Evolutionary Algorithms for Resource Constrained Project Scheduling

Ismail Mohamed Ismail Ali

B.Sc. (Information Systems and Technology) Zagazig University, Egypt

A thesis submitted in partial fulfilment of the requirements

for the degree of Master by Research

School of Engineering & Information Technology

University of New South Wales at

the Australian Defence Force Academy

March 2016

http://doi.org/10.26190/5dc4f1a1f31c9

PLEASE TYPE

THE UNIVERSITY OF NEW SOUTH WALES

Thesis/Dissertation Sheet

Surname or Family name: **Ali**

First name: **Ismail** Other name/s: **Mohamed Ismail**

Abbreviation for degree as given in the University calendar:
Master (Research)

School: **Engineering and Information Technology (SEIT)** Faculty:

Title: **Evolutionary Algorithms for Resource Constrained Project Scheduling**

Abstract 350 words maximum: (PLEASE TYPE)

Resource constrained project scheduling problems (RCPSPs) are well-known NP hard combinatorial problems. Due to the drawbacks of existing solution approaches, many researchers have proposed different evolutionary algorithms (EAs) for solving them. Although EAs are able to achieve near-optimal solutions, they cannot guarantee optimality and, in fact, no single EA has consistently been able to solve all types of problems. This has led to the emergence of hybrid methods which have shown good performances but their search capabilities for solving RCPSPs have not yet been fully explored.

In this thesis, to efficiently solve RCPSPs, an algorithmic framework involving multiple methodologies is introduced. Firstly, a memetic algorithm (MA) consisting of a new heuristic for converting infeasible solutions to feasible ones in the initial population and multiple local search (MLS) strategies for increasing the exploitation capability of the algorithm is proposed. Secondly, an improved differential evolution (DE) algorithm containing new search operators that can guarantee the generation of feasible solutions, even from infeasible ones, is introduced. Finally, motivated by the encouraging performances of the proposed MA and DE, a bi-evolutionary algorithm (bi-EA) that utilizes the good search features of both these algorithms by automatically switching between them according to their performance, which implies placing more emphasis on the best-performing one during the evolutionary process, is proposed. In addition, two heuristic approaches developed to guide the solutions in both the initial population and every generation towards feasibility are adopted.

All the algorithms proposed in this thesis are tested on a set of well-known project scheduling problems taken from the PSPLIB, with the results for instances of 30, 60, 90 and 120 activities compared with both each other and state-of-the-art algorithms. It is found that: (1) the heuristic method improves the performance of the traditional GA by 80.66% in terms of quality of solutions; (2) the use of MLS techniques leads to much better solutions (11.83%) and saves 20.21% of the GA's computational time; (3) adopting the heuristic method in DE improves the quality of solutions by 28.96% and saves 10.62% of CPU time; (4) the improved DE operators provide much better solutions and greater savings in computational time than a traditional one; (5) bi-EA outperforms both MA and DE in terms of solution quality, especially for large-scale problems as, on average, it obtains 4.33% and 3.5% higher-quality solutions than MA and DE, respectively, and also provides competitive solutions compared with those from state-of-the-art-algorithms.

ORIGINALITY STATEMENT

‘I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.’

Signed ..

Date .

COPYRIGHT STATEMENT

'I hereby grant the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or part in the University libraries in all forms of media, now or here after known, subject to the provisions of the Copyright Act 1968. I retain all proprietary rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.
I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstract International (this is applicable to doctoral theses only).
I have either used no substantial portions of copyright material in my thesis or I have obtained permission to use copyright material; where permission has not been granted I have applied/will apply for a partial restriction of the digital copy of my thesis or dissertation.'

Signed .

Date

AUTHENTICITY STATEMENT

'I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis. No emendation of content has occurred and if there are any minor variations in formatting, they are the result of the conversion to digital format.'

Signed

Date

Abstract

Resource constrained project scheduling problems (RCPSPs) are well-known NP hard combinatorial problems. Due to the drawbacks of existing solution approaches, many researchers have proposed different evolutionary algorithms (EAs) for solving them. Although EAs are able to achieve near-optimal solutions, they cannot guarantee optimality and, in fact, no single EA has consistently been able to solve all types of problems. This has led to the emergence of hybrid methods which have shown good performances but their search capabilities for solving RCPSPs have not yet been fully explored.

