5,6,9,8,6,7,14,1,2,5,15,4,4,1,13,5,10,6,4,14,1,7,2,2,2,5,3,15,3,4,3,13,6,9,9,10,2,11
,14,1,12,9,8,7,10,2,11,9,1,10,4,10,9,4,15,6,6,8,5,10,13,3,10,11,4,5,8,2,9,14,6,8,4,9,1,6,7,7,6,13,3,4,3,10,8,12,9,
19,13,10,4,5,5,13,3,6,10,3,4,1,3,9,6,8,5,12,3,11,5,7,1,4,2,15,3,13,2,6,11,4,14,5,12,7,13,10,10,6,9,2,3,2,7,8,8,5,6,10,4,5,15,
4,8,1,8,15,3,12,2,10,7,11,10,8,3,3,9,6,10,7,9,2,8,6,3,3,1,4,11,6,10,4,7,6,13,5,10,12,8,2,7,4,1,11,8,14,6,15,10,8,10,7,9,6,4,1
0,2,13,5,3,7,11,8,1,14,9,13,6,3,10,11,6,9,7,12,7,15,7,10,10,1,9,6,4,15,6,4,5,7,3,9,7,3,15,4,9,9,14,10,8,6,2,6,12,10,4,3,10,8,
11,3,1,10,13,2,3,5,1,14,4,3,6,3,7,3,11,3,1,9,9,10,9,14,9,13,5,4,3,7,3,9,10,15,4,12,3,11,8,10,9,4,4,2,15,8,2,9,11,3,14,7,5,2,1,
4,10,2,8,6,1,4,10,7,1,2,12,3,13,10,9,5,7,8,15,10,3,5,5,5,5,12,4,1,4,2,7,4,7,7,8,4,10,5,13,5,5,1,1,10,14,7,12,10,15,12,5,3,9,1,
9,14,9,5,6,10,3,6,3,8,3,7,2,4,4,15,4,2,8,11,10,13,8,9,9,2,13,5,10,8,15,12,7,11,10,13,5,1,1,10,7,5,7,4,3,9,2,14,3,7,9,8,10,15,
4,2,6,3,2,6,4,2,14,9,3,1,
17,12,3,6,14,2,4,6,7,1,13,4,15,7,12,9,9,9,6,5,2,3,5,10,10,9,6,3,9,4,2,6,2,5,6,9,3,15,3,9,6,6,4,2,8,7,1,2,11,6,15,10,5,3,7,6,9,8
,10,15,5,5,2,13,8,11,8,3,4,6,5,9,2,8,5,12,1,10,8,15,1,4,7,1,12,6,9,7,13,9,2,1,8,1,5,9,3,7,6,8,10,4,4,4,14,7,11,9,15,3,10,7,1,1
3,3,8,5,5,10,10,10,11,6,9,2,2,10,4,5,12,3,13,8,1,9,2,6,7,15,3,2,4,3,8,7,7,11,3,14,6,12,6,5,9,4,1,13,10,5,10,9,8,6,2,3,11,8,5,7
,7,2,3,11,3,14,3,12,5,3,6,5,2,10,6,6,1,5,13,4,10,4,7,2,2,10,5,9,7,6,1,12,6,1,3,15,1,3,1,8,10,10,5,15,7,6,13,2,12,9,3,10,11,5,1
0,5,8,8,6,9,14,4,15,4,5,7,9,3,4,9,1,5,2,8,3,2,3,6,1,13,3,1,12,1,10,10,6,6,9,13,7,8,3,9,5,15,5,3,8,14,4,2,10,7,6,3,5,2,12,2,14,6
,7,1,10,2,10,6,2,5,9,11,1,7,5,12,4,
19,4,5,5,6,9,8,7,12,8,10,10,2,8,2,4,6,3,8,12,6,5,4,14,4,1,6,6,8,7,5,13,1,9,14,3,15,9,7,4,6,7,2,1,13,8,11,4,10,5,8,5,5,5,12,10,
3,3,9,1,9,5,9,9,10,15,3,10,8,8,7,4,7,12,7,2,3,2,15,4,3,6,10,2,6,10,5,13,5,15,6,1,8,3,10,9,6,5,3,14,5,6,6,11,9,8,2,7,7,9,12,2,1
1,1,8,5,10,6,3,1,15,9,13,3,5,1,8,14,8,5,4,11,4,12,6,1,4,3,2,4,8,6,6,5,6,8,9,9,14,6,2,8,13,5,7,13,10,11,6,7,7,12,10,14,7,3,5,10
,7,6,4,4,13,10,7,1,15,6,11,8,3,3,5,4,3,3,5,10,9,9,2,12,1,4,11,8,5,1,14,10,12,3,12,10,9,14,1,12,9,11,7,5,7,3,4,8,3,13,9,2,7,6,9
,1,5,4,5,14,6,6,3,6,13,4,2,3,11,3,10,1,8,4,15,4,9,5,14,4,5,8,12,1,7,2,1,4,14,15,10,14,6,7,9,9,7,10,10,5,10,12,2,8,3,4,10,11,6,
6,1,13,4,3,8,2,5,9,15,9,7,7,1,8,2,5,13,1,14,7,9,7,12,8,6,9,2,4,5,1,4,3,7,6,12,6,15,7,
19,15,14,10,5,9,15,1,8,3,9,3,10,8,6,3,13,4,7,8,11,1,1,3,12,4,2,7,4,2,3,7,3,15,2,14,8,2,7,13,13,7,6,3,11,6,2,9,15,7,10,1,12,10
,9,6,3,7,7,4,5,4,14,9,4,9,11,15,10,1,5,13,8,3,10,8,3,11,10,9,1,5,5,12,7,7,6,2,7,2,10,5,12,7,1,2,3,15,2,5,12,7,6,5,14,6,9,9,10,
9,3,1,15,4,7,10,13,7,8,10,4,1,1,10,5,9,11,4,8,11,6,5,1,14,1,13,3,10,4,4,5,8,6,6,3,6,15,5,11,7,2,4,3,9,9,3,14,1,9,10,2,15,9,4,5
,2,2,5,10,8,10,14,9,13,5,7,2,7,15,10,7,5,14,8,12,7,2,2,13,3,1,6,14,1,7,6,4,11,2,12,10,7,10,10,2,8,6,15,4,13,5,14,3,5,8,3,5,4,
10,9,2,5,15,8,10,1,6,2,14,4,11,10,1,14,1,5,12,8,15,4,5,5,11,9,3,5,6,11,6,15,4,5,3,6,1,2,9,12,4,4,14,2,13,8,12,8,1,5,5,1,10,11
,8,13,6,12,7,2,10,5,3,2,14,8,6,7,12,10,2,4,
16,11,5,1,14,8,12,8,7,7,1,2,13,9,10,10,8,7,15,6,6,8,3,10,10,12,7,14,9,8,8,15,7,10,5,4,1,9,2,11,7,2,4,7,2,6,3,9,14,2,10,5,4,4,
5,5,13,9,14,1,1,14,9,9,5,12,6,6,2,11,5,15,2,13,7,5,5,7,10,3,2,8,2,4,1,2,2,15,3,10,11,10,9,1,7,8,14,2,10,9,8,1,6,10,12,3,15,3,
1,10,4,2,2,1,13,6,5,4,2,11,8,4,8,12,15,1,6,10,1,2,2,8,13,2,12,2,9,5,11,7,7,6,8,3,14,10,4,7,3,1,8,12,1,6,1,12,3,9,11,10,8,3,2,4
,9,7,5,6,15,10,13,8,4,3,6,2,12,6,10,6,13,6,7,12,8,14,9,9,10,11,8,10,7,13,1,15,9,3,6,7,3,5,10,1,6,4,9,5,1,1,5,1,15,7,11,5,13,8,

5,12,8,14,8,6,5,9,8,10,3,11,15,7,10,3,4,5,7,8,11,8,3,7,14,4,8,8,9,5,6,1,13,1,13,5,3,14,9,13,8,4,5,3,6,15,8,10,2,1,3,8,7,9,9,6,
5,7,1,11,5,6,3,5,8,1,13,9,14,5,9,9,12,5,8,15,7,10,2,14,7,3,2,7,6,4,8,11,3,8,9,
18,14,14,3,4,8,11,9,12,10,9,8,7,10,8,7,1,8,6,2,13,7,10,7,15,10,3,1,5,10,2,2,7,6,3,9,1,8,4,4,2,1,3,9,14,8,13,4,5,8,7,3,12,2,10,
2,5,14,3,13,4,12,4,10,5,1,3,9,1,6,1,11,9,5,3,9,3,8,8,5,14,9,1,3,2,4,10,5,11,3,9,2,7,4,6,15,1,13,10,8,3,1,4,9,5,5,10,8,13,1,12,
1,6,1,2,1,9,7,10,2,5,5,15,5,2,4,8,9,10,12,4,1,5,6,12,1,14,7,9,1,1,10,6,4,8,10,13,3,10,4,3,4,15,7,15,6,2,11,4,15,5,14,7,8,5,5,3
,13,6,9,6,4,1,10,8,12,3,2,2,3,5,1,8,7,10,2,3,9,11,10,5,9,9,4,10,1,8,10,4,2,7,9,5,9,9,6,7,9,12,7,11,9,8,1,15,3,13,1,14,4,3,9,7,6
,7,2,7,12,12,3,3,4,13,5,14,4,9,9,10,3,15,8,4,1,8,7,2,1,5,5,11,8,1,10,4,12,7,9,13,4,9,4,1,7,8,4,5,9,15,2,11,4,12,4,6,2,2,1,3,3,7
,8,1,4,8,6,7,15,6,13,2,1,9,2,2,
17,13,14,9,7,8,4,4,3,3,11,7,10,2,13,3,12,2,1,3,6,6,8,9,2,4,9,3,12,12,1,15,10,8,2,2,6,11,3,10,7,9,6,3,5,14,4,6,3,1,1,4,3,15,2,4
,10,2,7,3,12,5,1,1,14,5,11,5,8,8,3,2,9,10,5,9,15,1,13,4,4,7,6,9,7,3,5,9,9,10,6,13,6,15,9,14,7,1,8,4,6,8,1,8,13,7,4,1,9,3,8,7,3,
5,3,6,2,13,2,10,5,15,5,1,1,4,7,1,5,2,11,11,8,14,10,13,3,7,8,8,7,9,2,12,9,6,1,1,1,15,2,4,4,6,12,6,10,10,15,9,3,2,14,10,13,10,7
,3,2,2,3,10,5,5,3,11,6,8,4,9,2,2,1,6,12,4,11,4,4,2,10,1,10,12,9,11,5,14,1,15,8,9,5,7,1,6,10,5,6,8,2,5,14,2,15,10,11,9,1,9,10,6
,5,10,3,9,9,2,1,13,5,1,10,5,7,4,10,10,1,3,2,8,10,12,6,15,10,1,4,7,8,3,6,8,1,1,14,2,9,8,13,5,5,4,3,2,15,5,12,4,11,4,2,10,7,6,9,
8,8,15,2,14,9,9,10,11,7,10,5,7,9,2,10,4,7,13,4,7,14,9,5,5,6,10,12,3,2,6,9,4,8,8,7,9,15,8,1,7,10,2,11,7,

C. 8 real instances with fuzzy processing time

Instance 1:

5 4 2
4 1 2 5 6 7 4 3 5 9 11 1 1 2 3 4 9 10 14 2 6 8 10 2 3 4 6 8 1 1 3 4 3 4 2 4 6 3 9 10 15 1 8 10 13
5 3 3 1 2 5 4 1 2 3 2 5 7 9 4 4 1 2 3 1 5 7 9 3 3 5 6 2 4 5 7 2 4 1 3 4 1 7 8 11 1 1 7 9 10 1 4 5 6 9
4 1 3 4 5 7 4 3 6 7 10 4 4 7 9 1 6 8 12 2 5 8 10 2 2 3 4 5 1 8 9 12 4 4 4 7 10 1 3 5 7 2 1 2 5 3 6 7 10
5 1 1 1 2 3 4 4 1 4 5 1 6 9 10 2 5 7 9 3 4 5 6 3 4 5 6 9 2 2 4 5 3 7 9 10 3 3 3 5 7 1 7 9 11 4 1 2 3 3 3 5 6 7 2 1 3 4 1 6 7 9
5 3 1 1 3 4 2 1 2 3 4 3 5 6 4 4 8 10 14 3 6 9 10 1 1 2 3 2 6 7 8 4 1 3 4 6 2 1 3 4 4 7 9 13 3 3 4 5 4 2 9 10 11 3 1 2 3 1 5 7 8 4
8 9 12 1 2 1 3 4

Instance 2:

8 8 2
8 3 8 1 2 3 6 1 2 3 1 6 9 11 6 5 5 6 8 2 1 3 4 6 4 6 8 4 6 8 10 7 3 5 7 3 1 3 4 5 1 4 5 6 4 7 8 12 6 1 4 7 2 3 5 7 5 7 8 10 8 7 5
8 12 3 1 2 3 6 1 3 4 8 5 6 7 4 8 10 12 5 1 3 4 1 3 4 6 2 6 9 13 2 7 1 3 4 5 4 7 9 6 6 1 2 3 3 2 4 6 2 1 2 3 5 3 4 5 4 4 5 6 7 3 5
7 7 1 8 10 14 6 1 2 3 8 1 2 3 5 2 4 5 2 4 5 6 4 9 10 15 7 1 2 3 8 5 1 3 4 7 1 2 3 6 4 7 9 2 1 4 5 1 7 9 11 3 7 9 12 4 5 6 7 8 1 3
5
8 2 3 5 8 11 6 4 7 9 4 3 5 6 9 6 9 10 14 7 1 3 4 5 1 2 3 2 2 1 4 7 7 9 10 14 4 1 1 3 5 8 5 6 7 6 3 5 7 5 5 6 9 7 8 1 4 6 4 6 8 9 7
1 3 4 5 3 4 5 6 3 4 5 3 1 2 3 2 5 8 11 2 5 1 2 4 8 6 8 10 1 7 5 8 11 3 6 9 10 15 4 9 10 12 1 4 5 7
8 3 6 5 7 10 7 9 10 15 8 5 8 12 3 3 8 9 12 2 5 7 9 8 7 9 11 2 4 4 5 7 6 1 2 3 2 4 1 4 7 3 1 3 4 2 3 3 5 7 4 5 9 12 5 5 6 9 12 2 5
8 12 6 1 3 4 3 7 8 9 4 7 8 9 3 1 5 7 9 3 1 3 4 8 7 9 10 7 1 6 7 10 8 6 9 11 3 4 7 8 5 1 2 3 6 7 10 15 2 7 10 12 4 1 3 4
8 8 6 6 9 13 8 4 7 9 3 7 8 11 4 5 7 9 7 3 5 7 1 1 3 4 2 3 4 6 5 3 4 6 6 8 2 4 5 3 6 9 11 1 5 6 7 5 1 2 3 4 7 9 13 7 5 6 7 8 6 4 5
7 8 3 5 6 2 7 8 11 5 6 9 13 7 1 3 4 3 5 9 10 1 6 7 9 4 5 6 7 2 6 7 10 14 8 4 6 8 4 6 2 4 5 7 5 9 13 4 4 5 7 5 5 6 9 3 1 7 9 12 5
7 8 11 7 2 4 5 8 1 5 9 11 5 1 3 4 2 3 5 6 3 7 8 11 6 5 6 8 4 3 4 5 7 8 10 14 8 3 4 5 7 6 1 3 6 7 5 6 8 3 2 4 5 2 1 2 4 8 1 2 3 4 8
10 13 1 3 4 6
8 5 6 3 5 6 7 7 10 12 4 5 6 9 5 1 2 3 8 9 10 15 7 1 1 3 4 3 6 7 10 2 6 9 10 8 6 8 10 4 3 4 5 6 4 5 7 7 6 8 12 2 2 1 2 3 3 5 6 7 1
5 9 10 15 6 6 3 4 6 7 3 4 6 2 2 4 6 3 4 7 9 8 4 5 6 5 5 6 9 4 7 7 10 13 8 6 7 10 1 3 5 7 5 8 10 12 2 8 1 3 4 3 2 4 6 5 5 2 4 6 2
2 4 5 3 9 10 11 4 1 3 4 8 6 9 10
8 3 1 1 2 3 7 4 5 7 4 1 3 4 8 1 1 2 3 3 1 2 3 5 7 8 10 4 1 2 3 8 8 9 13 6 5 6 7 7 1 2 3 2 6 7 9 5 6 6 10 11 2 1 2 3 4 1 2 3 8 2 4
5 1 1 3 6 3 6 7 8 11 8 1 2 3 3 4 6 7 5 8 6 7 8 5 3 4 6 6 5 6 9 1 6 9 10 4 6 7 9 1 4 1 2 3 8 8 2 4 5 1 7 10 11 6 1 3 4 7 5 8 9 4 1
3 4 3 7 8 9 2 5 6 8 5 3 5 6 4 8 5 6 7 7 4 7 10 2 8 10 13 6 6 9 13
8 8 5 3 4 5 1 4 5 6 6 1 3 4 3 4 7 10 4 1 2 3 7 1 3 4 2 7 9 11 8 5 6 9 3 1 5 6 9 7 5 6 7 6 5 7 8 8 8 1 2 4 3 2 4 5 7 7 8 11 4 3 5 6
2 6 8 10 5 3 4 6 6 6 9 11 1 5 8 11 7 7 1 3 6 6 8 10 15 4 1 3 4 2 4 6 7 5 1 3 4 8 5 9 12 1 4 7 8 2 5 1 2 3 8 1 2 3 1 7 4 5 7 2 1 4
5 6 4 4 7 10 7 3 5 8 12 7 9 10 15 2 6 7 9 1 5 9 12 5 9 10 11 4 1 3 4 8 6 9 11
8 8 6 5 7 9 7 1 2 3 1 1 3 4 5 5 8 11 8 1 2 3 2 5 6 7 3 1 2 3 4 1 3 4 3 4 5 6 9 2 1 2 3 5 4 7 10 3 1 6 10 13 2 4 5 7 4 8 9 13 4 6 9
10 13 2 5 6 7 7 6 7 10 5 5 7 10 5 8 1 3 4 5 1 2 3 7 1 2 3 1 5 9 12 6 5 7 8 1 7 1 3 4 7 2 6 7 8 5 4 6 8 3 1 3 4 8 6 9 12 4 2 4 6 1
3 4 5 7 6 8 11 4 8 1 3 4 7 1 3 4 6 1 2 5 5 1 2 3

Instance 3:

10 6 2
8 3 2 5 6 7 3 1 2 3 5 3 4 6 6 5 6 7 9 2 4 7 9 3 2 4 6 4 8 10 11 6 5 9 11 1 1 3 4 2 4 4 5 6 5 9 10 11 6 4 7 8 12 5 1 2 3 2 4 7 10
3 1 3 4 1 6 7 10 6 1 2 3 2 2 6 7 8 5 6 7 8 5 4 7 9 10 5 7 10 11 3 1 3 4 2 8 9 11 1 1 3 4 5 2 3 4 6 6 5 8 10 4 6 9 12 1 8 9 12 5 7
8 9 3 1 9 10 11 5 6 9 11 3 7 8 12