In this thesis, to efficiently solve RCPSPs, an algorithmic framework involving multiple methodologies is introduced. Firstly, a memetic algorithm (MA) consisting of a new heuristic for converting infeasible solutions to feasible ones in the initial population and multiple local search (MLS) strategies for increasing the exploitation capability of the algorithm is proposed. Secondly, an improved differential evolution (DE) algorithm containing new search operators that can guarantee the generation of feasible solutions, even from infeasible ones, is introduced. Finally, motivated by the encouraging performances of the proposed MA and DE, a bi-evolutionary algorithm (bi-EA) that utilizes the good search features of both these algorithms by automatically switching between them according to their performance, which implies placing more emphasis on the best-performing one during the evolutionary process, is proposed. In addition, two heuristic approaches developed to guide the solutions in both the initial population and every generation towards feasibility are adopted.

All the algorithms proposed in this thesis are tested on a set of well-known project scheduling problems taken from the PSPLIB, with the results for instances of 30, 60, 90 and 120 activities compared with both each other and state-of-the-art algorithms. It is found that: (1) the heuristic method improves the performance of the traditional GA by 80.66% in terms of quality of solutions; (2) the use of MLS techniques leads to much better solutions (11.83%) and saves 20.21% of the GA's computational time; (3) adopting the heuristic method in DE improves the quality of solutions by 28.96% and saves 10.62% of CPU time;

(4) the improved DE operators provide much better solutions and greater savings in computational time than a traditional one; (5) bi-EA outperforms both MA and DE in terms of solution quality, especially for large-scale problems as, on average, it obtains 4.33% and 3.5% higher-quality solutions than MA and DE, respectively, and also provides competitive solutions compared with those from state-of-the-art-algorithms.

Keywords

Resource constrained project scheduling, scheduling problems, evolutionary algorithms, genetic algorithm, differential evolution, memetic algorithm, multiple local searches, hybrid algorithms and multiple methodologies.

Acknowledgments

PRAISE IS TO ALLAH WHO HIS GRACE IS RIGHTEOUS. All praises are to ALLAH the almighty God, the most Beneficent and the most Merciful. I am thanking Him for guiding and helping me to complete this thesis which I hope will be useful to scientific research in the fields of operations research and computer science.

This thesis owes its existence to the help and support of several people. Firstly, I would like to express my sincere appreciation and gratitude to Prof. Ruhul Sarker for his guidance during my research. His support, time and inspiring suggestions have been precious for the development of this thesis content. Secondly, I thank Prof. Tapabrata Ray for his useful discussions and suggestions regarding my work. I am also indebted to my co-supervisor, Dr. Saber Elsayed, who has been a constant source of support and encouragement, not only during the two years of my Master program but also through the four years of my undergraduate study. I would like to thank them all for the time and effort they spent reviewing my work and publications. I am also grateful to all the members of my research group for their valuable comments and advice.

Special thanks go to my ever-supportive partner in life, my wife (Sara), for her love, patience and support through helping me even in the most difficult situations and providing me with all the means required to complete this thesis.

My greatest gratitude goes to my parents for their constant prayers and encouragement for me to attain this degree. I would not have been able to complete this thesis without their support and motivation. May ALLAH bless and provide them with health and wellness. I would also like to thank my brothers and sisters (Mohamed, Mahmoud, Samah, Doaa and Sabah) and brothers- and sister-in-law (Ahmed, Mohamed and Etab) as well as all the members of their families for sharing their joyful moments with me.

I will never forget all the chats and beautiful moments I shared with my friends and office mates and wish to thank all of them – Saber Elsayed, Ibrahim Hamed, Hassan Abbas, Essam Soliman, Abdelmonaem Fouad, Karam Sallam, Ahmed Thabet, Ahmed Samir, Ehab Taher, Elwy Hassan, Nour Mostafa, Ahmed Fathi, Amr Ghoneim, Mohamed Mabrok, Ehab Zaki, Ayman Ghoneim, Ripon Chakrabortty, Forhad Zaman, my brother Mohamed AbdelBaset and my friends in Egypt (Sayed Mostafa, Sayed Sami and many others).

I express my gratitude to Denise Russell for her suggestions regarding improving my thesis writing.

Finally, I am very grateful to all the people I met along the way and who contributed to the development of my thesis. Also, I would like to thank the University of New South Wales in Canberra, ACT (UNSW@Canberra) for its financial support and the assistance provided to me by the school's administration and IT staff.