10 3 5 7 8 9 3 2 4 5 1 6 7 8 2 5 1 2 3 3 1 2 3 6 5 1 2 3 2 2 4 5 4 9 10 12 1 1 3 4 3 7 8 10 6 9 10 12 3 1 1 2 3 2 5 6 8 3 1 4 6 1
5 1 3 4 5 2 3 4 6 5 6 7 8 6 4 7 9 3 1 3 4 4 5 6 9 4 3 6 7 8 4 1 3 4 1 3 5 7 6 1 2 3 2 5 5 6 9 6 9 10 11 3 6 1 3 4 1 3 4 6 4 9 10 12
1 6 6 10 14
7 3 6 4 6 9 5 1 3 4 3 5 9 10 2 3 6 9 11 6 5 9 12 1 4 9 10 15 2 1 7 8 12 6 3 4 6 2 3 1 2 3 2 1 2 3 3 3 1 3 4 5 7 10 11 1 4 5 6 2 4
1 3 4 6 9 10 12
8 1 6 1 3 4 1 5 1 3 4 5 1 3 5 6 3 6 7 8 2 9 10 11 4 9 10 13 5 4 7 8 2 6 6 9 10 2 1 3 4 4 2 3 5 6 3 8 10 14 4 3 5 6 6 7 9 10 4 6 1
2 3 4 6 9 11 5 4 5 7 3 4 6 9 5 3 7 10 15 1 1 2 4 4 1 3 4 6 9 10 14 5 4 7 9 5 2 4 7 9 5 4 5 7 6 3 5 7 3 1 2 5 4 9 10 13
9 6 3 1 2 3 6 7 9 12 4 9 10 11 2 8 10 15 5 3 4 5 1 3 5 7 4 5 4 5 7 1 5 7 8 4 3 5 6 2 9 10 13 3 2 4 7 9 5 8 10 14 4 1 3 4 5 4 1 4
6 1 1 2 3 2 6 7 9 3 1 2 3 6 5 7 10 5 4 1 3 5 2 8 10 11 1 7 8 9 6 1 2 4 3 1 2 3 1 5 5 6 7 1 2 2 4 6 2 2 8 9 11 4 5 6 9 5 1 5 6 8 4
1 4 6 5 1 4 5 6 7 9 12 2 4 5 6
8 5 4 1 3 4 2 5 7 9 5 9 10 13 3 4 6 8 6 1 3 4 3 1 4 6 8 3 4 5 7 5 1 3 4 6 6 1 4 5 1 1 3 4 4 6 9 10 3 2 4 5 2 9 10 15 5 6 8 11 2 6
4 6 9 5 1 3 6 2 5 9 10 14 2 1 3 4 6 6 6 8 12 5 1 4 5 4 5 7 8 1 2 4 6 3 8 9 13 2 5 6 8 4 6 2 4 6 5 4 6 8 1 1 4 5 4 1 3 4 6 1 1 3 4
5 6 10 11 3 1 3 4 4 1 2 3 2 2 4 6 6 3 5 7
6 5 5 5 6 7 3 4 7 8 4 4 6 7 1 7 8 10 6 6 8 9 4 1 1 2 3 3 4 6 7 2 1 3 4 5 7 10 13 1 1 1 3 4 4 3 7 8 10 4 5 8 12 5 4 5 7 2 5 7 8 2 6
6 8 9 1 2 4 6 6 2 4 5 6 1 3 4 5 5 2 4 5 3 2 4 5 4 7 8 9 6 5 6 9
8 5 3 8 10 15 4 1 2 3 1 1 2 3 5 8 9 11 2 8 10 13 2 1 6 9 10 5 6 7 9 1 1 3 5 7 2 3 1 3 4 6 7 10 11 3 5 4 5 6 2 1 2 4 6 6 8 11 1 6
7 8 9 1 3 5 6 8 5 3 2 4 5 6 1 4 5 4 6 9 13 2 8 10 13 5 6 8 12
9 1 4 4 5 6 5 3 5 6 8 1 6 9 11 6 1 2 3 2 9 10 12 4 5 6 8 1 6 3 5 7 6 2 5 9 13 1 7 10 13 6 8 9 13 5 1 3 4 4 1 2 3 3 1 2 3 5 1 5 6
8 3 1 2 5 2 6 10 12 4 6 8 10 6 9 10 14 3 1 5 8 11 4 1 3 4 3 3 5 7 1 4 7 10 13 1 5 2 4 6 1 6 1 2 3
8 4 1 1 2 3 5 1 2 3 2 8 10 14 6 3 5 6 5 4 4 6 9 1 4 5 6 3 5 9 10 6 1 3 4 2 9 10 15 2 3 5 9 11 5 1 3 4 2 3 1 2 3 1 5 7 10 2 4 6 9
11 3 1 3 4 4 3 5 9 12 5 1 2 3 6 1 2 3 2 5 6 7 1 4 2 4 6 6 3 3 4 6 1 4 5 6 2 7 8 9 6 5 7 9 5 4 6 9 4 3 4 6

Instance 4:

10 10 2
10 6 8 1 2 3 9 5 9 11 4 1 4 7 3 1 2 3 6 3 4 6 5 4 7 9 2 6 6 8 10 3 5 8 10 10 2 9 10 15 9 7 8 9 5 4 7 10 7 5 6 9 8 1 3 5 4 3 4 6 6
5 6 8 10 1 2 3 3 1 3 4 1 1 3 4 3 4 1 4 5 8 1 2 3 2 1 2 3 4 10 1 3 4 1 8 9 11 6 1 2 3 7 6 8 12 7 6 1 2 3 2 5 6 8 9 3 4 5 8 8 9 10 4
7 9 11 1 1 2 3 7 1 2 3 5 8 7 8 9 1 5 8 9 3 6 7 8 9 5 8 11 10 6 9 10 7 5 3 5 6 1 2 4 5 7 1 2 4 10 1 4 5 6 2 4 6 2 1 2 3 8 5 6 7 10
7 5 7 9 9 1 2 3 8 8 10 15 3 6 9 11 6 9 10 14 10 1 2 3 1 1 3 4 5 5 6 9 2 1 2 3 4 1 2 3 8 5 6 7 8 4 7 8 12 1 5 6 9 8 7 8 9 3 6 8 9
2 6 9 11 6 5 7 9 10 6 10 14
10 9 10 2 4 5 9 4 6 7 4 5 8 9 2 1 2 3 6 1 2 3 8 1 3 4 1 5 6 7 3 1 4 7 7 1 3 4 2 7 7 9 10 4 3 5 7 8 6 1 4 6 5 9 10 14 3 5 6 8 9 4 5
7 10 5 7 10 8 2 4 5 7 9 10 14 4 1 3 4 7 9 1 2 3 6 5 6 8 8 4 6 7 1 6 8 9 10 4 5 6 4 1 2 3 2 4 5 6 9 2 1 3 4 7 5 7 8 3 1 3 4 6 7 10
15 8 4 6 7 9 2 4 5 1 1 2 3 4 1 2 4 5 7 9 13 3 3 6 8 12 5 2 4 5 7 4 6 8 1 4 4 5 7 3 3 5 7 9 5 1 2 3 6 1 2 3 4 4 1 4 6 5 5 6 8 2 1 3
4 9 1 2 3 10 6 1 2 3 4 6 10 15 2 1 3 4 9 1 2 3 5 7 8 9 1 1 3 4 8 1 3 4 3 6 8 11 7 2 4 5 10 7 10 15
10 10 10 5 7 8 4 5 6 8 6 3 5 7 7 1 3 6 2 5 7 9 3 3 4 6 9 8 10 15 8 7 9 12 5 1 4 7 1 1 2 3 2 4 4 6 9 5 7 8 10 8 1 5 6 9 8 5 6 8 5
1 4 6 2 6 7 8 4 1 2 3 9 3 4 5 10 4 5 6 6 1 3 5 9 1 9 10 13 5 4 5 6 8 4 7 10 6 4 5 7 3 6 9 13 9 3 5 6 4 3 4 5 10 5 6 9 2 1 3 4 4 7
3 5 6 4 3 4 6 8 6 9 12 6 1 2 3 1 2 1 2 3 4 8 9 10 14 1 9 10 14 6 6 10 14 7 1 2 3 10 9 5 9 13 6 2 4 6 10 5 6 9 8 6 7 9 4 7 8 12 3
3 4 5 7 6 7 8 1 4 5 7 2 7 8 11 5 1 4 7 8 7 2 4 6 2 1 4 6 6 1 3 5 4 1 4 7 1 1 3 4 9 1 2 3 10 7 8 10 3 7 8 10 4 5 4 5 6 1 5 6 9 7 8
9 13 8 9 10 12
10 9 8 1 2 3 4 1 3 4 2 9 10 12 6 5 8 12 7 1 4 6 3 6 8 10 10 9 10 14 1 1 2 3 9 6 9 12 8 2 6 7 8 1 5 6 7 7 5 9 12 10 1 2 3 4 1 3 4
6 1 3 4 5 9 10 11 9 7 10 12 6 3 4 5 7 10 7 8 11 6 5 7 8 7 6 7 9 1 1 3 4 5 1 3 4 3 10 3 5 6 9 1 2 3 8 4 6 8 9 5 1 2 5 7 7 8 9 9 9
10 13 1 5 6 9 4 1 2 3 10 4 7 10 2 1 3 4 3 1 3 4 8 5 7 10 2 4 1 4 6 9 6 8 10 6 8 6 10 12 2 6 9 12 5 3 4 5 7 7 8 10 10 3 5 7 9 5 6
9 1 9 5 6 7 1 9 3 4 5 3 9 7 8 9 3 5 7 10 5 5 6 8
10 4 5 7 9 12 4 1 3 5 6 4 5 7 1 4 5 7 8 1 3 5 6 7 4 5 7 3 1 3 5 10 1 2 3 2 4 6 7 6 4 5 6 4 3 5 6 9 7 8 9 3 10 8 10 14 8 6 7 10 5
4 7 10 1 6 8 10 15 4 6 3 5 6 4 5 7 10 3 4 6 7 2 5 8 10 10 2 5 8 11 6 1 2 3 8 1 3 4 7 8 10 14 3 1 2 3 9 1 3 5 4 9 10 12 1 1 2 3 5
5 6 7 10 7 8 12 9 5 4 5 7 9 4 5 6 8 6 7 10 7 3 4 5 2 5 9 13 10 5 6 7 1 5 7 9 6 8 9 13 3 1 3 4 4 1 1 3 4 8 9 10 15 3 7 8 10 6 5 7
8 6 3 3 4 6 9 5 7 9 1 3 5 6 6 3 4 5 10 3 4 6 8 6 8 10 9 1 4 6 8 8 9 10 15 9 1 2 5 6 7 9 13 4 3 4 5 7 6 9 12 2 2 4 6 3 1 3 4 10 4
6 9
10 3 1 2 4 5 5 6 10 12 8 3 4 5 2 5 1 3 5 4 4 6 8 6 2 1 3 4 8 4 6 7 10 9 10 12 1 5 6 7 4 5 8 11 7 1 2 3 10 7 1 3 4 1 9 10 13 3 5
7 8 4 2 4 6 6 7 8 11 9 1 3 4 8 1 4 5 5 6 7 9 2 1 3 4 10 1 2 3 1 9 9 10 14 1 9 9 10 12 10 10 5 6 8 2 5 9 12 7 4 6 7 9 1 3 4 4 9 10
12 8 4 7 10 6 9 10 13 5 1 3 4 3 6 10 12 1 4 5 6 10 5 4 6 9 4 7 8 12 9 9 10 11 1 8 9 10 10 5 6 9 3 5 6 7 7 5 6 7 8 4 5 6 2 4 7 8
6 1 2 3 2 2 1 2 3 10 4 7 8 5 9 6 7 10 7 2 4 6 4 1 2 3 1 4 5 7 8 7 9 10
10 5 4 7 10 12 1 1 2 3 8 1 2 5 6 3 5 6 3 3 5 7 1 10 5 6 7 1 8 3 4 5 9 4 7 8 12 5 4 6 8 1 3 4 5 10 1 2 3 7 3 5 7 2 2 4 5 6 4 7 10
9 3 4 5 3 5 6 9 6 6 7 8 9 3 1 3 4 10 8 10 15 7 3 5 7 2 5 8 9 5 5 7 10 1 7 1 4 7 8 1 1 4 7 2 5 9 12 4 2 4 6 3 4 6 8 7 1 2 3 5 3 5 6
9 8 9 13 10 1 3 4 5 5 5 7 9 1 1 4 5 8 5 7 9 10 5 6 9 6 9 10 11 4 1 1 3 4 8 1 4 5 6 1 2 4 7 4 5 6 7 9 3 5 7 7 9 10 14 1 5 9 11 5 9
10 12 6 9 10 13 8 5 8 10 4 2 4 5
10 5 8 6 8 11 5 1 3 4 10 1 2 5 7 1 3 4 3 5 8 11 8 1 5 6 8 8 1 4 7 10 1 2 3 7 4 5 6 6 1 3 4 5 1 2 3 4 7 9 11 9 6 7 9 5 5 1 2 3 9 4
7 9 2 4 5 7 4 5 6 8 8 1 2 3 7 1 5 6 9 3 1 2 3 10 1 3 4 4 5 6 9 6 1 2 3 5 7 8 12 7 1 4 6 9 9 5 9 12 8 5 6 7 10 1 2 3 7 1 2 4 5 7 10
15 6 1 2 3 2 5 8 11 1 9 10 11 3 1 3 4 5 4 5 9 12 9 6 7 10 8 2 4 5 10 1 3 4 6 5 6 8 8 6 5 6 9 7 9 10 12 8 3 5 7 4 1 3 4 2 1 3 6 10
7 8 10 3 5 7 8 5 3 5 6 1 9 1 3 6 7 8 3 5 7 10 4 5 6 9 1 2 3 1 5 6 7 4 3 5 6 6 6 7 8 7 1 3 4 9 5 1 3 4 1 7 8 9 4 5 6 7 10 8 10 11 7
7 10 13 3 5 6 7 9 1 3 4 2 1 3 4 6 1 2 3

10 10 4 7 8 10 10 1 3 6 7 7 10 15 2 5 9 13 6 1 2 3 5 6 10 14 8 5 8 11 1 3 5 7 9 3 5 6 3 5 6 8 5 7 3 4 6 1 4 5 7 8 1 2 5 2 1 2 3 6 3 5 7 3 7 1 2 3 4 7 8 9 8 1 3 4 6 2 1 2 3 7 1 3 4 1 5 6 8 9 1 2 3 3 1 3 4 5 4 5 6 6 8 6 9 11 4 4 6 7 6 6 10 11 9 9 10 12 2 3 5 7 10 1 2 3 10 5 9 10 11 7 7 9 13 9 9 10 12 1 3 4 6 6 1 2 3 10 6 9 11 8 5 6 7 3 7 9 12 4 1 2 3 2 3 5 6 5 1 3 5 7 2 4 5 7 3 4 5 7 4 6 9 12 9 1 3 4 3 8 1 3 4 9 4 5 7 2 3 5 7 1 6 6 10 14 1 3 7 9 12

10 7 6 2 4 5 9 5 6 8 10 1 4 5 4 1 2 3 8 1 3 4 5 8 10 12 7 4 6 7 9 2 6 8 12 8 4 6 9 7 8 9 13 3 1 3 4 10 5 6 7 9 4 6 7 4 6 9 12 6 3 5 7 5 6 7 8 6 4 3 4 5 3 5 7 10 8 7 8 12 7 1 2 3 10 5 7 8 6 6 10 13 9 4 9 10 13 3 6 8 9 5 3 4 5 7 7 9 12 6 1 2 3 8 1 3 4 10 3 4 5 1 1 2 5 9 8 9 12 3 3 6 7 8 1 5 7 9 8 9 10 12 9 8 4 5 6 4 4 7 10 2 4 6 7 9 8 10 14 6 2 4 6 1 7 9 12 5 8 9 11 3 1 2 3 7 4 6 8 5 6 3 5 7 9 2 4 6 10 2 4 6 2 9 10 13 1 1 4 7 5 2 5 6 9 9 5 9 13 8 3 5 6 7 1 2 3 5 7 9 13 4 5 7 8 11 9 1 3 4 7 7 8 12 3 4 6 9 2 1 5 7 9 9 1 4 7

Instance 5:

15 8 2

10 6 3 6 8 9 8 7 10 13 6 5 6 9 4 9 10 14 2 4 6 8 7 9 10 15 6 8 1 3 4 7 4 6 7 5 1 2 3 1 3 4 5 6 2 4 6 3 7 8 12 7 5 1 3 4 1 4 7 10 3 6 10 12 8 5 8 11 4 5 7 10 7 1 4 5 6 1 4 5 7 5 8 10 14 2 1 4 7 3 6 8 12 1 6 7 10 7 4 6 8 8 1 3 4 6 1 3 5 3 2 1 2 3 3 8 9 12 4 1 3 4 2 2 7 8 10 1 1 2 5 6 8 9 10 11 2 1 3 5 6 5 8 9 5 5 6 7 4 1 3 4 3 4 5 6 1 8 6 7 10 4 2 1 2 5 4 3 5 6 5 5 9 11 7 7 9 11 1 3 5 6 9

14 8 5 3 4 6 2 2 4 5 3 1 4 6 1 4 7 9 4 3 4 6 6 7 9 12 7 4 7 9 8 4 5 6 6 7 2 4 5 6 1 3 4 8 7 8 12 5 6 7 8 4 5 6 7 3 7 10 11 5 6 1 3 4 1 9 10 15 2 1 2 3 5 4 5 6 8 6 9 10 7 7 4 6 9 4 4 6 7 1 5 8 9 6 5 6 7 8 3 5 6 3 3 5 7 2 1 3 4 1 8 6 7 10 7 4 9 10 13 6 2 4 5 5 5 7 9 2 8 9 12 8 1 3 4 3 7 10 14 1 1 3 4 6 2 1 3 4 7 9 10 11 6 5 8 11 1 7 8 12 4 5 9 12 5 3 4 5 2 7 4 6 7 3 1 3 4 6 5 7 8 9 8 2 4 5 2 1 3 5 7 2 4 5 6 3 5 6 4 2 4 5 7 5 6 7 9 3 4 5 7 1 6 10 12 6 4 6 7 7 5 8 12 8 7 8 12 4 6 9 13 4 2 4 6 8 4 1 3 4 6 4 5 7 5 1 3 4 3 2 3 5 6 8 4 5 7 4 1 4 6 4 8 1 3 4 4 4 5 7 6 8 9 12 5 3 4 5 7 6 7 9 13 1 3 4 6 4 6 8 12 2 7 8 9 3 4 5 6 5 6 8 12 8 3 4 5

13 1 3 1 2 3 1 8 6 8 12 6 3 8 9 10 4 7 8 11 2 8 10 14 8 5 8 11 7 1 2 3 5 7 8 12 7 4 4 5 6 1 7 8 11 6 9 10 12 3 5 7 8 5 4 7 10 2 3 4 6 7 1 3 4 6 4 2 4 5 2 3 5 6 8 1 2 3 6 2 4 5 3 6 9 10 1 1 3 6 5 7 1 2 3 3 1 2 3 5 1 2 3 2 1 3 4 8 1 3 4 1 6 3 4 6 3 7 1 3 4 1 1 4 5 5 3 5 6 3 8 9 10 11 2 9 10 15 7 1 2 3 1 2 6 9 12 6 5 4 6 9 2 4 5 7 6 4 7 10 3 4 5 7 4 4 5 7 8 2 4 5 8 5 3 5 7 2 5 8 10 4 5 7 10 8 1 4 6 1 3 5 7 3 4 6 8 6 5 6 7 7 5 7 10 5 5 1 4 5 6 4 5 6 1 4 7 8 7 1 3 4 4 1 3 5

12 8 4 1 4 7 8 1 3 4 1 4 7 8 7 9 10 11 6 1 2 3 3 6 10 11 2 1 3 4 5 4 6 7 3 2 9 10 12 7 7 9 13 6 2 4 5 8 6 1 2 5 5 5 8 12 2 1 3 4 1 5 7 8 8 9 10 14 7 1 3 4 3 6 8 11 4 4 5 6 1 6 1 2 3 4 6 5 6 9 5 1 3 4 1 5 6 9 2 1 3 4 5 4 1 2 3 1 5 6 7 6 6 7 9 7 5 7 10 3 3 4 5 1 5 7 9 10 8 7 8 10 11 1 3 5 6 2 7 8 11 5 3 4 5 4 2 4 5 6 9 10 13 3 1 3 4 8 6 9 10 6 4 5 6 9 3 1 3 4 6 7 10 12 2 1 2 3 7 5 6 7 8 3 4 5 8 1 4 5 6 6 3 4 6 5 7 8 12 8 6 9 12 2 1 3 4 3 1 3 4 7 1 3 4 4 3 4 5 6 8 9 10 12 5 4 5 6 6 1 2 3 4 2 4 5 7 5 6 8 3 2 4 5 1 1 5 7 9

12 4 7 1 2 3 3 6 8 10 8 1 2 5 1 6 9 13 4 4 2 4 6 8 7 9 12 6 1 3 4 3 3 4 5 3 5 1 4 5 2 3 4 5 4 5 7 9 5 3 1 2 3 8 1 3 4 2 6 7 10 4 1 2 5 6 5 8 12 2 6 1 2 3 2 1 2 3 2 6 1 2 3 2 4 7 9 6 6 1 3 4 8 8 10 14 1 3 4 5 7 9 10 12 2 5 6 7 5 5 6 8 5 8 5 6 7 1 5 6 7 3 8 10 13 6 6 10 13 7 8 9 10 7 4 5 7 9 6 5 8 12 1 4 5 7 5 1 2 3 8 1 2 3 7 7 9 10 2 5 6 7 4 8 1 3 4 7 1 3 4 4 5 9 11 1 6 10 15 3 3 5 7 9 4 7 10 14 7 4 5 6 6 6 4 5 7 3 6 8 10 4 4 7 9 8 8 9 13 1 1 3 4 5 1 2 3

15 7 5 1 2 4 7 6 7 8 3 8 10 13 2 6 10 12 6 1 2 3 4 5 9 12 1 1 4 7 3 4 1 3 4 7 1 3 4 3 4 5 6 6 8 3 4 6 3 4 7 8 7 7 8 9 2 1 3 4 4 3 5 6 6 4 6 9 5 5 5 6 7 6 5 6 9 1 1 2 3 7 7 8 10 2 1 3 4 6 2 1 2 3 1 5 6 7 8 6 7 9 4 7 8 12 6 2 4 5 7 1 3 5 7 8 4 7 9 7 5 8 9 5 3 5 6 2 3 4 5 6 3 4 6 3 1 3 4 4 1 2 5 4 3 9 10 13 5 1 2 3 4 1 3 4 7 4 5 6 1 6 3 5 7 8 4 6 7 10 8 5 7 8 3 1 3 4 7 7 8 10 1 7 9 11 2 5 6 7 5 9 10 14 6 3 5 7 6 8 4 5 7 1 4 5 6 3 1 2 3 5 9 10 11 6 5 8 10 4 7 8 9 6 7 8 9 11 1 7 8 9 6 1 2 3 4 2 4 5 2 3 5 6 3 9 10 11 7 8 5 9 11 6 1 4 5 1 7 10 12 5 4 6 8 3 6 8 11 2 3 4 5 7 3 4 6 6 3 1 2 3 2 1 3 6 7 5 9 11 5 1 3 4 1 1 3 6 8 6 7 9 1 6 6 8 9 4 2 6 7 10 3 1 2 3 8 4 7 8 1 7 9 10

15 1 7 1 4 7 5 3 8 9 11 6 5 7 8 5 1 2 3 2 1 2 4 7 7 8 12 3 6 4 7 10 4 1 2 3 3 6 7 10 6 5 5 6 8 8 5 6 7 3 8 9 11 6 5 8 11 2 3 4 5 7 1 3 4 5 7 3 5 7 2 4 7 9 6 8 9 10 5 3 5 7 8 6 9 12 3 6 7 10 13 5 3 4 5 7 6 7 8 2 1 9 10 14 6 5 6 7 8 2 1 2 3 7 6 7 8 6 1 2 3 4 1 3 4 1 5 8 11 5 2 4 6 8 6 7 10 3 1 2 3 5 2 4 5 6 3 5 9 10 1 5 6 7 6 6 7 9 5 6 8 10 2 3 5 6 8 1 5 9 13 4 4 9 10 14 5 6 7 9 6 5 8 12 8 5 6 9 5 5 8 9 10 6 7 9 12 3 2 4 6 8 4 6 7 7 7 8 12 7 7 9 10 15 3 5 9 12 6 1 3 4 8 3 4 6 2 5 9 11 1 3 5 6 4 5 6 7 6 1 5 7 10 7 4 7 8 5 6 7 8 3 1 3 4 6 7 8 10 8 4 7 9 2 4 3 4 6 7 1 3 4

8 2 4 1 2 3 6 1 3 4 4 2 1 2 3 4 7 8 12 8 2 4 6 5 5 7 8 1 7 7 8 9 7 7 5 6 8 5 5 6 9 6 5 8 12 1 6 9 12 4 3 5 6 2 6 8 11 8 4 7 9 7 1 9 10 14 5 1 2 3 3 6 8 12 6 9 10 13 4 5 6 7 2 5 6 7 8 5 6 8 4 6 1 2 3 5 6 7 8 1 9 10 12 8 4 5 7 5 2 1 3 4 4 6 9 13 6 1 3 4 1 8 10 11 8 1 3 4 5 4 1 2 3 5 4 6 9 6 1 3 4 8 1 3 4 3 8 10 15

9 2 4 1 3 5 2 4 5 7 3 3 1 2 5 2 7 10 11 7 4 6 9 1 1 6 10 13 7 4 1 3 6 8 6 10 15 2 5 7 8 1 2 4 5 7 3 4 6 5 1 3 4 3 6 10 12 7 4 1 3 4 1 6 8 10 2 3 5 6 5 1 3 6 8 5 6 7 3 1 2 3 7 3 5 6 8 3 1 3 4 5 5 6 9 1 5 8 10 8 1 2 3 2 2 4 6 6 3 4 6 4 1 4 5 7 1 3 4 1 4 1 2 3 1 8 1 3 4 5 6 4 6 7 1 1 2 3 3 8 10 12 5 1 3 4 8 4 6 7

12 4 5 5 9 11 8 7 8 12 4 2 4 6 3 1 3 5 2 1 9 10 15 5 2 4 5 4 6 1 2 3 3 6 8 9 8 6 8 10 7 9 10 13 2 6 1 2 3 3 9 10 14 3 2 3 4 6 5 1 2 5 7 8 9 13 1 5 8 10 13 7 4 3 4 5 3 6 9 10 2 4 5 6 5 9 10 12 6 6 9 12 7 1 3 4 1 1 3 6 3 7 8 10 12 3 1 2 3 5 2 4 5 4 2 1 2 3 8 3 5 7 5 1 3 4 4 6 10 15 4 1 7 8 11 8 9 10 12 2 8 9 11 3 3 4 5 4 6 4 5 6 7 7 10 13 8 3 4 6 1 5 9 12 4 7 9 10 12 1 1 4 5 3 1 3 4 5 3 4 5

9 5 4 8 9 13 1 7 9 11 6 3 5 6 5 4 5 7 8 6 7 8 3 5 9 10 11 7 5 6 9 6 1 2 3 7 3 1 2 3 5 4 5 7 4 2 4 6 7 5 6 9 1 1 4 7 8 7 9 12 2 1 3 4 6 3 3 5 7 8 4 5 6 6 1 3 4 7 4 6 7 1 4 6 9 2 2 4 6 8 8 1 3 4 1 1 2 3 3 3 5 7 4 3 4 5 5 1 2 3 6 1 3 4 2 4 5 7 7 1 2 4 7 4 5 8 11 2 1 2 3 5 1 2 4 3 1 3 5 6 1 3 4 7 1 3 4 8 5 7 10 1 8 5 8 11 2 1 4 5 6 7 3 4 5 6 7 7 9 10 6 1 2 3 5 3 4 6 1 1 4 7 8 5 7 10 3 1 3 4 8 6 6 1 2 4 3 6 10 14 2 6 9 10 1 1 3 4 7 5 7 10 5 3 4 5 5 3 1 3 4 4 4 5 6 2 7 8 10 1 6 10 12 8 1 3 4 8 7 9 10 14 3 1 2 4 4 7 10 14 1 5 9 12 6 7 8 9 5 1 2 3 8 1 4 5 2 2 4 6 1 6 5 6 7 4 1 3 4 5 2 5 9 11 8 5 6 8 6 5 6 9 8 5 7 8 10 4 1 2 3 2 5 7 8 1 5 8 9 3 2 4 6 6 4 5 7 8 9 10 13 7 1 3 4 8 7 7 8 9 2 5 9 10 1 7 9 13 5 5 6 9 4 4 5 7 8 5 6 7 3 9 10 13 6 2 4 5 3 8 8 10 15 7 1 3 4 4 1 2 3

14 8 2 1 3 5 8 1 2 3 7 6 7 9 1 5 6 9 4 1 3 6 3 1 4 7 6 1 3 4 5 1 3 4 2 8 5 7 9 5 7 9 11 4 4 6 9 11 7 1 4 5 8 1 2 3 1 6 7 9 2 4 4 7
10 1 4 5 6 6 6 1 2 3 2 3 5 7 3 7 8 9 4 2 4 5 7 4 6 9 8 1 3 4 1 8 1 2 3 1 1 3 4 6 4 7 3 5 6 1 3 5 6 8 7 8 10 5 9 10 12 7 3 1 3 4 2
1 3 4 1 1 3 4 7 7 10 13 4 6 7 10 5 1 2 3 8 1 2 3 5 2 1 3 4 5 5 6 7 6 3 5 7 1 1 2 3 8 3 5 7 4 2 3 4 6 3 7 8 9 6 4 7 10 5 6 10 15 8
2 1 3 6 7 1 2 4 1 5 8 11 5 1 2 3 4 4 7 10 6 1 3 6 3 1 3 4 8 1 2 4 5 1 5 8 11 7 3 4 6 4 5 8 10 5 6 7 8 6 1 3 6 8 8 7 8 9 7 1 2 3 6
5 6 8 5 5 6 9 3 6 7 8 4 1 4 7 2 5 6 7 1 1 2 4
10 2 7 3 5 6 5 1 3 4 6 5 5 6 7 8 3 4 5 7 5 8 11 2 3 5 6 6 3 5 7 3 7 8 10 3 4 1 3 4 5 8 10 14 2 1 3 4 4 2 6 8 12 8 6 9 11 5 1 2 5
4 7 10 15 8 5 6 8 12 8 7 8 9 7 2 4 5 6 2 4 5 2 3 4 6 3 3 5 7 4 6 7 9 1 4 5 6 4 3 3 4 5 5 5 6 9 6 2 4 6 1 5 6 9 1 6 5 6 9 1 3 5 6 9
7 5 5 9 12 1 1 2 3 8 6 7 10 7 1 2 3 3 1 3 4 6 5 6 8 2 5 7 9 7 1 1 3 4 7 1 2 3 2 5 6 8 6 8 10 12 3 5 6 8 5 4 5 6 4 6 7 8
10 4 4 6 7 8 6 1 2 4 7 4 6 7 5 1 2 3 4 5 1 2 3 7 1 3 4 8 1 2 4 4 1 3 4 7 1 1 3 4 5 1 2 3 7 1 2 3 8 7 8 11 4 4 6 7 2 1 4 6 6 3 4 6 2
6 8 10 11 8 3 4 5 8 8 6 8 10 4 5 9 10 5 7 9 10 2 5 6 8 3 6 10 14 7 7 8 9 6 1 3 4 1 1 3 6 3 6 4 6 7 2 6 10 12 8 4 6 8 7 5 3 4 6 6
6 10 12 7 7 9 13 1 6 8 12 8 4 5 6 3 7 8 10 2 1 2 3 6 8 6 8 10 7 1 3 4 2 1 4 5 5 8 9 11 4 5 8 10 3 3 4 5 8 1 1 3 4 2 1 3 4 6 1 2 3
7 2 4 6 8 8 10 11 5 5 6 8 3 3 5 6 4 6 9 10 6 6 6 10 11 1 1 2 3 3 7 8 12 4 1 3 4 8 4 6 7 5 1 2 3

Instance 6:

15 10 2
13 6 8 6 7 8 1 1 2 3 5 5 6 9 3 1 3 4 2 3 4 6 7 9 10 15 10 10 8 10 11 1 3 5 7 8 3 4 6 9 8 10 13 6 7 9 13 5 7 8 12 7 5 8 12 4 4 6
9 2 1 3 4 3 8 10 15 7 6 1 3 4 3 6 7 8 10 1 4 5 5 5 6 7 2 5 6 9 1 1 3 4 8 7 8 10 5 9 7 9 10 3 4 5 7 8 9 10 13 4 3 4 5 6 8 9 12 3
10 7 8 9 9 2 4 6 3 4 7 10 6 7 9 10 11 5 9 10 13 8 3 5 6 9 1 3 4 2 9 10 15 4 8 10 13 6 8 1 2 3 2 7 8 9 4 4 7 9 9 5 9 11 3 1 2 3
10 7 9 12 3 7 5 7 8 4 2 4 5 10 1 4 6 3 6 8 10 15 5 1 2 3 2 1 3 4 5 9 5 8 12 3 1 3 4 10 1 3 4 2 6 7 8 6 3 4 6 10 7 3 5 7 9 5 7 8 8
6 7 10 6 6 7 9 5 6 7 9 1 8 10 15 3 7 8 11 10 4 6 7 4 5 8 9 2 5 8 11 1 3 1 3 4 5 6 1 2 3 9 9 10 14 1 3 5 7 3 7 8 12 7 8 10 15
15 6 3 9 10 13 7 9 10 15 9 3 5 7 8 1 4 6 10 1 2 3 5 7 10 14 8 1 1 3 5 3 2 4 6 8 1 4 7 10 2 4 5 6 1 4 5 9 9 10 11 4 1 3 4 5 4 7 8
8 10 6 8 9 7 1 4 5 5 1 3 4 3 5 7 9 6 2 4 5 1 3 5 6 2 3 5 6 9 5 6 8 7 5 4 7 8 3 1 2 5 7 7 8 12 9 4 6 8 6 2 4 6 8 5 7 8 4 1 2 4 8 8 1
2 3 5 6 10 13 1 1 3 5 7 1 2 3 6 4 5 7 9 5 6 8 4 3 5 6 2 6 8 10 6 2 1 4 5 8 6 8 11 5 3 5 7 1 6 7 8 10 1 3 4 7 1 3 6 9 6 9 10 13 2
1 3 4 5 4 6 7 10 5 9 10 9 1 4 6 1 4 5 6 4 4 5 6 3 3 5 7 8 5 7 10 1 9 1 2 3 9 7 8 10 11 8 1 2 3 5 1 2 3 2 1 2 3 10 1 3 4 6 4 6 9 1
5 6 9 3 2 4 5 4 1 2 3 3 9 1 2 3 1 1 4 5 8 4 7 8 5 7 4 7 10 1 1 3 4 8 5 7 9 2 9 10 13 6 1 3 4 2 4 9 10 12 9 1 2 4 8 8 6 7 10 2 7 10
12 9 4 5 6 10 2 4 5 3 4 5 6 7 2 4 5 1 3 4 6 5 4 5 6 8 7 9 10 11 4 7 10 13 9 6 8 11 5 5 6 7 6 1 3 4 8 3 5 6 1 2 4 6 3 5 9 11 1 8 1
2 3
10 6 9 5 6 8 5 1 2 4 1 4 5 6 10 1 2 3 4 4 5 7 8 1 3 4 2 6 4 5 6 1 4 7 8 10 3 5 8 9 9 1 2 3 4 5 8 12 10 5 7 8 2 2 4 5 1 3 5 7 8 1 4
6 7 6 9 10 6 1 2 3 5 3 4 6 4 4 1 3 4 1 5 6 8 9 5 8 9 6 3 4 6 3 2 3 5 7 7 1 2 3 1 3 4 6 3 10 5 7 9 5 1 2 3 3 4 6 9 1 10 3 4 5 2 3 5
8 11 9 3 5 6 6 2 5 7 10 6 6 10 13 1 5 7 9 9 9 10 11 5 7 10 12 7 1 2 3 4 10 5 6 9 9 4 5 6 1 7 8 9 2 7 9 11
13 2 6 5 7 8 9 1 3 4 7 2 8 9 10 4 7 10 12 5 1 2 3 7 7 10 11 6 6 8 9 9 7 8 9 1 5 9 11 5 9 4 5 7 7 1 3 4 8 3 4 6 4 9 10 12 5 7 10
14 9 5 6 7 10 1 1 3 4 9 6 8 11 7 1 2 3 10 3 5 7 3 3 5 7 6 1 3 4 4 9 10 12 8 7 8 9 6 6 1 3 4 1 1 2 3 7 5 7 9 9 3 5 6 8 6 10 12 10
1 2 3 5 9 1 2 3 4 4 7 10 8 5 7 10 3 1 2 3 7 1 3 4 5 9 1 2 3 2 1 2 3 8 5 6 7 5 4 6 7 6 5 6 8 1 8 1 2 3 6 7 1 2 5 5 1 3 4 9 4 7 10 3
3 5 7 6 4 7 8 10 5 6 8 1 5 1 3 4 3 5 1 3 4 8 4 5 6 1 5 7 10 9 6 3 4 5 8 5 9 12 2 3 5 6 3 5 6 8 9 1 4 5 10 1 4 5 7 1 3 4 1 5 6 9 5
3 4 5 6 3 1 2 3 1 1 2 5 6 1 3 4 2 1 3 4 5 4 5 7 8 3 5 6
10 9 2 3 4 6 5 1 2 3 9 5 6 9 7 3 5 7 3 6 10 15 10 7 8 12 6 3 5 6 8 1 3 4 4 1 2 3 7 5 6 9 11 2 4 5 7 8 8 9 13 3 3 4 5 1 4 7 9 6 8
9 12 4 5 6 7 7 6 7 8 10 10 1 2 3 1 7 10 11 2 5 9 12 9 4 5 6 5 3 4 5 4 3 4 5 7 3 4 5 6 4 1 3 5 6 4 7 8 7 3 4 5 9 5 7 8 5 3 5 6 8 2
4 6 5 3 6 10 14 8 1 2 3 5 1 3 4 6 1 4 5 1 7 8 11 1 9 7 8 12 3 2 1 3 4 8 8 9 11 4 8 10 15 8 3 5 6 9 1 5 8 12 8 7 9 12 9 2 4 5 2 5
9 12 7 6 8 9 10 6 7 10 4 1 3 4 7 5 7 10 15 4 7 8 11 1 4 7 9 3 3 5 7 6 4 7 8 7 1 2 3 10 2 4 6 8 5 7 8 10 3 7 8 9 8 2 4 6 9 7 8 10
6 2 4 5 4 4 5 6 10 1 2 3 1 4 6 7
12 1 9 8 10 12 7 8 4 6 7 6 6 8 11 4 8 10 13 2 4 7 9 5 8 9 11 7 5 6 9 3 5 9 12 2 10 4 6 7 6 2 4 5 6 5 4 6 8 7 5 6 9 9 3 5 6 2 5 6
7 10 4 5 7 1 6 10 15 4 1 5 7 10 9 4 5 7 8 2 4 6 7 5 9 10 1 7 4 7 10 1 6 3 4 5 2 4 3 4 5 5 8 9 10 5 4 7 10 15 7 1 2 3 1 5 9 11 6
5 8 9 2 2 4 5 2 6 5 6 9 9 7 8 10 1 3 5 8 11 1 5 2 4 6
13 4 9 1 3 4 1 7 9 13 4 1 3 4 7 9 10 11 7 9 3 4 6 8 8 10 13 7 7 8 12 2 1 2 3 10 1 2 3 6 5 6 7 1 1 2 3 3 5 7 9 11 2 7 9 12 7 6 7
8 3 10 5 9 11 6 5 6 8 2 5 8 10 5 6 6 8 9 9 7 8 11 3 9 10 14 4 1 2 3 10 1 3 4 3 7 5 6 7 5 6 8 10 8 5 6 8 1 5 3 5 6 3 4 7 9 13 3 7
8 11 2 8 9 11 8 8 6 7 10 2 5 8 9 5 6 7 8 10 6 7 9 6 6 9 11 3 1 4 6 4 1 2 4 9 7 8 10 5 4 1 2 3 10 1 2 3 7 1 4 7 8 1 3 5 6 3 5 6 10
6 5 6 9 8 3 5 6 3 4 5 7 7 3 5 7 1 3 5 6 5 1 3 4 2 4 5 7 9 5 8 10 4 1 3 4 10 1 2 3 9 2 1 3 4 8 3 4 6 6 7 10 14 7 1 2 3 1 6 7 10 9 1
2 4 10 4 7 9 4 8 9 11 5 4 5 7 7 9 5 6 7 5 6 10 15 6 1 2 3 7 6 10 15 8 3 4 6 3 7 9 10 4 5 7 9
13 2 10 2 4 6 9 2 4 6 4 4 5 6 9 3 5 9 10 8 9 10 13 1 4 6 8 2 3 3 4 5 1 3 5 7 3 4 5 6 9 9 1 3 4 8 6 7 8 5 1 5 6 7 6 5 6 8 4 3 4 6
10 1 3 4 9 3 5 6 7 9 4 5 7 3 8 9 11 4 3 4 6 2 4 5 7 8 3 5 7 7 3 4 5 5 7 8 11 7 10 2 4 5 1 3 5 7 4 9 10 11 6 1 2 3 9 5 7 9 2 1 3 4
8 8 9 11 5 2 1 3 4 4 7 9 13 3 1 2 3 5 5 7 10 9 6 10 15 1 4 1 2 3 7 7 5 9 12 9 8 9 11 10 6 7 10 1 8 9 10 5 1 2 3 6 3 4 6 4 2 4 5
1 10 1 2 3 1 10 9 10 12 7 2 5 9 13 10 1 2 3 9 5 6 7 7 1 4 7 6 9 10 12 1 7 8 10 8 8 10 11
10 4 10 1 4 7 8 1 2 3 3 1 3 4 5 1 3 4 5 5 1 3 4 8 7 8 10 3 6 8 10 10 1 3 4 9 3 4 5 4 4 7 8 12 8 3 5 7 6 5 7 10 9 1 3 4 1 8 1 3 5
2 2 5 6 8 10 1 3 4 1 4 1 2 3 2 1 3 4 5 6 5 7 8 2 10 2 4 6 9 3 4 6 6 8 9 10 14 3 1 3 4 5 7 9 12 7 7 8 12 4 4 5 7 9 5 7 8 2 3 1 3 4
10 4 6 8
10 2 2 1 2 3 10 6 8 11 1 10 5 6 7 10 5 3 5 7 8 1 3 4 4 6 8 11 3 7 10 15 7 8 9 12 9 7 8 10 1 9 10 15 6 7 8 9 2 3 5 6 10 1 3 4 9
9 5 6 9 4 4 5 6 6 9 10 11 5 3 4 5 10 1 2 3 8 3 5 6 7 9 10 12 3 5 6 7 2 1 2 5 8 7 1 3 4 8 4 5 7 6 3 5 7 9 7 8 11 5 4 7 8 2 5 6 7 4
4 7 9 3 2 4 6 5 1 8 10 12 9 1 2 4 7 3 4 6 3 1 3 4 6 1 4 7 8 9 6 10 13 1 1 3 4 7 4 6 7 8 3 5 7 5 4 7 10 2 6 10 13 4 7 8 12 10 4 6
9 3 6 1 2 3 9 3 4 6 8 3 4 5 6 8 1 2 3 2 1 2 3 7 4 7 8 3 2 4 6 1 1 3 4 9 1 2 4 7 2 3 5 6 10 1 3 4 8 1 3 4 7 9 10 15 6 5 6 9 4 3 4 5
9 1 3 4