List of Publications

- Ismail M. Ali, Saber M. Elsayed, T. Ray, and Ruhul A. Sarker, "Memetic algorithm for solving resource constrained project scheduling problems," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*, 2015, pp. 2761-2767.
- Ismail M. Ali, Saber M. Elsayed, T. Ray, and Ruhul A. Sarker, "A Differential Evolution Algorithm for Solving Resource Constrained Project Scheduling Problems," in Artificial Life and Computational Intelligence: Second Australasian Conference, ACALCI 2016, Canberra, ACT, Australia, February 2-5, 2016, Proceedings, T. Ray, R. Sarker, and X. Li, Eds., ed Cham: Springer International Publishing, 2016, pp. 209-220.
- Ismail M. Ali, Saber M. Elsayed, T. Ray, and Ruhul A. Sarker, "Bi-Evolutionary Algorithm for Resource Constrained Project Scheduling," **under preparation.**

Table of Contents

List of Tables

List of Figures

List of Abbreviations

RCPSP	Resource constrained project scheduling problem
PSPLIB	Project scheduling problem library
NP-hard	Non-deterministic polynomial hard
Bi-EA	Bi-Evolutionary algorithm
EA	Evolutionary algorithm
GA	Genetic algorithm
DE	Differential evolution
ES	Evolutionary strategy
EP	Evolutionary programming
EC	Evolutionary computation
GP	Genetic programming
MA	Memetic algorithm
MLS	Multiple local search
ACO	Ant colony optimization
SI	Swarm intelligence
PSO	Particle swarm optimization
AL	Activity list
RS	Resource strength
RF	Resource factor
NC	Network complexity
STD	Standard deviation
GA_LR	Lagrange relaxation-based GA
B&B	Branch and bound algorithm
IP/LP	Integer programming and linear programming
NS	Neighbourhood search
SGS	Scheduling generation scheme
SSG	Serial scheduling generation
PSG	Parallel scheduling generation

Chapter 1

Introduction

This chapter provides a brief background to the research conducted for this thesis. The problem definition and its practical importance are discussed and then the objectives and scientific contributions of this study are presented. Finally, the organization of this thesis is discussed.

1.1 Background

Scheduling is a decision making process used regularly in many manufacturing and service industrials. It deals with allocation of resources, such as machines, airport runways, crews at a construction site, to tasks, which may be production operations, landings and take offs at an airport, over given time periods and its aim is to optimize one or more objective (Pinedo, 2012).

Scheduling is not a new subject! It has a long and active history dating back almost to the 1950s when the first scheduling strategies were proposed and analyzed. The use of projects and applications of project management continues to increase in our society and organizations which aim to achieve significant outcomes with limited resources and critical time constraints; for example, almost every activity undertaken, such as advertising and political campaigns, voter registration drives, a family's annual vacation and even seminars on the topic of scheduling, are organized as projects (Meredith and Mantel Jr, 2011). Scheduling plays an important role in many industrial and production systems and in most of information processing environments. It also has a significant role in service industrials, such as transportation and distribution settings (Pinedo, 2012). In fact, it has emerged in our society due to the exponential growth of human knowledge and the increasing demand for complex and sophisticated projects, goods and services, the development of which can be accomplished by the expanding amount of knowledge provided by various academic disciplines. The use of science/knowledge for these developments requires high levels of coordination and collaboration among individuals and groups, with a powerful tool needed to control and manage relationships among them (Burke, 2013, Meredith and Mantel Jr, 2011)

During the last few decades, scheduling or project management has provided organizations with powerful tools that improve their capability to design, organize, implement and control their activities and discover the best ways of using their resources. For instance, the United Sates Navy's Polaris program and NASA's Apollo space program were able to successfully accomplish their tasks by applying scheduling approaches (Meredith and Mantel Jr, 2011).

The main purpose of creating a project is to achieve some objectives/goals. Based on actual experiences, most organizations indicate that using scheduling can help them obtain better control and client relations because it allows managers to be responsive to customers by expecting, identifying and solving problems at an early stage. Moreover, many users report numerous advantages of scheduling, such as (1) achieving goals with lower costs and higher quality, (2) providing reliable results with higher profit margins and (3) requiring shorter development times (Meredith and Mantel Jr, 2011, Munns and Bjeirmi, 1996).

On the other hand, some organizations have reported that using scheduling may lead to increasing organizational complexity and higher costs. Therefore, in practice, a proper appreciation of the difficulty of a problem and the extent of the need for scheduling is very important for achieving an appropriate balance between the advantages and disadvantages of using project management/scheduling for that problem (Meredith and Mantel Jr, 2011, Baccarini, 1996).

A wide variety of exact, heuristic and meta-heuristic strategies for comprehending scheduling problems has been proposed. Many simple models based on exact methods for solving RCPSPs have been proposed and were able to find optimal solutions; however, in most cases, they are time consuming, particularly when solving large problems (Demeulemeester and Herroelen, 1992, Jalilvand et al., 2005). Heuristics were initially based on experts' knowledge and experience and aimed to explore the search space in a particularly convenient way (problem-dependent techniques) (Gavrilas, 2010). On the other hand, meta-heuristics are problem-independent techniques, which do not take advantage of any specificity of the problem and, therefore, can be used as black boxes (Beheshti and Shamsuddin, 2013). Both heuristic and meta-heuristic approaches are powerful and flexible search mechanisms that have successfully solved complex problems. Their algorithms aim to obtain good-quality solutions in reasonable computational times and are suitable for practical problems which often have large dimensions and very complex constraints (Das and Acharyya, 2011a, Kolisch and Hartmann, 1999).

Evolutionary algorithms (EAs) are well-known meta-heuristic methods and it contains a number of algorithms that have been used to solve scheduling problems, such as the genetic algorithm (GA) (Toklu, 2002) and differential evolution (DE) (Damak et al., 2009), and swarm intelligent algorithms such as ant-colony optimization (ACO) (Dorigo, 1992) and particle swarm optimization (PSO) (Kennedy, 2010).

1.2 Problem statement

Resource-constrained project scheduling problems (RCPSPs) are well-known scheduling problems. It is also a challenging research topic because of its significance in real life and its emerging in numerous fields. In classical RCPSPs, a project comprises of set of activities, where each activity must be executed just once in a single mode and each activity has its own pre-known resource requirement and execution time. RCPSP aims to schedule the project activities in such a way that minimizes the total duration (makespan) of the project subject to resource availability and precedence constraints that must be strictly satisfied. Precedence constraints (or predecessors-successors relationships) guarantee a logical process of the project activities (i.e. each activity cannot be scheduled until all its predecessor activities are scheduled). Resources in RCPSP can be categorized as renewable and non-renewable. Renewable are available with their full capacity in every time period and periodically renewed, but their quantity may differ from one period to the next. For instances: machines, manpower, equipment, and energy. In contract, non-renewable resources are limited for the entire project, not for each time period, such as raw materials and budget. In this thesis, solving single-mode RCPSPs with different resource types are the main focus.

Although the simplicity of its definition, the RCPSP was proven to be one of the NP-hard optimization problems (Garey and Johnson, 1979) and the most intractable classical problems in reality. Due to the significance of RCPSP in our daily life, its essential role in the growth of activities in many fields and its industrial relevance, solving the RCPSP has become a prosperous research subject.

Despite the fact that obtaining the optimal schedule is a very difficult task because of the highly constrained nature of scheduling problems, particularly for large ones, and the lack of algorithms that either have the capability to solve or is suitable for a wide range of them, over time, many scientific research studies have proposed methods for solving increasingly complex RCPSPs. However, some were only applicable for solving small problems and others were able to achieve near-optimal solutions, but there was no guarantee that they could achieve the optimal solutions.

1.3 Motivations and scope of research

As mentioned above, due to the high complexity of RCPSPs (as explained in Chapter 2), many exact, heuristic and hybrid algorithms have been used for optimally solving them in reasonable computational times. However, exact methods are only applicable for solving small project instances (Demeulemeester and Herroelen, 1992, Jalilvand et al., 2005) as they are not computationally practical for large ones. Heuristic methods can find near-optimal solutions at an acceptable computational cost. However, they do not guarantee optimal results (Abdolshah, 2014). Meta-heuristics are an appealing choice for implementation in general-purpose software as they can be easily adapted to a particular problem (Ólafsson, 2006). However, as they have many drawbacks, improving the performances of existing ones or developing new ones appears to be necessary. Of all methods, the hybrid algorithms show a very promising performance while dealing with RCPSPs; however, their actual capabilities have not yet been fully explored.