13 10 8 6 7 10 2 1 2 3 1 4 5 6 9 2 4 5 3 3 5 7 5 1 3 4 6 4 5 6 10 7 8 9 4 4 6 7 7 4 5 6 2 9 5 8 12 6 5 6 9 9 9 2 4 6 6 2 4 6 8 5 6
8 10 8 10 12 1 8 9 10 5 1 3 4 3 6 10 12 7 6 9 13 2 2 4 5 6 4 7 8 9 5 4 5 6 3 3 5 6 7 7 9 10 10 5 8 10 6 7 8 10 7 7 5 9 11 5 5 6
7 8 1 3 4 1 3 5 7 6 6 10 12 4 1 2 3 2 7 8 10 6 10 7 8 10 9 2 4 6 1 5 8 9 7 1 2 3 4 5 6 9 6 9 10 13 2 5 5 7 8 7 1 2 3 2 2 5 9 10 8
5 6 7 5 9 4 5 6 7 6 8 10 8 3 4 6 2 5 9 11 3 1 2 4 4 8 4 7 8 4 1 2 3 6 1 2 3 9 4 5 7 4 9 7 10 12 3 2 4 6 2 4 5 7 7 7 9 11 3 1 1 3 4
10 1 2 3 9 6 7 8 10 2 2 4 5 6 3 5 7 5 6 8 10 10 5 8 12 4 3 5 6 8 9 10 15 7 3 4 5 3 3 4 5 9 5 7 8 1 1 3 5
15 7 1 2 4 5 3 3 5 6 6 7 8 10 10 5 6 8 9 5 6 9 8 6 8 12 7 5 6 7 2 7 3 5 6 2 4 7 9 4 1 1 2 4 6 9 10 11 8 3 4 5 9 1 2 3 3 2 7 10 13
10 1 2 3 3 4 5 6 3 8 1 4 5 2 1 3 4 6 6 7 8 4 3 1 3 4 8 6 10 11 10 2 4 6 7 8 9 13 1 5 2 4 5 3 10 1 2 3 2 1 2 3 8 2 4 6 6 1 4 7 9 9
1 3 4 4 4 6 8 2 6 8 9 10 6 10 11 5 6 10 11 9 7 1 3 4 9 3 5 6 3 9 10 13 5 1 3 5 8 7 8 9 10 5 7 10 2 7 10 14 6 9 10 14 1 1 2 3 7
7 1 2 3 2 6 7 8 3 3 4 6 10 1 2 3 5 1 2 4 8 1 2 3 1 7 9 10 3 9 7 9 12 8 8 10 13 1 5 8 11 2 8 3 5 6 10 6 7 8 8 7 4 6 7 10 1 2 3 8 6
8 9 1 3 5 6 4 1 3 4 5 4 6 9 6 1 2 5 3 6 9 11 5 6 4 7 9 7 7 8 9 2 1 3 4 1 5 6 7 8 1 2 4
14 1 1 1 4 5 3 10 1 3 4 8 6 9 12 4 4 6 9 9 1 1 3 4 5 5 6 7 2 1 2 3 4 5 7 10 6 4 5 6 3 1 2 3 8 5 7 8 10 6 7 10 7 1 2 3 2 2 6 7 10
1 6 7 9 7 3 5 6 8 9 3 4 5 5 2 4 5 7 1 3 5 2 3 4 5 6 1 4 6 4 1 3 4 5 1 5 6 7 9 3 4 5 4 7 9 13 2 6 9 10 7 3 5 7 5 9 7 8 10 1 1 2 3 4
4 5 6 7 9 10 13 3 1 2 3 6 6 4 5 7 10 5 7 8 8 8 9 12 2 4 5 7 7 6 7 9 4 6 9 11 1 4 2 4 5 8 5 9 10 15 6 3 4 6 8 3 5 6 4 1 3 4 3 1 4
6 1 4 7 10 10 4 5 6 9 1 2 3 5 5 6 9 10 4 4 7 8 9 4 7 10 2 1 2 4 10 4 5 7 4 1 6 10 12 7 7 9 13 10 6 9 13 9 1 2 3 5 10 2 4 6 1 1 3
4 8 4 5 7 9 1 3 4 5 6 9 11 2 4 3 4 6 3 1 3 4
14 6 1 1 3 4 2 5 8 12 8 1 2 3 9 8 9 13 5 7 8 10 6 2 4 6 7 6 6 7 9 5 4 6 9 9 3 4 5 10 2 4 5 7 3 5 6 2 4 5 7 1 8 10 13 6 6 7 8 11 3
1 2 3 4 8 10 12 7 2 4 6 9 2 4 5 5 1 3 5 5 8 6 9 10 1 3 4 5 9 1 3 4 7 6 8 9 3 5 8 12 10 5 3 5 6 2 5 7 8 6 8 9 13 4 4 6 9 9 3 4 6 8
4 7 10 10 5 9 10 3 6 9 12 1 1 2 4 7 3 4 5 8 4 1 4 6 9 7 8 10 2 1 3 4 8 2 4 5 10 7 9 12 7 1 2 3 5 5 6 7 3 4 5 6 9 2 1 2 3 5 5 6 7
9 4 7 10 7 3 4 6 10 5 9 11 1 7 8 12 8 1 2 3 3 4 7 8 6 2 4 5 7 2 1 2 3 10 1 3 4 4 1 4 5 7 2 4 5 1 4 5 7 9 7 9 13 6 5 6 7 3 9 6 9 13
2 5 7 10 8 8 9 13 6 3 1 3 4 2 1 2 3 1 6 9 13 7 1 3 4 6 9 10 12 5 2 4 6 8 4 5 6 9 6 1 3 4 7 1 2 3 3 4 5 6 2 7 10 12 8 5 6 7 1 1 3
4 9 6 7 10 9 8 1 2 4 2 1 2 3 5 1 3 4 3 7 8 11 1 5 8 11 4 6 7 8 9 8 9 13 10 5 6 7 7 3 5 6 10 3 4 5 6 8 5 8 10 10 5 6 8 7 2 4 6 1 8
10 13 2 1 2 3 5 5 6 7 4 1 4 7 9 9 10 13 6 4 5 7 4 2 1 2 3 9 1 4 7 6 9 10 13 10 5 7 10
10 5 5 2 4 6 8 3 4 6 9 7 8 12 7 7 8 10 2 3 4 6 2 10 6 9 10 3 1 3 6 2 10 4 5 7 1 9 10 15 5 8 1 2 4 1 6 7 10 5 1 3 4 7 7 8 12 3 7
8 10 4 3 5 9 10 7 4 5 7 9 1 4 6 6 8 10 15 9 10 9 10 13 1 1 2 3 6 1 4 5 7 7 8 12 2 5 9 13 8 2 4 6 4 6 7 9 3 6 10 14 5 3 4 6 7 7 3
4 6 6 1 3 4 1 1 3 4 4 3 5 6 5 3 5 6 3 6 10 13 10 7 9 13 9 5 1 2 3 7 6 9 10 8 7 10 12 6 2 4 6 2 4 7 8 4 4 5 7 3 5 6 7 10 1 3 4 9 1
2 3 6 9 1 2 3 8 6 7 9 1 6 10 12 7 7 8 11 2 6 10 15 10 7 8 10 8 9 7 10 14 5 1 2 3 10 7 8 12 4 2 4 6 2 1 2 4 7 7 9 10 3 3 4 6 6 8
9 10

Instance 7:

20 10 2
17 8 9 1 2 3 6 4 6 7 3 5 6 9 10 2 4 5 7 4 7 9 4 3 5 7 8 1 4 5 5 3 5 7 1 5 5 6 7 10 4 1 4 7 10 1 2 3 8 7 8 12 7 5 7 9 6 3 4 6 5 6
10 12 9 6 7 10 3 7 8 12 2 1 4 5 1 3 5 6 6 8 7 9 10 7 5 6 9 10 1 3 4 1 7 8 10 6 7 10 11 9 4 6 9 6 1 5 6 8 9 6 8 11 3 1 3 4 5 1 2
5 6 4 6 8 7 8 10 14 3 2 2 4 6 6 1 2 3 9 1 4 5 6 9 1 2 3 7 5 6 8 8 6 9 10 4 1 4 7 2 4 7 10 10 1 3 4 1 2 1 2 3 9 8 7 8 10 9 9 10 12
7 1 2 3 5 7 10 12 1 5 9 12 3 7 9 13 4 1 3 4 10 7 9 12 2 1 2 4 5 8 6 8 9 2 9 10 13 10 2 4 5 7 4 5 7 6 3 5 7 10 6 1 3 4 5 5 6 9 8
7 9 10 3 6 9 10 4 8 10 15 9 7 8 11 7 6 10 14 10 3 5 6 2 1 2 3 1 4 7 9 1 1 5 7 8 10 1 9 10 14 9 7 9 13 2 7 9 13 8 4 5 7 4 1 3 4
3 4 7 9 10 1 2 3 5 4 5 6 7 4 5 6 6 2 4 6 5 4 5 7 9 5 3 5 7 8 2 4 5 2 5 9 12 10 3 5 7 2 10 3 5 6 3 1 2 3 6 2 5 9 13 8 6 8 9 4 1 2 3
7 1 3 4 6 5 6 8 1 3 5 6 3 10 2 4 5 2 5 6 9 3 6 7 9
19 7 6 4 7 8 9 3 5 6 5 7 8 12 10 5 8 12 1 3 5 7 7 9 10 14 8 1 2 3 1 7 4 6 8 2 7 7 8 10 4 1 3 5 2 2 1 2 5 6 6 9 11 2 7 6 10 12 9
7 9 12 7 7 5 6 7 3 8 10 15 5 1 2 3 10 1 2 3 8 6 9 11 2 9 10 11 4 1 3 4 8 7 1 3 4 9 6 8 12 10 6 9 12 1 5 7 8 5 1 3 4 3 5 6 7 4 7
8 11 6 1 2 4 1 5 1 3 4 8 6 5 7 9 1 8 10 11 9 1 4 7 2 7 9 12 4 6 7 10 10 1 2 3 7 2 4 6 3 3 4 6 9 6 1 3 4 8 1 2 3 4 1 4 6 3 1 3 4 9
3 5 6 10 1 2 3 5 5 6 9 2 6 10 15 7 1 3 6 5 5 4 6 8 7 5 8 9 8 6 8 12 2 1 4 7 4 5 8 11 3 3 1 2 3 2 8 10 12 8 1 3 6 6 7 7 9 12 2 1 4
7 9 6 9 12 6 1 2 3 1 4 5 7 5 6 8 11 2 10 1 3 4 9 3 4 6 2 7 9 10 13 5 2 4 6 9 6 1 3 4 9 7 10 12 7 1 2 3 4 1 2 4 8 4 6 8 10 1 2 3 1
4 5 7 3 1 2 3 2 6 10 11 3 10 5 8 10 2 1 2 3 6 2 4 6 6 3 3 5 7 9 1 3 4 8 2 4 5 4 3 5 7 2 7 8 10 6 1 2 3 1 1 7 8 11
13 7 2 7 8 12 4 9 10 12 3 7 10 13 6 5 9 10 10 7 8 10 8 5 8 12 5 3 4 6 1 9 1 2 3 8 3 6 9 10 9 6 10 15 8 6 9 12 4 1 4 5 7 1 3 4 6
3 5 6 5 5 7 10 10 1 2 3 3 3 5 6 8 5 1 2 3 10 6 7 8 8 7 4 5 7 1 8 10 13 8 1 3 4 5 1 2 3 4 1 3 4 10 1 3 4 3 1 2 3 2 4 5 6 10 8 7 10
12 4 4 7 10 6 4 7 8 1 1 3 4 9 3 5 6 10 5 6 7 2 1 3 4 3 1 3 4 5 1 3 4 7 1 2 3 8 6 6 8 11 10 6 8 10 8 4 5 7 4 5 6 7 5 1 2 3 1 2 4 6
2 1 2 3 3 1 2 3 8 8 5 7 8 7 1 2 3 5 4 7 9 9 1 2 3 1 3 5 6 3 1 2 5 4 5 6 8 10 1 2 3 2 2 9 10 14 3 8 10 14 1 1 4 5 7 5 1 1 3 4 10 7
10 13 6 1 3 4 4 7 8 11 8 5 9 12 1 4 4 7 8 2 4 1 3 5 3 1 3 4
14 3 6 4 5 7 7 6 7 10 8 2 4 5 9 8 1 3 4 4 4 6 7 6 9 10 12 3 2 4 5 7 5 8 11 5 1 3 6 10 1 3 4 9 7 8 11 2 5 9 12 1 2 1 3 4 4 5 5 6 8
9 1 3 4 2 1 2 3 3 3 4 6 10 10 3 4 5 1 4 5 6 3 3 4 5 7 7 10 15 9 3 4 6 6 3 4 6 4 4 7 8 5 7 8 9 2 1 4 5 8 6 8 9 10 2 1 3 4 9 5 9 11
6 1 3 4 7 1 3 4 4 1 4 6 3 8 9 10 8 8 9 13 5 6 8 10 10 6 10 14 1 9 10 15 3 10 3 4 5 1 4 6 8 8 4 7 9 8 4 8 10 15 1 7 8 10 7 1 3 4
10 7 9 12 6 5 8 11 5 8 9 12 9 6 8 11 3 5 6 8 7 5 9 10 15 1 7 8 12 8 6 9 10 9 9 10 12 6 1 3 4 4 2 4 5 3 1 2 5 10 2 7 8 9 6 1 2 3
9 5 6 7 5 1 2 3 8 7 10 11 3 1 2 3 4 5 6 7 10 6 7 9 1 8 9 13 7 5 8 9 6 10 5 8 12 3 3 5 6 1 5 6 8 6 1 4 5 8 4 5 6 2 1 2 3 5 10 1 4
6 3 1 2 5 8 1 4 7 2 4 7 9 9 1 2 5 7 10 4 7 8 6 1 3 4 8 1 2 3 2 1 3 4 3 4 5 6 5 1 4 7 7 3 5 7 4 5 1 3 4 10 6 8 11 3 4 7 8 8 1 3 4
17 8 7 8 10 12 6 9 10 14 10 1 2 3 1 5 9 11 9 4 5 6 5 3 4 6 4 1 2 3 2 1 2 3 9 4 1 3 4 5 3 5 6 10 4 7 9 6 5 9 11 7 7 10 15 1 3 4 5
8 1 2 3 9 1 2 3 2 1 3 6 9 6 1 2 5 9 5 9 11 7 2 4 5 8 1 3 4 10 1 2 4 4 4 5 6 3 8 9 10 2 1 3 4 5 4 5 7 3 8 1 2 3 2 2 4 5 10 5 6 9 7
9 1 4 7 1 7 10 11 8 5 8 11 7 4 6 7 5 8 9 11 10 2 4 6 6 7 9 10 9 7 5 7 10 10 4 5 7 5 1 3 4 4 6 7 8 8 4 5 7 3 2 4 5 1 1 3 4 6 6 10
13 9 6 10 14 10 5 7 8 11 7 3 4 5 8 1 2 5 1 5 7 9 6 7 10 13 9 1 3 4 4 2 4 5 2 5 6 9 10 2 4 5 3 1 4 5 4 9 1 3 4 7 4 6 9 10 1 2 3 2
5 8 11 3 4 5 6 8 7 1 4 7 5 6 7 10 9 2 5 6 9 8 5 8 9 6 3 4 5 1 2 4 6 7 4 5 7 5 5 9 13 4 1 3 4 3 1 3 6 10 5 6 8 10 2 5 9 13 7 1 2 3
8 1 3 4 10 7 8 11 1 1 3 6 9 1 2 3 6 6 7 8 5 2 4 6 3 3 5 6 4 1 3 4 7 6 1 3 4 3 4 7 9 7 1 3 4 8 6 7 8 1 5 7 10 10 6 9 13 4 1 3 4 2 2

6 7 8 4 1 2 3 2 3 7 8 9 10 1 3 4 6 8 9 10 12 4 1 3 4 5 5 8 9 10 2 4 6 7 1 2 3 2 6 10 15 8 3 1 3 4 4 7 8 9 5 1 2 3 9 5 8 10 8 1 3 5 1 1 4 7 6 1 4 6 7 6 9 12 6 7 2 4 5 5 1 3 4 8 4 6 7 6 6 7 10 4 1 3 5 9 1 3 4

14 3 7 8 10 11 9 7 10 13 8 3 4 6 7 9 1 4 5 2 2 4 5 4 7 10 14 7 3 4 6 1 4 5 7 6 1 2 3 5 5 7 8 9 3 1 2 3 10 6 10 14 8 4 7 10 7 5 6 7 4 9 10 11 2 2 4 5 9 5 7 9 1 4 5 7 5 4 7 10 2 2 1 3 4 4 7 8 12 6 2 3 4 6 3 7 8 9 7 2 4 5 1 7 9 10 9 3 5 6 5 1 2 3 9 7 5 6 9 2 7 8 9 8 3 4 5 9 1 2 3 6 5 8 9 3 2 4 5 10 5 6 7 4 1 3 4 5 7 8 11 3 7 1 3 4 6 1 2 3 9 5 8 10 2 8 4 7 8 2 8 10 11 5 5 1 2 3 1 1 2 3 2 1 2 3 6 5 8 12 3 1 3 4 2 9 1 2 3 1 9 10 15 2 7 5 6 8 1 5 6 9 6 7 3 5 7 4 1 4 7 9 3 4 6 2 5 7 9 3 6 8 11 5 1 2 3 7 1 9 10 13 6 1 3 4 9 6 7 8 7 8 10 15 5 9 10 13 4 6 8 10 8 4 6 7 8 6 1 3 4 1 7 8 11 8 6 7 9 9 8 10 13 2 8 10 13 5 3 4 5 3 4 7 9 7 3 5 7