Moreover, many GAs and DE have been introduced for solving RCPSPs. However, GA was able to achieve good results, it had a tendency to converge towards local optima or even arbitrary points rather than the global optimum of the problem. Also, the performance of DE deteriorated as the dimensionality of the search space increased (Das et al., 2009) and although DE was good at exploring the search space, it was slow at exploiting the solution (Noman and Iba, 2008).

Furthermore, no single algorithm has been yet able to solve a wide range of optimization problems with consistent quality (Elsayed, 2012). Although the idea of multi-method has been emerged to tackle this drawback, it has not been adopted to solve RCPSPs. This indeed needs further work to carefully and efficiently design a multi-method framework. Similar ideas, such as hybrid approaches, have been proposed to deal with this drawback, but they still need further research, as their performance is still not good enough.

Therefore, all of these issues encourage the development of more efficient algorithms for solving RCPSPs.

1.4 Objectives of this thesis

As the above are significant gaps in the literature, the development of new algorithms that could solve a wide range of test problems in a reasonable time with good-quality solutions would be valuable.

This research investigates the use of GA and DE to solve RCPSPs. Its overall objective is to study, construct and apply an improved GA and DE to obtain solutions of high quality in a reasonable time for RCPSPs, and then develop an appropriate ensemble of them for solving RCPSPs with good quality solutions and less computational time.

In order to accomplish this primary objective, several sub-objectives are to:

- Carry out literature review in project scheduling in general and RCPSPs specifically in order to comprehend the difficulties of these problems, and to review relevant methodologies that researchers have developed to handle them (Chapter 2);

- develop a heuristic repairing method for improving the feasibility of individuals in the initial population for a RCPSP and study its effect on the performances of GA and DE (Chapter 3);
- improve the performance of GA by designing a new memetic algorithm (MA) (Chapter 3);
- enhance the performance of existing DE algorithms by proposing a new DE algorithm by enhancing its mutation and crossover operators (Chapter 4);
- develop a bi-evolutionary algorithm (bi-EA) that incorporates the heuristic repairing method with GA and DE (Chapter 5);
- analyze the effects of the different parameters used in the proposed algorithms (Chapter 5);
- carry out a systematic experimental study of MA, DE and bi-EA for RCPSPs;
- validate the performances of the proposed algorithms by comparing them with those of each other and state-of-the-art algorithms.

1.5 Contribution to scientific knowledge

Most of the exploratory investigation in this research is conducted using well-known benchmark RCPSP instances, based on which the performances of different algorithms, including the MA, DE and bi-EA developed in this study, are examined.

The following are the scientific contributions from this research.

- An experimental analysis of the suitability of a GA for RCPSPs is conducted. From its results, it can be concluded that a traditional GA with simple crossover and mutation operators and without any local search (LS) strategies is able to produce good solutions for small scale instances, but its performance is not that good when the size of the instances is increased. A new multiple LS strategy included in GA helps to improve the solutions with reasonable computational expenses.

- A new repairing method proposed for GA is basically a heuristic strategy that enhances the probability of individuals in the initial population being feasible.

- A MA that integrates the multi-LS and heuristic repairing technique with a standard GA is proposed. It is capable of generating some of the best-quality solutions for the benchmark RCPSPs used in the experimental study.
- An enhanced DE algorithm that incorporates improved DE operators and the proposed repairing method is presented. It has proved to generate good solutions across all standard benchmark instances.
- A new algorithm that utilizes the power of two EAs (GA and DE) is developed. An adaptive mechanism is used to emphasize the best-performing EAs, and heuristic repairing methods proposed to enhance individuals' feasibility in both the initial population and generated population in each iteration. This new algorithm improves solutions for RCPSPs in terms of both solution quality and computational time.

1.6 Organization of thesis

This thesis consists of the following six chapters.

- Chapter 1: Introduction
- Chapter 2: Literature Review
- Chapter 3: Genetic Algorithm for RCPSP
- Chapter 4: Differential Evolution for RCPSP
- Chapter 5: Bi-evolutionary Algorithm for RCPSP
- Chapter 6: Conclusions and Future Research Directions

In Chapter 1, an introduction to this research, which includes the background, motivation, objectives and highlights some of its scientific contributions, is presented.

Chapter 2 provides a review and analysis of the basic fundamentals of the topics covered in this thesis. Firstly, it introduces project scheduling and RCPSPs. Then, a survey and analysis of the different methodologies proposed in the literature for RCPSPs are discussed.