20 4 7 6 9 12 6 1 3 4 10 1 2 3 9 1 3 4 4 7 6 8 11 10 2 4 5 4 6 7 10 8 7 8 10 7 2 6 9 10 9 1 2 3 6 2 4 5 1 5 7 8 8 5 6 9 5 6 9 10 7 8 9 12 8 3 8 9 12 7 3 4 6 4 1 2 3 6 5 6 7 2 8 10 11 5 4 6 9 10 1 3 4 9 1 3 4 6 8 7 9 11 2 6 10 12 5 1 2 3 7 6 8 10 6 6 9 11 4 1 2 3 5 6 2 4 6 3 1 2 3 5 7 8 12 7 5 9 13 8 1 3 4 9 4 3 5 7 7 3 5 6 3 1 3 4 2 3 5 7 6 3 4 5 9 1 3 4 5 4 6 7 8 1 2 3 10 7 9 11 4 5 1 2 3 9 7 8 9 10 7 8 12 7 4 5 6 9 10 3 4 5 5 1 3 6 8 7 10 12 2 9 10 11 6 1 2 3 1 1 2 3 9 1 3 4 7 7 8 9 4 1 3 6 5 3 5 6 8 10 1 2 3 9 1 4 7 2 7 8 9 6 4 7 9 4 9 2 4 5 10 2 4 6 3 4 5 7 2 9 10 12 2 6 5 6 9 8 6 9 13 4 1 4 7 9 9 5 7 10 8 4 5 6 7 5 8 10 2 1 1 2 3 8 1 3 4 9 3 3 4 6 9 8 9 10 4 9 10 15 8 6 10 12 7 1 2 5 10 5 6 7 6 4 7 9 5 1 4 5 2 2 4 5 5 8 8 10 11 3 5 8 10 5 1 2 3 2 1 3 4 1 4 5 6 8 7 5 6 7 9 2 4 6 3 1 4 6 5 1 2 3 6 6 10 15 2 3 4 6 10 7 8 10 4 2 4 5 1 5 2 4 6 8 8 5 6 9 9 6 7 10 4 7 8 11 10 3 4 5 1 5 6 8 2 8 10 11 5 9 10 14 7 4 5 7 6 5 1 3 4 10 9 10 13 2 5 8 11 7 1 2 3 6 8 10 14 8 1 3 4

11 1 9 6 8 10 1 9 2 4 5 5 9 5 6 8 1 5 7 10 4 3 5 7 7 4 5 6 10 5 6 7 8 2 5 6 7 8 3 5 6 4 3 5 7 6 3 4 6 7 7 10 13 10 5 6 9 1 6 7 9 3 7 8 11 1 10 5 6 7 5 7 5 8 9 5 1 3 4 4 7 9 10 9 8 10 11 6 7 9 11 10 4 5 6 8 9 6 10 11 3 6 9 13 7 2 4 5 6 7 9 12 1 3 5 7 8 5 9 13 2 6 9 13 10 1 2 3 5 1 2 3 10 8 4 7 8 7 4 5 7 10 1 3 4 9 1 3 4 1 7 10 12 5 9 10 14 4 5 6 7 3 3 4 5 2 6 7 10 6 4 6 9 8 8 7 9 10 5 4 7 10 6 7 9 12 3 5 6 9 7 1 3 4 9 1 3 4 1 2 4 6 10 9 10 13 7 7 3 5 6 6 5 6 9 5 6 8 12 4 4 6 9 10 4 5 7 1 6 10 13 8 1 3 4 3 6 1 3 6 1 1 3 4 9 1 2 3

12 2 3 5 6 8 1 1 4 7 8 4 1 2 3 7 7 8 12 8 3 5 7 6 1 2 4 1 5 6 9 9 2 4 5 10 7 8 11 3 8 9 10 8 1 1 4 6 9 5 9 12 6 3 5 7 7 1 2 3 8 4 5 7 5 1 3 4 4 6 10 12 3 4 5 7 9 8 4 5 7 9 3 5 6 5 3 4 6 3 1 2 4 1 5 6 9 10 1 2 3 4 5 9 10 7 1 3 4 2 5 8 10 8 2 1 2 3 8 1 3 4 9 6 9 10 1 1 2 3 6 3 5 7 5 6 8 9 4 5 6 9 3 5 7 10 7 8 2 4 5 10 1 3 4 2 1 2 3 7 3 5 7 9 1 3 4 4 2 4 5 6 1 3 5 10 9 4 5 6 6 3 5 6 8 6 8 11 10 6 7 9 7 6 7 9 5 1 3 4 4 1 2 3 3 4 5 6 2 7 9 13 1 6 9 10 9 1 2 4 6 9 1 4 5 6 6 9 12 8 1 3 4 7 9 10 13 3 7 8 12 4 3 4 6 10 1 3 4 2 3 4 5 3 8 1 3 4 5 1 2 3 2 1 2 3 5 5 2 4 5 2 7 8 11 10 1 2 3 9 5 8 9 7 1 3 4 4 6 1 4 6 4 7 8 10 1 8 10 14 7 1 2 3 5 9 6 8 9 4 4 6 7 10 1 2 3 7 1 2 4 5 1 3 4

12 7 2 3 4 6 9 4 6 7 4 6 8 12 7 6 10 12 6 4 6 7 3 6 10 11 5 4 5 7 4 3 5 7 8 10 6 8 10 2 1 2 3 6 1 2 3 2 4 6 8 11 1 5 6 7 2 10 1 2 3 7 1 2 3 4 1 1 3 4 4 5 7 9 2 8 9 12 7 3 5 6 5 5 8 10 15 9 6 7 8 3 6 7 9 2 3 4 6 8 1 3 4 8 8 8 9 12 9 3 5 7 5 1 2 3 7 7 9 11 10 9 10 15 4 1 2 3 2 1 2 3 3 5 6 8 10 4 7 9 10 9 1 4 7 8 3 5 7 2 5 6 9 7 2 4 6 6 8 9 13 5 7 8 10 3 6 8 12 1 1 2 5 10 5 6 9 4 6 1 4 7 9 4 6 8 3 6 10 12 7 1 3 4 7 1 1 2 3 2 5 9 13 3 7 8 11 4 1 2 3 5 9 10 13 9 1 3 4 7 1 2 3 3 4 4 6 8 9 3 5 6 6 2 4 5 7 7 6 8 10 6 9 10 11 1 1 4 7 4 5 9 12 10 2 4 5 2 6 7 9 3 6 8 11

15 6 7 1 3 4 3 7 8 11 8 1 2 3 5 4 5 7 10 1 3 4 4 7 10 12 3 5 3 4 6 7 5 7 10 1 7 8 12 10 8 1 3 4 7 6 9 11 9 4 6 7 10 1 4 5 6 9 10 12 4 2 4 6 5 1 3 4 3 4 5 7 2 6 9 13 1 6 9 12 1 2 7 10 12 6 5 5 7 8 2 1 2 3 6 4 5 6 10 4 5 7 8 6 10 12 4 6 9 11 7 4 1 4 7 5 3 5 6 8 1 3 4 3 1 3 4 2 1 3 4 1 7 8 11 10 6 9 10 6 6 1 3 4 9 1 3 4 3 8 10 11 7 4 6 8 2 2 4 6 5 8 9 13 6 1 7 10 14 8 2 4 5 3 6 9 11 2 3 5 6 6 3 4 5 9 5 6 7 7 4 1 2 3 6 2 4 6 5 1 3 4 10 7 10 13 1 8 10 15 7 1 4 5 3 4 5 6 6 4 6 8 11 10 1 3 4 8 5 6 9 5 7 8 9 6 1 2 3 2 5 8 11 3 10 5 6 7 9 1 2 3 3 1 3 4 5 2 8 10 14 9 1 4 6 4 8 10 15 10 1 2 3 7 1 2 4 6 2 4 5 6 8 1 4 7 6 7 10 11 9 2 4 6 7 4 5 6 4 1 3 4 4 8 1 3 6 1 4 5 7 6 5 8 12 10 7 8 12 6 10 2 4 5 1 5 6 7 3 7 8 12 7 3 5 7 9 1 3 4 8 4 6 9

15 7 2 6 7 8 1 1 2 3 8 6 8 10 7 1 3 4 10 4 5 6 5 1 2 3 9 8 10 11 5 5 5 6 8 10 6 8 12 8 1 3 4 2 2 4 5 4 4 5 7 4 7 5 6 9 4 1 4 5 8 4 7 9 2 7 9 13 4 6 5 6 9 5 1 2 4 7 1 3 4 1 6 10 12 4 6 1 3 4 9 2 4 6 8 1 3 4 3 8 9 12 8 3 4 7 10 1 5 6 8 8 4 6 7 4 5 6 9 10 2 4 6 6 5 6 8 2 9 10 14 7 1 3 6 2 5 6 7 10 9 7 10 12 3 5 2 4 6 2 1 2 3 1 8 10 11 5 1 7 10 14 2 5 6 9 10 1 2 3 3 4 6 7 8 1 2 3 8 10 6 8 12 2 5 8 11 5 4 7 10 7 5 6 9 4 6 10 13 9 8 9 12 3 7 9 10 1 1 2 3 5 4 5 7 8 5 3 5 6 3 1 3 4 7 1 3 4 9 2 4 6 10 3 1 3 4 5 5 6 8 8 3 4 6 9 1 2 5 4 6 8 10 1 1 3 4 10 9 10 11 6 5 9 13 2 7 8 11 7 2 4 6 3 7 2 4 5 8 5 9 11 1 4 5 6 6 9 7 8 12 6 7 8 10 8 2 4 5 3 6 10 13 5 4 7 10 10 7 8 9 6 5 5 6 7 9 1 4 5 10 5 6 9 3 1 3 4 6 3 4 5 8 5 7 10

18 3 1 1 3 5 5 6 7 9 8 6 8 12 2 7 1 2 4 8 1 2 3 10 10 5 6 8 4 9 10 14 1 6 10 12 6 6 10 13 2 7 9 13 5 1 3 4 7 5 8 11 8 7 9 13 3 7 9 10 9 5 6 8 5 6 1 2 3 9 3 4 5 8 8 10 12 1 4 6 8 5 1 3 4 2 9 6 8 12 8 7 10 15 4 3 6 7 9 7 1 3 4 5 5 6 7 1 2 4 5 3 8 3 4 6 9 3 4 6 10 3 5 6 7 4 7 10 11 8 5 6 8 9 4 5 6 7 5 6 9 6 3 4 5 5 5 9 11 2 5 6 9 3 5 6 10 13 10 1 3 4 8 1 3 5 9 7 1 3 4 4 6 9 10 5 3 4 5 6 5 8 11 9 3 4 5 2 5 7 8 10 3 5 7 3 7 10 13 1 4 5 7 1 4 1 2 3 7 6 8 10 11 7 9 10 15 9 4 5 7 3 3 5 6 10 1 3 4 5 7 8 10 2 6 7 8 9 10 4 7 10 5 1 4 7 3 1 2 3 9 2 4 5 6 1 2 3 2 1 3 4 4 1 2 3 8 4 6 8 1 8 10 15 8 4 1 2 3 6 1 2 3 10 1 2 3 7 3 4 6 8 3 5 7 3 3 4 5 9 1 2 3 5 7 10 11 9 4 1 2 3 8 1 2 5 9 7 10 11 5 5 8 10 3 6 9 13 7 1 3 5 6 5 7 8 10 7 8 11 2 9 10 13 9 5 1 2 3 8 1 3 4 10 4 7 10 2 9 10 12 6 2 4 6 3 1 2 4 4 1 2 3 7 7 10 14 9 1 3 4 4 9 1 2 3 7 6 10 15 1 4 5 6 4 1 2 5 5 4 5 7 10 10 1 2 3 8 1 2 3 9 1 2 3 5 3 5 7

11 3 3 3 5 7 9 1 3 6 1 1 3 4 8 2 5 9 12 1 4 7 10 8 5 9 13 4 1 2 3 6 7 8 9 5 5 9 13 10 7 10 13 3 7 8 11 3 4 1 2 4 3 2 4 6 8 1 2 3 1 5 2 4 5 7 1 6 9 13 10 5 7 9 6 3 4 5 2 7 10 15 4 4 5 7 5 1 2 4 7 9 10 13 7 3 9 10 14 7 6 9 11 1 4 5 7 2 1 2 3 4 9 10 12 8 5 6 8 5 1 2 4 1 4 4 5 6 8 8 5 9 11 1 2 4 6 6 7 9 11 5 5 8 9 7 8 10 12 9 7 9 12 10 1 2 4 3 1 3 4 7 8 9 10 14 6 6 9 11 10 3 5 7 1 2 4 6 4 4 5 6 7 3 4 6 5 6 7 9 1 8 6 10 14 7 5 5 7 10 1 5 6 8 6 1 3 4 9 4 7 8 10 1 2 3 4 1 4 5 7 5 9 11

20 6 9 1 3 4 1 2 4 5 4 7 10 13 8 1 2 3 2 1 2 3 10 5 9 13 5 2 1 4 7 9 1 2 3 8 6 7 9 5 7 8 9 10 4 6 9 1 2 3 5 6 1 1 1 2 3 4 4 1 3 4 5 1 3 4 1 7 8 11 2 4 6 9 4 6 1 3 4 9 1 2 4 1 5 9 11 4 4 5 7 2 8 5 9 11 2 3 4 5 10 2 8 9 10 9 7 10 11 1 5 6 9 4 1 2 3 10 1 3 5 5 1 4 7 3 1 2 3 6 6 9 11 8 9 10 11 7 1 2 3 10 5 1 3 6 1 4 7 8 8 4 5 7 9 7 9 10 4 6 7 9 2 8 9 12 3 4 5 6 7 4 7 8 10 5 6 8 6 5 6 9 6 7 1 2 3 10 1 3 4 2 1 3 4 9 1 3 5 6 6 8 12 4 1 3 4 10 7 5 7 10 9 5 7 8 8 6 10 12 5 5 9 10 6 4 7 8 10 9 10 11 3 7 10 14 1 6 7 9 4 1 3 4 2 1 2 3 6 8 5 8 10 9 1 3 4 4 2 4 6 2 1 2 4 10 7 8 9 7 7 8 12 10 3 1 3 4 9 5 9 12 5 5 6 8 4 1 2 3 2 4 6 7 7 1 4 7 1 1 3 4 8 8 9 10 10 3 5 7 6 2 4 5 1 5 1 2 3 6 5 6 10 12 10 3 5 6 9 5 9 13 2 2 4 5 3 2 4 5 4 7 10 11 7 9 2 4 5 3 5 7 8 4 7 10 14 5 5 6 7 6 5 6 9 7 8 9 12 8 5 6 7 8 8 8 9 13 2 2 4 6 4 1 2 3 1 5 6 7 7 4 7 10 5 1 3 4 3 1 3 4 9 4 6 8 5 6 1 2 3 2 7 8 11 8 4 7 8 10 5 7 10 3 4 5 6 7 8 1 3 6 2 4 6 8 5 5 6 7 7 9 10 13 6 1 4 5 3 7 10 11 10 3 4 5 7 10 9 10 14 4 3 5 7 7 2 4 5 1 8 9 10 6 4 6 8 3 4 5 7 5 1 3 4

20 5 3 3 5 6 8 1 3 4 6 1 4 7 4 3 4 5 10 7 8 11 4 9 3 4 5 4 1 4 7 1 5 6 9 3 1 2 3 6 8 5 6 9 3 1 4 7 6 5 8 9 4 2 4 6 1 5 6 8 5 1 3 4
1 5 1 4 6 5 7 6 10 14 6 6 7 9 5 3 4 6 9 1 3 4 10 3 5 6 5 9 2 4 6 6 7 8 11 2 3 4 5 4 1 2 3 10 1 2 3 9 3 7 8 9 9 1 3 5 4 7 8 12 6 7
8 11 2 6 7 10 5 7 8 10 1 5 6 8 10 5 8 12 8 5 9 13 8 2 7 10 13 5 6 8 10 8 5 6 8 10 7 8 9 6 1 3 4 1 7 10 13 7 5 7 9 4 3 4 6 9 2 3
4 6 8 5 6 7 9 7 8 12 4 2 4 5 10 1 4 6 5 5 6 7 1 1 2 3 3 1 4 5 6 4 5 7 5 10 6 8 9 3 2 4 5 8 6 10 13 5 2 4 6 6 5 6 8 10 6 7 8 9 10
2 4 5 2 4 5 6 1 4 6 9 3 2 4 5 5 1 4 7 4 1 2 4 9 2 4 5 8 6 10 14 7 6 8 11 3 10 6 7 8 2 4 7 10 5 5 6 8 6 9 3 5 7 8 9 10 15 4 1 3 4
10 1 2 3 6 5 6 8 3 1 3 4 8 5 4 5 6 9 4 5 7 3 2 4 6 8 7 8 12 7 4 7 10 2 5 8 10 4 4 6 7 10 3 5 7 9 10 2 4 5 5 1 2 3 2 6 8 12 7 6 7
8 9 2 4 5 3 5 6 7 8 4 5 7 4 5 9 13 1 1 3 4 9 7 7 9 12 10 1 3 4 5 1 2 3 1 1 4 6 4 1 3 4 3 1 2 3 2 7 9 11 8 1 3 4 9 6 8 11 1 2 1 2 5
9 2 7 8 9 9 4 6 7 4 3 4 5 7 7 10 14 10 4 5 6 6 5 6 9 3 5 7 8 5 1 4 5 1 9 10 14 7 10 9 10 13 5 5 7 8 1 5 6 9 9 6 8 9 4 1 3 4 8 1 2
3 6 1 2 3 3 5 1 3 4 9 3 4 5 1 5 6 8
15 8 2 4 5 6 1 9 10 11 9 5 6 7 8 9 10 13 3 1 2 3 5 2 4 6 10 6 7 9 4 5 9 11 1 9 1 3 6 2 7 6 10 15 4 3 5 7 2 7 2 4 5 1 1 2 3 9 6 4
5 7 3 4 6 8 10 6 8 12 9 2 4 6 8 5 6 8 7 1 3 4 4 8 10 13 5 1 2 3 1 3 5 6 8 8 1 2 4 9 1 3 4 5 1 4 7 2 1 2 3 6 1 2 3 7 5 7 10 4 4 5 7
3 6 7 9 7 7 5 7 8 9 1 3 4 8 4 6 9 3 4 5 7 5 6 8 10 6 5 6 9 10 5 9 11 2 4 6 7 9 2 5 9 10 5 2 6 10 11 3 9 10 14 1 9 10 15 10 4 6 8
9 1 2 3 8 8 5 8 11 4 8 10 11 6 1 4 6 7 1 3 4 9 9 10 11 5 3 5 7 10 5 9 12 3 8 10 11 6 5 4 7 8 10 4 6 8 6 9 10 14 7 4 7 10 9 7 8
10 2 4 7 8 4 9 1 2 5 10 7 8 11 5 7 8 11 6 7 9 10 7 5 7 9 11 9 2 4 6 10 5 9 10 1 3 4 5 3 9 10 12 8 9 10 11 4 4 5 7 2 10 2 4 5 7
2 4 6 5 4 8 9 13 2 5 6 8 10 1 3 4 1 9 10 11 8 7 8 9
18 2 4 1 2 3 10 5 6 9 10 5 4 5 6 9 4 7 8 8 3 5 7 6 5 9 12 3 5 8 10 10 6 10 12 1 5 7 10 2 3 5 7 4 1 3 4 7 1 3 5 2 8 3 5 6 4 6 9 11
5 2 5 6 9 3 1 3 4 10 8 9 10 9 1 2 3 6 5 6 7 8 10 3 4 5 4 3 4 6 8 7 8 12 7 6 8 10 3 1 2 4 2 8 9 13 6 8 9 13 9 1 3 4 5 1 5 6 8 8 1
2 3 3 1 2 3 4 7 8 10 6 6 10 14 5 2 7 8 11 9 5 7 9 3 3 4 5 10 1 3 4 4 1 2 3 1 3 7 8 9 8 2 7 9 10 7 1 2 3 4 1 3 4 10 5 6 9 8 1 3 4
6 8 9 12 3 9 10 14 9 6 8 11 2 6 5 6 8 2 1 3 4 3 2 9 10 12 9 7 8 10 5 6 10 13 8 7 5 9 10 4 5 6 9 10 1 2 3 9 5 6 8 6 5 7 8 5 2 4 6
1 2 4 6 3 1 3 5 4 7 1 3 4 1 1 2 3 3 4 7 10 8 5 6 8 4 2 1 2 3 4 9 10 13 1 9 10 11 7 7 10 13 9 4 5 7 9 2 4 6 7 7 4 5 6 9 7 9 11 6 7
10 15 3 6 7 8 5 1 3 5 10 6 8 9 8 6 9 10 2 4 5 6 7 1 1 3 5 9 7 2 4 6 9 3 5 6 6 5 9 10 1 2 4 6 10 8 9 13 5 6 7 10 4 7 8 12 3 5 8 11
2 9 10 11 8 2 4 6 8 4 4 5 7 8 3 4 6 7 4 5 6 1 1 3 4 3 4 6 9 5 5 7 10 9 5 7 8
10 10 5 1 4 5 8 6 8 12 4 5 6 9 3 1 3 6 6 1 2 3 2 7 10 13 7 5 9 13 10 1 3 4 1 9 10 12 9 1 3 5 2 3 1 3 4 5 5 6 7 6 4 5 6 9 9 1 2 3
2 1 2 3 1 1 3 5 10 6 10 14 5 7 8 11 1 10 1 3 4 5 9 1 4 6 4 1 3 4 6 7 10 15 1 8 9 13 7 5 7 10 1 5 3 4 5 1 8 2 4 6 8 4 6 7 10 10 7
8 9 8 6 7 9 1 3 5 6 6 1 2 4 5 8 9 12 7 1 3 4 2 7 10 13 2 10 2 4 6 9 5 6 7 3 6 1 2 3 8 4 5 6 1 7 8 12
17 3 5 4 5 6 9 3 4 6 4 1 2 3 10 9 6 10 14 2 4 5 6 5 5 6 9 10 3 4 6 4 5 6 7 7 5 9 12 8 1 4 7 6 6 8 10 3 4 5 7 1 1 2 3 9 3 7 8 9 5
1 3 4 6 5 6 7 8 1 2 3 4 4 7 8 10 1 2 4 1 1 2 3 7 7 9 11 2 2 4 5 6 6 4 6 9 5 1 2 3 8 5 6 7 7 5 6 8 10 1 2 3 2 1 3 4 6 4 4 6 9 5 3 4
6 8 4 5 6 6 7 9 12 1 1 3 4 10 4 5 7 7 3 6 10 15 7 1 2 3 6 1 2 3 2 6 9 10 9 4 5 6 4 6 9 13 1 3 5 7 6 4 5 9 13 9 6 9 12 3 1 2 3 7 1
2 3 5 7 9 11 6 5 6 8 5 5 6 7 9 9 1 3 4 7 3 4 6 2 2 4 6 8 2 4 6 4 8 3 4 5 3 1 2 3 2 1 3 4 5 6 7 8 4 1 6 9 13 10 1 3 4 3 2 4 6 8 4 6
9 5 8 1 3 5 5 3 4 5 9 7 9 11 1 6 8 10 7 1 3 4 2 9 1 4 6 4 7 10 13 9 10 8 9 13 9 1 2 4 3 9 10 14 7 5 7 10 6 3 5 7 8 1 2 3 5 4 6 7
4 5 7 8 2 1 3 4 10 9 5 6 9 8 8 10 14 5 3 4 6 4 1 3 5 2 1 2 3 3 6 8 12 7 6 7 9 10 5 8 11 6 2 4 5 1 7 8 12 1 2 4 5 6 8 9 3 4 6 1 1
4 6 3 1 4 5 8 6 8 10 6 6 10 13 4 1 3 4 2 1 2 3 7 4 7 10 1 5 6 9 13

Instance 8:

20 15 2
20 1 11 3 4 5 12 10 5 9 11 5 6 8 10 8 3 5 6 13 1 2 4 1 4 6 8 3 6 8 10 15 1 2 3 4 5 6 7 7 3 4 5 6 9 10 11 14 1 2 3 11 5 9 13 6 7
1 2 3 10 1 3 5 8 1 3 4 3 7 8 12 12 1 3 4 5 5 6 7 12 3 7 9 10 2 8 9 13 1 5 6 7 11 7 8 10 15 4 7 9 12 6 8 12 9 1 2 3 14 1 3 5 7 4
7 8 13 3 4 5 5 2 4 6 6 4 5 7 3 8 7 8 12 12 3 5 6 1 1 2 3 11 11 1 4 5 2 3 5 6 14 1 2 3 1 2 4 5 6 2 4 5 8 4 7 9 15 3 5 7 10 1 2 3 9
7 8 11 12 8 9 11 5 5 7 9 4 15 6 7 9 3 2 4 6 4 6 9 11 12 1 3 4 14 1 1 2 4 7 3 4 5 5 1 2 3 12 5 6 7 11 1 2 3 10 6 7 8 3 5 6 7 9 4
5 6 14 6 7 9 8 6 7 9 2 1 3 4 4 8 9 12 6 4 5 6 13 1 3 4 15 3 7 10 13 10 5 8 9 2 1 3 4 12 1 3 4 15 4 6 7 7 1 3 4 8 1 3 4 11 1 3 4
5 4 7 9 14 1 2 3 6 1 3 4 4 5 7 9 9 7 9 11 1 4 5 7 13 1 3 4 3 1 2 4 5 5 1 2 4 14 3 5 7 9 7 4 7 8 12 3 4 5 13 3 5 7 2 3 4 5 11 7 8
9 10 2 4 5 3 8 10 14 4 4 7 8 15 3 5 6 2 10 5 9 12 6 6 8 10 14 13 1 3 4 8 1 2 5 7 8 9 13 1 4 7 9 11 5 7 8 15 1 3 4 4 5 8 9 2 1 2
4 6 4 5 7 14 3 5 7 5 2 4 5 10 1 2 3 3 3 4 6 12 7 10 13 3 5 9 10 11 13 6 7 8 4 9 10 13 6 12 1 2 3 11 1 3 4 7 6 8 12 4 1 2 3 14 8
9 12 13 4 7 9 12 2 4 7 8 12 5 8 9 6 3 4 6 13 4 5 6 9 1 4 6 11 2 4 6 15 5 8 11 5 5 8 12 8 1 3 4 3 3 4 5 1 7 9 12 14 1 4 7 12 12
7 9 12 13 1 2 3 15 8 9 11 14 5 8 9 11 1 2 3 3 6 7 9 9 3 4 5 8 4 6 7 6 1 3 4 10 5 6 9 5 1 2 3 4 5 6 9 9 8 4 5 6 9 9 10 12 7 4 7 9
5 9 10 15 11 1 2 3 13 5 6 7 1 8 10 12 2 4 5 6 3 7 8 11 2 3 8 9 13 8 4 5 6 2 8 3 4 5 7 1 3 6
17 6 4 4 7 10 5 9 10 14 15 6 10 14 7 5 6 7 8 1 3 4 2 1 2 3 9 11 1 3 6 13 8 9 11 15 7 8 12 3 1 2 3 10 9 10 12 5 7 8 12 9 7 8 11
4 4 5 6 6 7 10 15 2 9 3 5 6 7 5 6 8 5 9 3 5 7 7 2 4 6 15 1 2 3 1 4 6 7 11 7 9 11 10 1 4 6 7 14 1 2 3 11 1 2 3 12 5 9 13 15 7 9
13 3 5 8 9 9 3 4 6 13 1 3 4 7 1 4 6 4 6 9 11 14 2 4 5 6 10 5 6 8 13 4 5 6 8 6 7 8 5 6 10 11 15 2 4 5 7 5 7 10 14 3 5 6 11 1 3 5
1 1 2 3 12 1 2 3 4 7 8 10 3 7 8 11 6 5 7 8 13 11 2 4 5 8 7 8 10 12 7 8 11 3 4 6 8 10 8 9 10 9 5 8 9 7 9 10 13 14 4 7 8 4 5 7 9
6 2 4 5 15 1 3 4 5 4 5 7 13 4 6 7 4 8 5 7 8 12 1 2 3 15 1 2 3 3 4 5 7 5 8 4 5 7 6 4 5 6 3 6 10 14 12 7 8 10 9 5 9 13 14 1 1 3 6
8 1 2 3 12 4 5 6 9 7 8 9 11 5 9 10 13 2 4 5 3 7 8 11 5 1 3 5 15 3 4 6 10 1 3 4 6 1 3 4 4 6 9 13 7 1 3 4 2 5 6 7 13 3 6 10 11 6 4
5 6 4 5 6 7 12 4 7 8 11 5 7 9 7 5 8 11 9 1 2 5 8 1 3 4 15 2 4 5 13 1 3 6 2 4 5 7 14 6 7 9 5 1 3 4 5 3 1 3 4 10 9 10 13 8 6 8 10
7 3 4 5 6 5 7 9 13 2 9 10 12 15 5 9 12 3 1 4 5 10 1 2 3 5 5 7 9 7 1 2 3 14 7 10 14 6 5 6 8 8 1 3 4 4 6 10 13 13 1 3 5 12 1 3 4
9 1 3 4 8 13 1 2 3 7 5 6 9 9 8 10 14 6 5 9 11 12 1 3 4 3 5 6 8 1 1 4 6 5 8 9 10 13 11 1 2 3 2 1 4 5 7 4 6 8 12 5 6 9 9 4 6 9 13
4 7 9 3 1 2 3 10 4 5 6 14 7 8 10 5 5 6 8 4 1 3 4 15 4 7 10 6 1 3 4 15 11 1 2 5 5 1 3 5 15 4 5 6 4 8 9 13 14 4 5 6 6 1 2 3 9 3 5
7 10 1 3 4 8 2 4 6 13 5 6 7 1 5 6 9 7 2 4 5 3 1 3 4 2 2 4 5 12 1 3 4 13 2 6 9 10 14 7 10 12 12 1 3 4 15 5 7 10 10 3 5 7 11 2 4
5 1 5 6 8 8 1 3 4 13 4 5 6 6 1 4 7 5 5 9 10 4 1 3 4 9 6 7 10
15 6 5 7 9 10 14 4 6 9 10 1 2 3 6 4 7 10 11 1 4 7 12 3 4 6 3 11 7 8 9 10 7 10 11 13 3 5 7 5 12 5 6 7 2 6 9 12 7 1 2 3 10 8 10
12 11 1 3 4 7 9 1 3 4 6 6 8 11 14 5 9 11 7 4 6 8 3 1 3 4 1 5 7 10 15 4 7 8 3 10 6 10 14 11 1 2 3 4 3 5 6 14 8 1 3 6 14 7 8 9 15
1 3 4 9 4 7 9 6 1 2 4 11 7 8 11 5 1 3 4 12 3 5 7 7 1 3 4 2 5 8 10 10 3 4 5 4 3 4 5 13 3 4 5 3 3 4 5 10 6 6 7 10 8 3 5 7 11 1 2 3

12 3 5 6 10 1 4 7 2 1 4 7 1 5 8 11 13 4 6 7 9 3 4 6 5 1 3 4 15 15 8 10 12 10 1 3 6 12 7 8 10 9 2 4 6 13 2 4 6 3 1 3 4 5 1 3 6 8
5 6 7 6 6 10 13 2 7 9 11 11 1 2 3 14 7 8 9 4 5 9 11 1 4 5 6 7 7 10 13 5 9 4 5 6 3 1 2 3 13 1 3 4 1 9 10 13 8 3 4 5 11 9 6 8 10
14 4 6 9 3 1 3 5 12 8 10 14 8 7 8 12 2 4 5 6 6 8 10 11 15 8 9 10 10 5 6 9 1 6 9 12 5 1 2 3 14 9 6 7 9 4 5 9 13 6 2 4 6 2 1 2 3
7 3 4 5 8 3 5 7 14 1 4 7 13 1 3 5 5 7 8 11 11 2 4 6 3 1 3 4 10 6 7 8 15 3 5 7 1 3 5 6 5 3 1 3 4 14 1 3 4 7 2 4 6 5 7 10 15 10 1
3 4 8 8 5 9 12 13 4 6 7 12 6 7 8 4 5 8 11 11 9 10 13 9 4 6 8 10 6 8 10 15 4 5 7 1 8 8 9 13 10 2 5 9 10 12 5 6 9 3 1 3 4 6 3 5 7
5 4 5 7 11 7 8 9 9 1 2 3 15 5 9 11 7 5 7 9 10 6 9 12
17 14 15 5 8 9 13 3 4 5 8 5 8 11 7 1 4 5 1 4 7 8 10 1 3 4 9 1 3 4 11 5 6 7 4 3 4 5 12 5 7 9 5 4 5 7 14 5 6 7 6 3 5 6 3 6 7 10 8
3 9 10 15 13 5 6 9 5 3 4 6 10 2 4 6 14 8 9 10 6 1 2 3 11 4 7 10 15 7 8 10 2 1 1 2 3 7 3 5 6 15 9 6 10 13 6 1 2 3 3 7 8 10 15 6
8 10 14 1 2 3 2 6 8 12 7 2 4 5 12 5 9 10 4 1 2 3 13 9 10 12 8 5 6 8 11 6 10 11 5 3 4 5 10 8 9 12 1 6 8 10 12 7 4 7 8 10 6 10
11 5 6 7 8 12 1 3 4 11 3 5 6 6 4 5 7 9 5 6 9 8 1 3 4 13 1 2 3 15 1 2 3 3 6 9 11 4 7 9 11 5 13 5 6 7 8 3 5 7 4 3 5 6 15 8 10 12
11 9 10 15 8 10 4 5 7 4 6 8 11 15 1 2 3 12 3 5 7 11 1 3 6 6 7 9 11 1 1 2 3 9 1 2 3 2 5 4 6 9 8 5 7 9 7 7 3 4 5 4 1 2 3 1 9 10 13
13 6 8 10 11 7 9 12 8 6 8 12 9 1 2 3 7 1 1 3 4 10 1 3 4 13 1 3 4 7 6 9 10 5 6 8 11 2 7 8 10 14 1 2 3 6 3 6 10 13 9 6 10 12 10
1 3 4 1 6 10 11 2 7 8 11 7 5 6 9 2 15 1 2 3 10 6 9 10 5 3 3 4 5 14 7 8 11 5 2 4 6 6 7 8 10 13 1 2 3 5 9 8 9 12 15 7 8 11 4 7 10
15 7 1 3 5 11 7 9 11 15 12 1 3 4 1 4 7 10 14 8 9 12 7 1 4 6 13 2 4 6 2 5 6 8 15 6 10 11 3 7 10 12 9 1 2 3 11 1 3 6 5 8 10 12
10 5 9 11 4 4 5 7 6 1 3 4 8 2 4 6 1 2 6 7 10 15 1 7 9 12 7 1 3 4 5 1 2 5 12 7 8 11 4 5 7 8 15 9 10 13 6 4 5 7 13 1 3 4 9 3 4 6
11 4 5 6 8 5 6 8 14 6 8 11 3 9 10 15 2 5 6 9 10 5 6 8
15 3 7 1 2 3 14 1 2 3 11 3 4 6 2 15 1 2 4 14 7 9 10 6 15 8 9 11 14 9 10 12 7 9 10 15 9 9 10 13 10 7 8 9 5 4 5 6 4 7 5 6 8 14 5
8 12 13 4 5 7 6 1 3 4 12 1 5 6 9 5 4 7 8 8 1 2 3 10 6 10 14 14 4 5 6 9 5 6 7 13 3 5 7 11 8 9 10 12 6 8 12 7 1 2 3 3 1 2 3 6 4 6
9 5 8 1 3 4 4 7 8 11 7 1 2 3 1 7 8 11 11 4 6 7 12 10 5 6 8 12 5 6 9 13 5 6 7 14 1 3 6 11 1 2 3 1 5 7 8 8 2 4 6 4 1 4 6 7 5 6 9 6
7 8 10 5 5 7 9 2 1 3 4 10 1 1 3 4 15 5 6 7 13 3 4 6 10 1 4 7 7 8 9 12 8 6 7 10 6 5 6 8 4 8 10 11 12 6 8 11 3 4 5 7 9 12 1 3 4 14
1 4 6 15 2 4 5 2 4 6 7 11 5 6 7 4 3 5 7 1 5 7 9 9 7 9 10 7 6 7 9 7 8 6 8 12 5 7 10 11 13 1 3 4 11 3 4 5 2 7 8 9 15 2 4 5 9 2 4 6
11 7 9 10 12 14 5 8 12 5 2 4 5 9 7 8 11 15 1 2 3 11 1 3 4 6 7 10 13 12 5 8 10 3 1 2 3 1 1 2 3 10 7 9 11 4 5 1 4 6 14 1 4 7 10
4 5 7 12 7 8 9 7 10 2 4 5 14 7 8 10 5 2 4 6 12 5 6 9 13 5 6 9 15 2 4 6 9 3 5 6 5 3 1 3 4 15 3 4 5 1 3 5 7 12 6 7 9 7 5 6 7 3 15
1 3 4 14 6 10 11 12 1 2 3
20 1 13 9 10 15 3 9 7 8 9 6 6 10 13 13 1 2 3 7 15 3 5 7 9 3 5 7 7 1 4 5 8 1 3 4 12 4 7 10 2 7 9 10 1 7 9 11 3 2 6 9 13 3 5 8 11
7 9 10 13 9 5 1 3 5 14 2 4 5 8 3 5 6 10 2 4 5 11 2 4 5 15 1 3 5 7 1 3 4 3 5 6 7 12 3 4 6 8 2 1 3 4 10 1 3 4 13 5 7 10 9 3 5 6 8
1 3 4 12 7 8 9 7 6 7 9 1 2 4 5 9 5 5 8 12 14 8 9 11 4 4 6 7 12 6 7 9 11 1 2 3 10 5 7 10 6 6 9 11 7 3 5 6 3 6 9 11 8 7 4 6 8 9 7
8 9 5 2 4 5 2 5 9 10 1 7 8 12 12 1 3 4 15 6 7 9 8 6 9 12 13 11 3 4 5 2 1 3 4 13 5 7 8 7 1 3 6 10 1 3 4 8 4 5 6 3 5 6 9 1 2 4 5 4
5 6 9 14 4 7 8 12 1 2 3 15 6 7 9 5 5 6 8 1 12 8 10 12 14 3 9 10 11 4 1 3 4 15 4 5 6 13 6 7 9 6 5 6 7 7 1 4 5 11 6 8 9 8 3 5 7 10
4 7 8 12 7 8 12 5 2 4 6 14 4 5 6 1 1 2 4 2 6 8 11 8 12 5 6 9 9 1 3 4 2 7 10 15 3 7 10 14 10 7 8 12 13 8 9 11 5 4 7 10 14 3 5 6
10 6 7 8 11 10 6 10 13 15 1 4 7 4 7 9 10 11 3 4 5 1 4 5 7 9 7 8 9 14 5 7 8 7 8 10 11 8 6 10 14 15 9 1 2 5 14 3 5 6 7 4 7 9 12
6 10 12 13 1 3 4 10 4 5 7 6 6 10 12 8 1 2 3 15 7 8 11 2 1 3 6 4 7 8 12 3 6 9 11 5 3 4 6 11 5 9 11 1 1 3 4 14 10 3 5 7 5 7 8 9
15 4 6 7 7 1 2 4 12 1 3 4 2 6 7 9 9 4 7 10 8 1 2 4 1 5 8 11 11 1 2 3 13 4 6 8 3 4 6 8 6 1 2 3 14 9 10 14 10 9 8 10 12 14 8 10
12 13 1 3 4 11 5 6 8 8 6 10 12 6 4 5 6 5 5 6 7 15 6 7 10 7 7 10 11 3 6 7 10 1 12 5 6 7 3 4 4 5 7 10 3 4 6 13 4 6 7 11 4 1 2 3
10 5 9 10 3 1 4 7 12 1 2 4 15 1 3 4 14 5 6 8 9 1 3 4 2 5 8 12 13 3 5 6 6 3 5 6 7 1 3 4 12 5 8 9 11 10 1 2 3 13 3 4 5 6 1 4 5 14
1 2 3 15 1 3 4 3 1 3 4 2 7 8 12 12 4 7 10 7 2 4 5 9 3 5 6 8 1 4 7
17 6 1 5 8 11 4 7 9 12 13 1 2 3 12 1 2 3 9 7 10 12 5 7 8 12 2 13 4 7 8 10 7 8 11 8 14 4 6 7 12 1 3 4 13 1 2 3 4 6 8 12 9 1 2 3
10 1 3 4 5 6 7 9 2 5 7 10 2 15 4 5 7 14 1 2 3 14 15 9 10 13 14 1 3 4 13 1 3 4 2 1 4 5 3 6 9 12 4 4 5 7 6 1 2 3 12 5 6 7 7 1 3 4
1 3 5 6 9 1 2 3 8 5 6 7 5 7 9 10 11 5 6 9 5 15 6 9 10 3 3 5 7 4 4 5 7 12 5 9 13 10 8 9 11 13 3 3 5 7 14 8 9 13 1 1 3 4 12 1 2 3
2 4 5 7 11 6 7 10 13 5 6 9 8 2 4 5 7 1 3 4 9 5 6 8 6 9 10 12 4 2 4 6 5 2 4 6 13 9 6 7 8 14 1 2 3 3 1 2 3 11 1 3 4 6 9 10 15 15 3
5 6 12 6 7 10 10 5 8 11 2 5 6 7 13 5 7 10 5 5 8 9 4 1 2 3 1 6 7 9 3 5 1 3 4 10 5 7 9 6 1 2 4 14 8 1 3 4 2 7 8 12 6 5 8 12 4 1 3
4 11 1 4 6 10 2 4 5 14 4 5 7 9 5 6 7 1 3 5 7 15 1 3 4 5 2 4 5 7 9 10 11 3 7 9 13 13 9 10 15 1 15 6 8 12 9 3 1 3 4 15 6 7 8 1 2
4 6 9 4 7 8 7 5 6 7 13 1 3 4 8 5 9 11 5 4 6 7 14 1 2 3 11 14 4 5 7 15 2 4 5 13 7 8 11 9 7 8 10 7 2 4 6 10 5 9 12 11 8 10 13 6 2
4 6 8 6 10 13 4 7 8 10 12 1 2 3 2 5 1 3 4 15 4 5 6 11 5 1 2 3 10 3 4 5 12 1 3 4 9 4 7 10 13 9 10 15 8 4 6 7 14 5 7 10 4 4 7 10
7 6 8 11 2 4 5 7 1 8 9 10 7 1 5 6 9 8 7 10 12 4 5 9 13 15 9 10 14 11 1 2 4 3 3 5 6 12 5 6 9 14 6 5 8 9 14 7 8 11 8 1 2 3 9 9 10
14 5 5 8 11 10 4 7 10 4 9 10 11 11 1 3 4 12 4 7 9 2 4 7 9 1 2 4 6 7 1 3 4 15 1 2 3 13 8 10 14
18 11 1 5 6 9 13 3 5 7 7 1 4 5 12 2 4 6 11 3 4 6 9 2 4 6 15 1 4 5 6 1 3 4 14 1 3 4 8 3 5 7 5 1 2 3 1 5 1 4 5 12 2 1 4 5 8 5 6 9
15 4 5 6 1 1 2 3 11 2 4 6 10 1 2 5 4 3 4 6 7 9 10 11 12 5 7 9 14 2 4 6 9 1 2 3 13 4 5 6 7 11 1 2 3 1 7 8 11 10 1 2 3 15 3 4 6 5
6 7 9 7 4 5 7 9 4 5 7 10 1 1 2 5 4 1 2 3 14 8 9 10 15 1 3 5 11 7 9 13 3 7 9 13 6 4 5 7 13 3 5 6 8 4 7 9 7 5 6 9 12 7 6 7 9 14 1
2 4 11 5 6 8 9 1 3 4 1 1 2 4 15 7 10 14 13 4 5 7 8 1 4 7 12 1 3 4 5 1 4 7 10 4 5 6 4 5 6 7 7 11 7 8 12 13 6 8 12 2 7 9 10 7 2 4
6 1 1 2 3 6 1 2 5 9 7 8 9 6 3 1 2 5 13 6 7 10 1 1 3 4 2 1 2 5 5 7 8 12 4 1 3 4 9 13 9 10 11 10 4 5 6 14 4 5 6 12 5 8 11 11 5 7 9
5 1 3 4 9 1 3 4 3 4 6 9 1 6 9 11 15 15 4 7 8 14 4 7 10 5 5 8 11 8 7 10 15 9 5 6 7 2 6 8 11 3 4 6 8 6 5 6 7 10 1 3 4 4 5 8 11 7 5
7 8 13 2 4 5 12 1 3 4 11 5 6 8 1 8 9 12 1 2 9 10 13 11 6 5 9 13 8 5 8 11 14 3 5 7 12 5 6 7 9 1 3 4 7 6 10 13 11 6 9 12 3 4 5 6
1 4 7 10 13 7 8 12 10 1 2 5 14 10 3 5 7 14 4 6 8 5 1 3 4 9 1 2 3 1 3 5 6 11 5 6 7 13 1 2 3 3 3 4 6 6 5 8 10 15 3 4 6 4 4 5 6 8 6
7 8 12 1 2 3 2 4 5 7 15 14 6 8 11 2 3 5 6 1 1 2 3 8 1 2 3 5 4 7 9 4 1 2 3 6 5 9 13 7 3 4 6 11 1 2 3 13 4 5 6 15 9 10 13 12 7 8
12 3 5 9 10 9 1 4 6 10 1 3 4 13 2 1 2 3 15 1 2 3 8 7 10 13 3 2 4 5 5 6 7 8 6 8 9 11 7 6 7 8 13 9 10 11 11 5 9 13 14 3 5 6 12 8
10 11 4 4 7 8 9 1 3 4 1 6 1 3 4 14 14 7 10 15 15 6 7 10 1 1 3 4 2 9 10 13 10 6 8 11 6 1 4 7 12 1 3 4 9 2 4 5 4 6 8 9 13 5 9 10
11 2 4 5 7 1 2 4 3 6 7 8 5 1 3 4 15 11 1 2 3 1 1 3 4 10 6 8 10 5 1 4 5 12 1 2 4 14 1 3 4 4 1 3 4 8 5 7 10 7 1 2 3 13 2 4 6 3 5 6
7 2 1 2 3 9 3 4 5 6 7 8 9 15 3 4 5
19 7 1 4 5 6 14 8 9 10 11 1 3 4 4 5 9 10 7 3 5 6 5 5 8 10 9 3 4 5 6 11 3 5 6 12 1 3 4 7 1 2 4 14 7 8 12 9 2 4 6 5 2 4 6 6 8 4 7
8 5 6 9 10 2 8 9 10 6 4 7 10 12 1 2 3 13 7 10 15 15 2 6 9 11 11 1 4 6 4 5 6 8 14 1 3 4 9 1 3 4 8 1 3 4 5 4 6 7 12 1 2 3 7 5 7 10
13 1 3 4 10 8 9 10 3 7 10 11 6 7 10 13 1 5 6 8 15 3 4 6 13 6 6 7 10 7 3 5 6 12 6 8 10 11 3 5 7 3 9 10 11 9 1 2 3 14 4 6 7 4 1