Finally, reviews of some exact, heuristic and meta-heuristic techniques, such as particle swarm optimization (PSO), GAs, DE, evolution strategy (ES) and evolutionary programming (EP), are presented.

In Chapter 3, descriptions of the PSPLIB benchmark problems, and the general framework of the proposed MA and its different components used in this study are provided. Detailed results obtained from the MA are then presented, along with comparisons of its performance with those of the branch and bound technique and state-of-the-art algorithms. The effects of its different components on its performance are also discussed.

Chapter 4 provides the general framework of the improved DE algorithm and its different components. The experimental results obtained by solving different sets of RCPSPs are reported and analyzed and then compared with those from the proposed MA and some state-of-the-art algorithms. The effects of different components on the performance of the proposed algorithm are also studied.

In Chapter 5, the general framework of bi-EA, that is, the ensemble of the proposed MA and DE is shown and its different components are explained. Then, the proposed algorithm is conducted and analyzed by using it to solve all sets of RCPSPs in the PSPLIB. The effects of its components are discussed and a comparison of its performance with those of proposed MA, DE and other state-of-the-art algorithms is provided.

Finally, Chapter 6 concludes the research of this thesis by summarizing its significant technical contributions in the domain of RCPSP research produced during this study and the major conclusions that can be drawn from the experiments conducted. Also, some conceivable directions for further research are also suggested.

Chapter 2

Literature Review

This chapter provides an overview of the basic fundamentals of the topics covered in this thesis. It begins with a brief description of project scheduling and its importance in real-world applications. Then, resource-constrained project scheduling problems (RCPSPs) are introduced and the efforts spent to solve them are reviewed. Also, descriptions of different exact, heuristic and meta-heuristic techniques are provided, followed by a detailed description of the genetic algorithm (GA) and differential evolution (DE) algorithm. Finally, a review of different hybrid algorithms applied to solve RCPSPs is presented.

2.1 Project Scheduling

Scheduling is one of the most common optimization problems which can be characterized as the allocation of resources to a set of activities restricted by a set of pre-defined constraints. In our daily life, the scheduling or allocation of activities is often complex and becomes extremely challenging when resources, such as time, budget and/or manpower, are limited. Effective scheduling is significant for different real-world problems and essential for the growth of activities in several fields, such as:

- production and project scheduling (De Carvalho and Haddad, 2012, Giffler and Thompson, 1960, Bierwirth and Mattfeld, 1999);
- robotic cell scheduling (Hall et al., 1998, Dawande et al., 2005);
- computer processor scheduling (Shan and Murphy, 1994, Błażewicz et al., 2013);
- timetabling (course and classroom scheduling) (Fang, 1994, Salman and Hamdan, 2012);

- personnel scheduling for assembly lines (Sabar et al., 2012, Brucker et al., 2011); and
- railway scheduling (Tian and Demeulemeester, 2013, Espinosa-Aranda et al., 2015).

However, constructing the optimal schedule is a very difficult task due to the highly constrained nature of scheduling problems. Moreover, due to the nature of problems, certain special constraints may be required in one particular instance and differ in another. Therefore, a general algorithm may not be suitable for all problems.

As RCPSPs are an important topic in both academic and practical fields, they are our focus in this thesis and described in more detail below.

2.2 Resource-constrained Project Scheduling

In a RCPSP, activities are characterized by their durations, resource utilization and relationships among their successor and predecessor activities (De Nijs, 2013).

In a typical RCPSP, the objective is to schedule all the activities in a project to minimize the total duration of the project (makespan) while satisfying the activities' precedence relationships and resource availability constraints. For a single project, let n be the number of activities to be scheduled, R_k the number of available resources of type k to be allocated, d_j the duration of the j activity and r_{jk} the number of resources (k) required by that activity.

In general, the activities in a project are represented by the set $P = \{a_0, a_1, \dots, a_n, a_{n+1}\}$, where activities a_0 and a_{n+1} are dummy ones used to indicate only the start and end of the project, respectively. Dummy activities have special values for their durations and resource usage, i.e., $d_0 = d_{n+1} = 0$ and $r_{0,k} = r_{n+1,k} = 0,\ \forall\, k \in K$. The set of non-dummy activities (actual activities) is represented by $A = \{a_1, \dots, a_n\}$, the set of resources by $0,1, \dots, r$ and PRE_j denotes the set of predecessor activities of any activity (j).