2 3 2 1 2 4 13 1 2 3 5 8 9 12 15 1 2 3 10 7 8 12 2 14 6 7 9 12 5 7 10 10 6 1 2 4 5 1 2 3 12 3 5 7 10 3 5 7 13 6 8 12 2 5 6 7 15
1 3 4 3 5 7 8 14 5 9 11 8 5 8 9 5 13 9 10 13 10 1 2 3 1 4 6 9 12 5 6 8 5 3 4 6 7 12 5 7 8 14 5 7 10 2 4 7 8 13 5 8 12 15 4 6 8
3 9 10 14 1 1 3 4 10 9 1 2 3 2 1 2 3 3 2 4 5 12 4 6 9 15 4 6 7 14 7 10 13 5 5 8 10 10 4 5 6 11 7 8 10 13 7 8 11 5 1 7 8 10 2 7
9 11 13 1 2 3 12 5 8 12 11 6 10 11 10 11 5 6 7 9 4 6 8 6 5 7 10 7 3 4 6 3 5 8 12 14 1 3 4 13 7 8 9 10 7 8 11 12 9 10 13 4 1 3
4 15 7 1 2 3 15 6 7 9 1 1 2 3 3 4 6 9 5 4 5 7 8 4 6 8 9 4 6 8 2 4 6 7 4 1 3 4 13 7 9 13 10 7 9 10 11 7 8 11 6 1 2 3 14 3 5 7 12
6 9 11 14 2 7 9 12 4 1 4 5 7 2 4 5 11 4 6 8 14 1 3 4 13 1 3 4 9 2 4 5 8 9 10 14 15 9 10 15 10 1 4 7 6 5 7 9 5 1 3 4 12 1 3 4 3
3 5 7 15 12 3 4 5 1 8 9 12 6 1 2 3 13 4 5 7 2 1 4 7 4 1 2 3 10 3 5 7 15 7 8 11 14 2 4 5 11 1 2 3 5 1 3 5 9 1 2 3 3 5 6 8 8 1 4 5
7 3 4 5 1 1 1 2 5 10 1 5 8 9 15 3 5 6 4 4 6 9 13 1 3 4 11 1 3 4 9 1 4 7 14 6 10 14 5 1 4 7 12 7 8 12 6 2 4 6 2 4 1 3 4 10 1 3 4
4 7 4 7 9 15 5 7 9 13 5 6 8 2 1 3 4
16 10 3 5 9 11 14 7 10 14 10 5 9 10 9 1 2 3 4 1 3 4 2 2 4 6 13 1 3 4 8 7 10 14 12 1 3 4 6 5 6 7 6 1 1 2 5 4 4 6 7 3 6 7 9 12 1
3 4 8 8 9 12 2 1 3 4 8 11 9 10 14 8 6 8 9 5 5 6 8 12 6 8 11 15 7 8 12 10 5 9 11 2 7 10 14 9 3 5 6 12 2 1 2 3 3 1 3 4 1 1 3 4 12
1 3 4 9 4 6 8 4 5 8 11 5 1 3 5 15 1 2 3 13 7 8 10 8 6 8 11 6 1 3 4 11 1 4 5 2 5 3 5 6 13 6 8 10 2 1 6 10 12 3 5 6 8 15 8 7 8 10
1 1 2 3 5 6 10 13 12 1 2 3 10 7 9 13 4 4 5 7 11 5 9 11 14 4 7 8 13 1 2 3 7 5 7 8 6 1 2 3 3 1 2 3 15 2 4 6 2 5 6 8 9 5 7 8 13 2 1
3 4 4 5 9 13 8 3 4 5 6 1 2 3 9 5 6 7 3 2 4 5 12 6 7 9 7 5 6 7 14 8 10 11 5 5 7 9 11 9 10 13 10 3 4 5 1 4 7 10 7 5 6 7 9 14 1 2 3
12 6 9 12 10 1 2 4 1 7 10 13 9 5 8 11 6 1 2 3 7 12 6 7 8 5 1 2 5 1 3 4 6 8 1 2 3 15 8 10 12 7 1 2 5 10 6 9 11 8 11 6 7 10 14 7
8 10 15 7 10 14 7 5 6 8 3 2 4 6 10 7 8 9 12 7 9 10 1 9 10 15 11 7 7 8 12 6 9 10 14 14 2 4 6 11 1 3 4 9 1 2 3 10 4 5 7 15 1 2 3
13 6 10 14 2 1 3 6 8 1 2 3 5 3 4 6 3 1 4 6 8 6 4 5 7 4 5 8 12 15 6 5 8 12 14 9 10 11 5 4 5 7 13 1 2 3 8 5 6 7 4 1 2 4 3 5 6 9 12
6 7 10 9 4 6 7 7 6 7 8 1 6 8 10 10 4 7 9 11 5 9 11 15 9 10 12 2 1 3 4 8 4 1 2 3 15 7 8 10 10 6 8 10 2 2 4 5 11 1 3 4 1 1 3 4 13
4 7 9 3 4 7 9 3 7 1 3 4 9 1 3 4 15 1 3 4
20 4 5 8 9 11 8 5 9 12 6 2 4 5 4 1 3 4 14 11 3 4 6 15 9 10 11 9 1 2 3 13 3 5 6 3 5 6 8 10 1 2 3 2 4 6 8 7 1 3 4 8 1 2 5 6 1 3 4
4 9 10 11 5 4 5 6 14 2 4 5 12 2 4 5 1 13 1 3 4 10 5 3 5 7 14 5 6 7 4 1 3 4 10 6 8 9 1 3 5 6 11 3 5 7 9 3 4 6 2 1 2 3 6 4 6 9 8 4
5 7 11 11 5 7 9 12 9 10 14 8 2 4 5 13 1 2 3 15 6 10 14 6 5 9 13 9 3 4 5 10 1 3 4 5 5 8 10 14 1 3 4 4 6 7 8 7 13 6 7 8 6 5 9 11
15 1 2 3 5 6 10 14 2 6 9 12 10 3 5 6 9 4 7 10 5 9 4 5 6 14 5 8 11 5 3 4 5 7 1 2 3 4 1 2 3 7 2 1 3 4 1 5 7 10 5 8 10 12 12 6 9 11
15 7 8 11 8 4 5 7 9 9 10 11 2 7 6 8 12 9 7 8 10 14 2 6 8 11 1 4 5 6 5 1 4 6 12 6 10 14 4 5 7 10 10 1 3 4 11 6 8 11 6 5 6 7 13
6 8 11 7 5 6 7 15 4 6 9 9 4 7 8 14 6 10 14 8 5 7 10 3 15 9 10 12 1 7 8 11 11 1 3 4 13 8 1 3 4 1 3 5 6 3 3 5 7 15 7 9 13 12 4 5
7 4 7 8 11 13 1 2 4 2 3 4 5 14 4 6 7 7 1 2 3 9 5 7 8 5 7 9 13 6 2 4 6 2 1 4 5 7 5 7 8 11 8 11 1 3 5 7 1 3 4 3 5 6 8 8 4 5 6 14 5
8 9 15 7 8 9 5 6 7 10 13 5 7 9 4 9 3 4 5 1 6 10 12 8 2 4 6 6 7 8 9 1 2 6 7 8 7 8 5 7 9 14 4 6 9 13 5 6 7 6 5 6 7 1 5 7 9 12 5 7 8
9 3 5 6 1 4 3 5 6 15 10 3 5 7 14 4 5 7 13 6 7 10 9 3 5 7 6 3 5 6 11 9 10 11 12 1 2 3 1 6 7 9 5 6 9 13 3 5 6 7 15 1 2 3 4 9 10 11
7 7 9 12 8 5 6 7 2 7 8 11 12 10 9 10 11 6 6 8 9 3 3 5 6 2 1 2 3 11 4 5 6 8 5 7 9 13 1 2 3 12 6 7 9 7 7 9 13 9 5 7 8 5 2 4 5 4 8
10 11
19 3 4 4 5 7 5 3 5 6 12 7 10 15 7 6 1 2 3 7 7 8 11 13 2 4 5 5 1 2 3 12 1 4 5 1 6 7 9 11 8 9 13 13 5 1 3 4 9 7 10 11 14 1 2 3 8
1 4 7 15 6 7 8 4 1 3 4 10 1 2 3 12 8 9 11 2 1 2 3 3 3 4 5 7 2 4 5 13 5 9 13 6 6 9 12 12 5 3 4 5 4 3 4 5 12 1 3 4 14 1 3 5 1 1 3
4 2 3 5 7 9 7 8 9 13 1 2 4 6 1 2 3 7 3 4 6 11 8 9 12 3 1 2 3 3 5 6 9 12 2 6 8 10 8 5 6 7 15 3 4 5 6 6 4 7 9 15 2 4 6 9 5 6 7 1 1
2 3 14 6 8 11 11 2 4 6 13 3 5 6 7 3 5 7 8 6 10 12 5 1 2 3 4 9 10 13 10 1 2 3 12 1 4 6 2 1 2 3 6 9 5 9 11 14 1 2 5 7 5 7 8 15 6
9 12 11 7 8 11 3 5 7 10 11 8 1 4 7 1 5 6 8 13 7 8 11 12 1 4 6 3 3 4 5 15 8 10 14 11 1 2 5 2 1 2 3 6 3 5 7 7 6 9 12 10 6 8 9 14
10 1 3 6 7 1 3 4 13 5 6 8 9 5 6 8 5 3 5 6 1 1 4 5 11 5 8 12 14 5 8 10 15 1 2 3 4 6 7 8 6 1 2 5 12 1 4 5 3 4 6 7 8 1 3 4 6 15 5 9
11 8 2 4 5 2 5 6 8 6 7 8 11 4 7 8 10 10 6 9 11 7 7 5 6 7 5 6 8 9 3 5 9 11 14 6 7 9 13 1 2 3 11 1 3 4 6 3 4 5 13 6 1 2 3 13 1 3 4
8 1 2 3 5 2 4 6 11 1 3 4 10 2 4 5 1 7 8 11 3 7 8 12 2 3 4 5 15 1 2 3 9 6 8 12 12 7 9 12 14 3 5 6 7 10 7 8 9 8 5 6 7 1 1 3 4 6 5
6 7 11 6 7 9 2 7 8 12 7 9 10 11 5 3 2 4 5 15 6 7 8 4 4 5 6 12 4 7 10 5 5 6 7 12 15 5 8 12 11 6 8 9 5 6 8 11 2 7 8 12 6 4 5 6 14
8 9 12 3 6 10 12 8 1 2 3 7 7 8 9 10 1 3 4 4 1 2 3 9 4 5 7 8 6 4 6 7 14 7 8 11 13 6 9 12 15 5 6 8 11 9 10 15 9 7 8 10 7 1 2 4 10
5 9 12 12 9 5 6 8 15 2 4 5 1 7 8 9 12 6 7 10 11 9 10 15 7 8 10 13 14 1 3 4 10 2 4 5 5 6 10 13 6 1 2 3 8 7 8 12 13 6 7 9 5 6 7
10 14 10 5 6 8 12 5 9 11 15 7 8 12 1 6 9 10 13 7 8 9 10 13 3 5 7 4 1 3 4 9 1 2 5 6 2 4 6 10 4 6 7 12 7 9 11 8 5 9 11 15 5 7 9
11 1 2 3 14 3 4 5 3 5 9 10 5 2 4 5
17 13 3 3 5 7 14 1 3 4 5 7 9 13 9 7 9 13 1 1 2 3 7 5 6 8 6 4 6 7 15 6 10 11 13 1 3 6 8 4 7 8 12 3 4 6 4 2 4 5 11 1 3 4 3 2 5 6 9
12 2 4 5 13 1 3 6 3 1 9 10 12 14 5 6 8 12 3 5 7 5 1 4 5 6 15 1 2 5 7 5 9 10 5 4 6 7 10 4 7 9 4 9 1 3 4 11 4 7 8 14 6 8 11 12 1
4 6 13 5 6 7 8 7 4 5 6 15 1 2 3 6 3 4 5 2 9 10 11 10 6 7 8 12 5 9 13 8 1 4 5 4 1 4 6 9 1 3 4 14 2 4 6 3 2 4 5 13 2 4 6 3 8 5 6 7
14 5 6 8 6 1 2 3 6 2 1 3 6 1 5 6 7 10 6 7 10 12 2 4 6 11 2 4 5 9 9 10 12 12 7 3 4 6 14 1 3 4 15 1 2 3 12 1 4 7 11 7 8 10 8 2 4
6 9 2 4 5 4 8 10 15 10 5 9 11 1 5 7 10 2 8 10 13 3 3 5 7 13 12 5 8 10 9 2 4 6 1 1 2 4 2 1 2 3 8 1 3 6 3 1 2 4 15 1 3 4 6 9 10 15
14 9 10 14 5 6 10 12 10 3 5 7 4 1 3 4 13 3 5 6 15 10 6 9 11 11 5 7 10 12 1 2 5 14 1 4 7 4 5 7 10 9 1 3 4 7 3 5 7 1 9 10 13 15
9 10 12 13 5 6 9 2 9 10 11 6 7 10 13 3 5 6 9 5 7 9 12 8 1 2 3 6 4 2 4 5 9 1 2 3 2 5 8 12 15 1 2 5 5 6 7 8 14 5 6 8 10 11 6 9 13
6 1 4 7 9 1 3 4 3 5 6 8 2 9 10 11 10 4 7 9 7 5 9 11 1 1 3 4 13 1 3 4 14 1 2 3 13 12 7 10 11 8 5 9 13 11 1 3 4 10 8 9 11 7 1 3 4
5 5 6 8 9 5 8 10 6 4 7 10 14 1 4 5 2 3 5 7 15 3 4 5 4 1 2 5 13 4 5 7 10 6 2 4 5 14 1 4 5 7 1 2 3 2 1 2 3 5 1 3 4 15 1 3 4 4 1 3 4
13 4 6 8 9 8 9 11 10 1 2 3 11 14 1 3 5 12 8 9 12 8 4 7 8 10 1 2 3 11 7 9 10 1 6 10 12 4 9 10 12 9 2 4 6 15 5 6 7 6 7 8 10 5 6
10 14 13 3 6 10 14 11 2 4 5 5 5 8 11 2 6 9 10 14 5 6 7 8 2 4 6 9 1 4 5 6 6 7 9 7 5 6 9 13 1 3 4 4 1 3 4 10 5 8 11 12 5 9 12
19 13 10 2 4 5 5 3 5 6 13 1 3 4 6 6 10 12 3 3 4 5 1 1 3 4 9 5 6 8 8 4 5 7 12 1 3 4 11 3 5 6 7 1 4 6 4 1 2 3 15 1 3 4 13 2 5 6 9
11 2 4 5 14 3 5 6 12 4 7 9 13 9 10 14 10 5 6 9 9 1 2 3 3 1 2 3 7 7 8 11 8 4 5 6 6 7 10 14 4 3 5 6 15 3 4 6 8 1 7 8 10 15 1 3 4
12 1 2 3 10 6 7 9 11 7 10 12 8 1 3 4 3 7 9 11 6 8 10 12 7 9 1 2 3 8 4 6 9 3 1 3 4 1 3 4 6 11 4 6 8 10 3 4 5 7 5 6 8 13 5 8 10 11
12 6 8 11 2 4 7 8 4 1 3 5 11 7 8 12 14 4 6 9 15 9 10 14 8 6 10 11 7 8 9 10 6 3 4 5 10 1 2 3 13 3 5 7 3 4 7 10 11 8 1 2 3 14 8
9 10 13 5 6 8 3 9 10 11 11 5 6 9 9 4 7 8 12 4 7 9 15 5 7 9 10 6 10 11 1 8 9 12 6 3 4 6 15 6 3 4 5 5 4 7 10 3 7 9 13 7 1 3 4 15
2 4 6 9 5 9 12 14 9 10 15 8 5 6 7 2 4 6 7 12 8 10 11 4 1 3 4 10 6 8 11 11 1 3 4 1 8 10 13 13 1 2 3 3 5 1 2 4 14 2 4 6 3 5 6 8 3
7 1 3 4 11 1 3 4 1 5 9 11 9 10 5 9 11 14 5 9 11 13 3 5 6 4 1 3 4 7 1 3 4 9 9 10 13 15 3 4 5 12 1 3 4 11 6 8 9 10 9 2 4 5 4 1 2
3 15 5 8 12 2 7 9 11 11 1 3 4 14 4 7 10 5 1 2 3 1 3 4 6 10 1 2 3 8 4 6 9 1 4 8 10 13 7 1 1 2 3 12 1 3 4 13 7 10 12 9 3 5 6 7 6

8 10 15 9 10 13 3 3 5 6 5 5 4 5 6 12 2 4 5 1 3 4 6 2 4 7 9 4 6 7 10 7 8 3 4 5 10 3 5 7 13 4 5 7 5 1 4 6 1 9 10 11 14 4 7 8 12 9
10 14 15 12 4 5 7 3 8 9 13 1 8 9 10 14 7 9 11 5 4 6 9 10 1 3 4 6 1 3 4 8 1 3 4 7 1 2 3 4 2 4 6 15 2 4 6 2 6 8 9 11 8 10 12 13 6
8 12 9 7 9 10 2 13 4 5 6 10 5 8 9 15 12 6 7 10 11 8 10 14 13 4 5 6 1 1 4 7 10 6 7 8 5 6 7 10 4 1 3 4 9 1 2 3 14 1 3 4 7 7 9 12
8 7 10 12 15 3 4 6 2 4 6 9 3 1 2 3 6 2 4 5 2 14 6 9 12 3 1 2 4
17 12 3 4 6 7 14 1 2 3 4 5 6 8 7 1 4 7 13 2 4 5 15 6 7 9 12 8 9 11 9 8 9 11 6 4 5 7 2 1 3 4 5 9 10 15 10 6 9 10 6 3 7 9 11 4 1
2 3 6 1 2 3 5 5 6 8 9 1 3 4 15 1 3 4 9 6 5 6 8 4 1 2 3 8 5 7 9 1 1 2 3 11 5 6 8 15 6 10 11 5 1 3 4 7 5 6 8 9 5 8 11 10 15 3 5 6 5
1 2 3 13 6 8 10 11 7 8 9 3 3 4 5 6 4 5 6 9 1 2 3 8 4 5 6 12 1 4 6 10 5 8 9 15 1 2 4 6 7 1 4 7 12 5 6 9 9 4 7 9 13 5 9 12 2 1 3 6
8 1 3 5 5 6 9 12 3 4 7 9 6 6 8 10 10 2 4 5 4 3 4 5 14 6 7 10 11 5 9 10 15 1 3 4 10 7 1 2 5 13 1 3 4 8 4 5 6 5 7 10 15 10 9 10
11 11 5 6 8 9 1 2 3 2 6 10 13 4 3 5 7 12 1 3 4 13 8 1 2 5 9 1 2 3 6 5 7 9 15 1 3 4 2 3 4 6 3 7 8 10 7 6 7 9 11 1 3 4 14 4 6 7 12
4 6 8 5 7 9 12 4 1 3 5 13 9 10 14 5 10 7 9 12 8 4 6 7 2 1 3 4 11 7 8 10 5 5 7 10 7 2 1 3 4 11 1 3 4 14 1 3 4 12 3 5 6 3 4 6 9 5
1 2 3 10 5 6 7 6 1 4 5 6 13 3 4 5 10 2 4 5 7 1 2 3 2 9 10 14 5 7 9 10 7 6 1 2 4 12 4 6 7 1 1 3 4 15 1 2 3 3 1 3 5 8 8 10 13 10 4
5 6 15 7 5 6 8 13 1 2 3 12 5 9 10 3 9 10 14 11 3 5 6 10 4 5 6 8 6 8 10 6 8 9 10 14 3 4 5 15 2 4 5 5 6 7 9 9 1 3 4 4 8 9 13 1 4
5 7 2 6 8 9 3 2 1 3 4 6 1 4 6 13 1 3 4 1 12 1 2 3 10 10 5 6 7 6 5 9 10 13 5 7 10 8 1 3 4 9 3 5 7 15 3 5 6 3 5 8 9 14 2 4 5 2 7 10
11 7 4 6 7 3 5 1 2 3 12 1 2 3 14 4 6 9 7 1 8 10 11 2 8 10 13 6 1 2 3 5 7 9 12 11 1 3 6 7 3 5 7 12 2 4 5
19 4 5 3 5 7 6 7 9 12 8 5 7 10 12 6 8 10 10 10 1 2 3 8 1 2 3 4 5 6 7 3 5 8 12 12 5 6 9 5 2 4 6 14 2 4 6 1 4 6 7 6 6 8 10 7 4 5 7
13 1 6 9 12 14 1 3 4 15 5 9 12 7 2 4 5 6 5 7 8 2 1 2 3 13 7 8 12 11 2 4 5 10 3 5 7 8 3 5 6 5 3 5 6 12 9 10 14 3 1 3 4 9 1 8 9 12
5 6 9 10 9 9 10 12 15 1 3 4 10 6 8 11 8 6 7 10 4 6 7 8 12 6 7 10 2 1 3 4 2 15 3 4 5 3 4 6 8 10 2 5 6 8 10 3 5 6 13 4 5 6 15 5 6
7 1 7 8 9 3 8 10 14 9 5 6 9 5 1 3 4 14 4 5 7 6 5 6 8 11 9 7 8 9 2 6 7 8 7 5 9 13 12 1 2 3 11 1 4 6 8 4 5 7 10 4 6 8 3 1 4 5 15 6
9 11 13 1 3 4 5 1 3 5 8 14 7 8 11 5 2 4 6 11 2 4 6 12 5 6 9 1 3 4 5 3 1 2 3 4 7 8 9 6 5 6 9 5 6 7 8 11 9 5 9 10 14 4 6 9 2 7 8 12
13 3 5 7 7 13 9 10 15 11 5 6 8 7 4 7 8 12 8 10 15 14 4 7 8 3 3 5 6 10 6 7 8 6 4 2 4 5 13 6 10 11 7 1 4 5 15 5 6 9 11 7 8 9 3 1
3 4 5 4 1 3 4 3 4 5 7 10 7 9 12 9 1 2 3 12 1 2 3 4 11 6 8 9 5 1 3 4 14 7 10 11 12 1 3 4 12 10 6 9 10 14 1 4 6 12 7 9 10 11 5 7
9 5 5 7 9 3 3 4 5 8 1 3 4 13 8 9 10 2 4 7 9 6 6 9 11 1 4 5 7 4 3 5 7 14 6 5 6 7 3 5 6 7 13 2 4 5 2 1 3 4 11 1 3 4 10 1 4 6 8 3 4
5 15 3 4 6 9 4 5 6 14 2 4 5 5 7 8 12 12 1 3 5 7 1 2 3 1 2 4 5 14 15 9 10 12 14 5 6 7 7 6 9 13 9 6 7 8 10 8 10 12 5 9 10 12 12
1 2 3 8 1 3 4 4 9 10 14 11 5 6 9 6 1 3 5 13 3 4 6 3 6 8 10 2 3 5 7 9 15 5 9 12 7 6 7 8 1 5 8 9 2 3 5 7 13 1 4 7 14 5 7 10 9 5 7
9 12 5 8 9 6 5 9 13 2 4 3 5 6 1 2 4 5 3 7 5 6 7 12 5 6 8 15 5 7 9
19 15 14 8 10 12 5 8 9 13 15 1 2 3 8 1 3 4 9 1 3 4 10 6 8 11 6 1 3 4 13 3 4 5 7 6 8 12 11 1 2 3 1 1 3 4 12 3 4 6 2 6 7 9 4 1 2
3 3 4 7 9 3 15 1 2 3 14 7 8 11 2 4 7 8 13 13 6 7 9 6 1 3 4 11 4 6 9 2 5 9 10 15 6 7 10 10 1 2 5 12 7 10 14 9 5 6 8 3 5 7 9 7 2
4 5 5 2 4 6 14 5 9 10 4 8 9 13 11 15 8 10 15 1 3 5 6 13 7 8 9 3 7 10 14 8 1 3 4 11 7 10 14 9 1 4 6 5 4 5 7 12 6 7 9 7 5 6 9 2 5
7 9 2 10 3 5 7 12 4 7 8 1 2 1 3 4 15 2 4 5 6 12 6 7 8 6 4 5 6 14 5 6 7 9 6 9 13 10 8 9 13 3 1 2 4 15 3 4 5 7 6 10 14 13 4 7 10
8 9 10 12 4 1 4 6 1 8 10 11 5 8 9 10 11 2 4 5 8 11 5 6 7 5 1 3 6 14 1 4 7 13 1 3 4 10 3 4 5 4 4 5 6 8 5 6 7 6 1 3 4 6 15 3 5 7
11 4 7 8 2 2 4 5 3 6 9 11 9 1 3 4 14 1 3 6 9 10 1 2 3 15 5 9 13 4 3 5 7 2 1 2 3 5 8 10 14 8 6 10 15 14 6 9 13 13 4 5 6 7 1 2 3
7 15 8 10 12 7 4 5 6 14 7 8 10 12 4 7 8 2 1 2 3 13 1 3 4 1 5 6 7 14 1 5 7 8 6 3 4 5 11 1 2 3 12 6 10 12 7 8 10 14 10 1 2 3 8 5
6 8 15 2 4 5 13 4 5 6 14 1 3 4 5 7 8 12 3 3 5 7 4 8 10 14 9 1 2 3 5 15 6 8 11 10 1 2 4 6 1 2 3 14 3 4 6 11 8 10 15 1 14 1 2 5 5
12 6 8 9 15 3 4 5 5 3 5 6 11 7 9 13 3 4 5 7 6 11 5 6 9 15 2 4 5 5 1 3 4 6 1 3 4 2 5 9 13 12 3 4 6 4 14 1 2 3 13 5 8 10 12 7 8 11
1 3 5 6 5 1 9 10 11 11 6 8 11 13 5 6 7 12 6 7 10 2 9 10 11 5 3 1 2 3 14 6 8 9 6 6 7 8 12 6 10 15 2 2 4 5
16 11 5 1 2 3 14 7 8 11 12 7 8 11 7 4 7 9 1 1 2 3 13 8 9 12 10 9 10 14 8 5 7 10 15 5 6 7 6 5 8 9 3 7 10 15 10 12 5 7 10 14 7
9 13 8 5 8 9 15 4 7 8 10 4 5 6 4 1 2 3 9 1 2 3 11 5 7 10 2 2 4 5 7 1 2 3 6 3 6 9 12 14 1 2 3 10 4 5 7 4 3 4 5 5 4 5 6 13 6 9 11
14 1 1 4 6 14 7 9 12 9 4 5 6 12 4 6 7 6 1 2 3 11 3 5 7 15 1 2 3 13 5 7 8 5 3 5 6 7 6 10 13 3 1 2 3 8 1 2 3 4 1 2 3 2 1 2 3 15 3
9 10 13 11 9 10 11 9 1 2 3 7 5 8 11 14 1 2 3 10 5 9 10 8 1 4 6 6 9 10 13 12 1 3 4 15 1 3 4 1 8 10 13 4 1 2 3 2 1 4 5 13 4 6 7
5 2 4 5 2 11 5 8 11 4 7 8 11 12 15 1 4 6 6 9 10 12 1 1 2 3 2 7 8 11 13 1 2 3 12 1 2 3 9 4 5 7 11 6 7 9 7 5 6 9 8 1 3 4 14 9 10
14 4 5 7 9 3 1 6 8 9 12 1 3 5 6 1 3 4 12 3 5 9 13 11 8 10 15 8 1 3 4 2 2 4 5 9 5 7 8 5 4 6 9 15 9 10 15 13 7 8 10 4 1 3 4 6 1 2
3 12 4 6 7 10 5 6 9 13 6 6 7 10 12 6 8 11 14 7 9 13 9 8 10 11 11 6 8 9 10 4 7 8 13 1 2 3 15 7 9 12 3 5 6 9 7 1 3 4 5 9 10 15 1
5 6 7 4 6 9 10 5 1 1 2 3 5 1 3 4 15 4 7 8 11 4 5 7 13 6 8 9 5 12 7 8 9 14 7 8 10 6 3 5 7 9 7 8 9 10 1 3 4 11 15 4 7 9 10 1 3 4 4
3 5 7 7 7 8 12 11 5 8 12 3 5 7 8 14 2 4 5 8 7 8 11 9 4 5 7 6 1 3 4 13 1 2 4 13 5 1 3 4 14 8 9 11 13 5 8 11 4 3 5 7 3 5 6 7 15 7
8 9 10 1 2 3 1 1 3 4 8 5 7 10 9 5 9 11 6 3 5 6 7 1 2 4 11 3 5 7 6 3 3 5 7 8 1 3 4 13 5 9 12 14 4 5 7 9 5 9 10 12 4 5 6 8 15 6 7
8 10 1 2 3 14 6 7 9 3 1 2 3 7 5 6 9 4 6 8 12 11 1 3 4 8 5 9 10
18 14 14 1 3 4 4 7 8 10 11 7 9 12 12 9 10 15 9 7 8 9 7 7 10 13 8 5 7 9 1 7 8 12 6 1 2 3 13 5 7 10 10 4 7 8 15 8 10 12 3 1 2 4
5 9 10 15 2 2 4 7 9 6 1 3 4 9 1 5 8 9 4 3 4 6 2 1 2 4 3 5 9 10 14 5 8 9 13 3 4 5 5 6 8 9 7 1 3 4 12 1 2 3 10 2 3 5 6 14 1 3 4 13
3 4 6 12 2 4 5 10 3 5 6 1 1 3 4 9 1 3 5 6 1 4 7 11 7 9 13 5 1 3 4 9 3 6 8 12 8 3 5 6 14 6 9 11 1 1 3 4 2 3 4 6 10 3 5 6 11 1 3 4
9 1 2 3 7 2 4 6 6 15 1 4 7 13 7 10 12 8 1 3 4 1 2 4 6 9 3 5 7 5 9 10 13 8 13 1 4 5 12 1 4 5 6 1 4 7 2 1 3 6 9 4 7 10 10 1 2 3 5
3 5 7 15 3 5 7 2 4 5 8 11 9 9 10 11 12 4 1 4 7 5 5 6 8 12 1 4 6 14 6 7 10 9 1 2 4 1 7 10 11 6 2 4 6 8 6 10 12 13 1 3 4 10 3 4 6
3 3 4 6 15 6 7 9 15 6 1 2 3 11 3 4 5 15 4 5 7 14 5 7 8 8 4 5 7 5 1 3 4 13 5 6 9 9 4 6 9 4 1 2 3 10 7 8 12 12 1 3 4 2 1 2 3 3 4 5
7 1 5 8 10 7 9 10 12 2 3 5 9 10 11 8 10 11 5 9 7 9 11 4 7 10 15 1 7 8 11 10 2 4 5 2 5 7 8 9 5 5 9 13 9 4 6 8 7 6 9 11 12 6 7 8
11 5 9 10 8 1 2 4 15 1 3 4 13 1 4 6 14 3 4 6 3 9 6 7 10 6 4 7 10 2 5 7 8 12 12 1 3 4 3 3 4 6 13 4 5 6 14 3 4 6 9 6 9 10 10 1 3
4 15 7 8 10 4 1 3 4 8 6 7 10 2 1 4 7 5 3 5 6 11 6 8 10 1 10 3 4 5 12 7 6 9 10 13 2 4 6 9 2 4 6 1 4 7 10 8 2 4 6 5 6 9 11 15 1 2
3 11 2 4 6 12 3 4 6 6 1 2 3 2 1 4 5 3 1 3 4 7 8 1 4 5 4 7 8 10 6 6 7 10 15 4 6 7 13 1 2 3 1 7 9 13 2 1 2 3
17 13 14 7 9 10 7 7 8 11 4 3 4 5 3 1 3 4 11 4 7 9 10 1 2 3 13 1 3 4 12 1 2 3 1 1 3 4 6 5 6 8 8 7 9 13 2 2 4 6 9 1 3 4 12 12 1 4
7 15 9 10 12 8 1 2 3 2 4 6 9 11 1 3 4 10 5 7 10 9 5 6 7 3 3 5 6 14 2 4 5 6 1 3 4 1 1 4 6 4 1 3 4 15 2 3 4 6 10 1 2 3 7 1 3 4 12
3 5 7 1 1 3 5 14 4 5 6 11 3 5 6 8 5 8 10 3 1 2 3 9 9 10 11 5 5 9 12 15 1 3 6 13 2 4 5 4 6 7 9 6 6 9 11 7 3 4 5 6 9 5 9 12 10 5 6
8 13 5 6 8 15 5 9 13 14 6 7 8 1 7 8 11 4 6 5 8 11 1 5 8 11 13 4 7 8 4 1 3 5 9 3 5 8 9 7 1 3 4 5 1 3 4 6 1 2 3 13 1 2 3 10 3 5 7
15 4 5 6 1 1 4 7 4 5 7 10 1 5 1 2 3 11 11 5 8 10 14 6 10 12 13 1 3 4 7 7 8 9 8 6 7 8 9 1 2 3 12 5 9 11 6 1 2 3 1 1 3 5 15 1 2 3
4 2 4 5 6 12 5 6 8 10 8 10 14 15 5 9 12 3 1 2 3 14 9 10 15 13 9 10 13 7 3 1 2 3 2 1 3 4 10 4 5 6 5 1 3 4 11 4 6 9 8 3 4 6 9 1 2

3 2 1 5 6 9 12 2 4 5 11 4 2 4 5 2 7 10 15 1 9 10 13 12 6 9 13 11 3 5 6 14 1 3 5 15 5 8 10 9 3 5 7 7 1 3 4 6 9 10 11 5 4 6 7 8 2
4 5 6 14 1 2 3 15 7 10 13 11 8 9 12 1 7 9 12 10 5 6 9 5 6 10 11 3 5 9 12 9 2 1 4 6 13 3 5 6 1 7 10 15 5 5 7 9 4 9 10 15 10 1 4
6 3 1 2 3 8 9 10 11 12 4 6 7 15 10 1 2 3 4 5 7 9 8 1 3 4 6 6 8 9 1 1 3 5 14 1 2 3 9 5 8 10 13 4 5 7 5 3 4 5 3 1 2 3 15 4 5 6 12
2 4 5 11 2 4 5 2 8 10 11 7 4 6 9 9 8 5 8 11 15 1 2 3 14 6 9 10 9 9 10 13 11 6 7 10 10 4 5 7 7 5 9 13 2 9 10 12 4 6 7 10 13 4 4
7 9 14 5 9 11 5 4 5 7 6 9 10 15 12 1 3 4 2 5 6 9 9 2 4 5 8 5 8 9 7 8 9 12 15 5 8 11 1 4 7 9 10 1 2 3 11 5 7 9