Generally, two types of resources are used by the activities in any project: *renewable resources*, such as manpower and machines, which are available at their full capacity in every period of time and can be used repeatedly as they are free for use again once an activity is finished; and *non-renewable resources,* such as a project's budget which, in contrast, have limited capacities and are available for use at only one time (Hartmann and Briskorn, 2010).

In traditional RCPSPs, the following assumptions are made (Krichen and Chaouachi, 2015):

- a single project consists of a number of activities with known durations;
- the precedence relationships among the activities are known;
- the starting time of an activity depends on the completion times of its preceding activities;
- renewable resources are available in limited quantities;
- the activities in progress cannot be interrupted and there is only one execution mode for each activity; and
- the objective is to minimize the project's duration.

In a RCPSP, a candidate solution is defined as *feasible* if it satisfies the following two main constraints.

1) *Precedence constraints* or *predecessor-successor relationships:* these are used to prevent each activity (j) from starting before the completion of its predecessors (PRE_j).
2) *Resource limitations:* the total resources allocated to all activities in a certain period of time must not exceed the limit for that period.

2.2.1 Mathematical Model of RCPSP

The mathematical model of a RCPSP is (Christofides et al., 1987, Kolisch and Hartmann, 1999):

$$Minimize \quad F_n \tag{2.1}$$

Subject to:

$$F_j \leq F_{j+1} - d_{j+1}, \qquad j = 1, \ldots, n \tag{2.2}$$

$$\sum_{j \in A(t)} r_{j.k} \leq R_k, \qquad k \in K\,; t \geq 0 \tag{2.3}$$

$$F_j \geq 0, \qquad j = 1, \ldots, n \tag{2.4}$$

In equation (2.1), the objective function, which aims to minimize the completion time of the entire project by reducing the finishing time of the last activity (F_n), is presented. The first constraint (2.2) ensures that none of the precedence constraints are violated and the second (2.3) that the amount of *non-renewable* resources (k) used by all activities does not exceed its availability (R_k) at any time (t), with $A(t)$ a set of ongoing activities at t. The last constraint ensures that the finishing times of all activities are non-negative.

An example of a RCPSP with 13 activities, including 11 executable ones, and their durations ($d_j \; \forall \; j = 0,1, \ldots, 12$) and resource requirements ($r_{j,k}$) is illustrated in Table 2.1.

P_j	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
d_j	0	1	4	2	4	2	4	1	1	3	1	1	0
$r_{j,k}$	0	5	4	1	3	2	4	1	1	4	1	5	0

Table 2.1: Example of project with 13 activities

Considering one resource (k = 1) with R_k = 5, the predecessor-successor relationships between activities in the project are presented as an activity-on-node graph in Figure 2.1. In Figure 2.2, the optimal schedule of the problem, that is, the best order of activities for minimizing the makespan of the project subject to the given constraints, is presented. In it, the x-axis represents the time, the y-axis the amount of resources utilized at any time and the number inside each box the activity number.

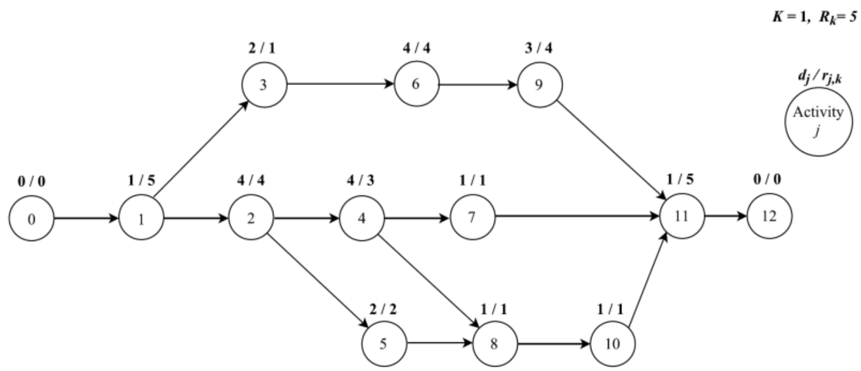


Figure 2.1: Activity-on-node graph

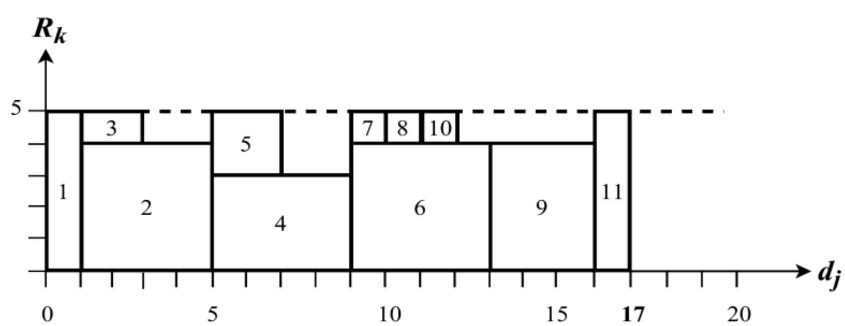


Figure 2.2: Optimal schedule of activities

2.2.2 Complexity of RCPSP

Many algorithms for solving scheduling problems have been introduced. Some have been capable of solving instances with thousands of activities/jobs; for example, the shortest processing time (SPT) priority rule is used to reduce the mean flow time, which is the total time required for a job to be finished, in a single-machine scheduling problem. However, small scheduling problems, such as coping with a few jobs and, sometimes, medium-sized ones, can only be solved using the best existing algorithms (Herroelen and Demeulemeester, 1994). Indeed, developing an applicable and profitable scheduling of activities can be an extremely difficult task for the following reasons.

Firstly, in terms of computational complexity, most scheduling problems can be placed in the class of computationally NP-hard problems which implies that there is an unknown general deterministic polynomial algorithm for solving them (Blazewicz et al., 1983). Moreover, a RCPSP is considered an intractable combinatorial problem according to the computational complexity theory (Garey and Johnson, 1979) which states that an optimization problem is NP-hard in a strong sense if its decision version is NP-complete. The decision variant of a RCPSP with a single resource and no precedence constraints have been proven to be NP-complete in strong cases by (Garey and Johnson, 1975).

Secondly, in real terms, every project has its own arrangement of scheduling constraints that need to be imposed. Also, it has its own interpretation of what is a feasible (applicable) and workable schedule which implies that an algorithm which is thought to be effective for one specific occurrence of a scheduling problem may not be suitable for others.

Due to the complexity of RCPSPs, the classical optimization-based approaches, such as integer programming with branch and bound (B&B) algorithms, are unable to solve large problems within a reasonable computational time. Therefore, heuristic algorithms have been essential for solving them. Although heuristic and meta-heuristic techniques produce solutions within a reasonable time limit, further research is required to improve their effectiveness and efficiency (Michalewicz, 1996, Widmer et al., 2010).

2.2.3 Solution approaches

As previously mentioned, RCPSPs belong to the class of NP-hard problems. Therefore, as a manual-based solution technique is inadequate for them, by all accounts, modified programming approaches are appealing alternatives.

A wide variety of strategies for comprehending RCPSPs has been proposed dating from 1959 when the first scheduling problem was introduced (Kelley Jr and Walker, 1959). There is no definitive means of sorting these strategies as, in the literature, the same technique is categorized in different groups. This thesis surveys the three classifications of strategies which deal with RCPSP problems, exact, heuristic and meta-heuristic methods.

Many exact methods for solving RCPSPs have been proposed. However, it can be concluded from the literature that they are applicable for only small project instances (Demeulemeester and Herroelen, 1992, Jalilvand et al., 2005). Since 1963, when serial and parallel schedule generation schemes (SGSs) were introduced (Kelley, 1963), a large number of heuristic approaches has been developed. Also, several meta-heuristic procedures, which are the last generation of heuristic methods, have been introduced during the last 20 years.

Traditional exact methods (i.e., linear and integer programming, B&B algorithms and dynamic programming) enable the finding of optimal solutions. However, in most cases, they are time consuming, particularly when solving large problems, such as ones with numerous dimensions, complicated constraints, multiple modes or uncertainty (Demeulemeester and Herroelen, 1992, Jalilvand et al., 2005, Patterson, 1984a).

Heuristic and meta-heuristic approaches are powerful and flexible search mechanisms that have successfully solved complex problems. Their algorithms aim to obtain good-quality solutions in reasonable computational times and are suitable for practical problems which often have large dimensions and very complex constraints (Das and Acharyya, 2011a, Kolisch and Hartmann, 1999).

In the following sections, an overview of exact methods for RCPSPs and a brief survey of heuristic ones, starting with SGSs which are basically used to construct feasible solutions, are discussed. Finally, detailed explanations of some meta-heuristic approaches, such as GAs and DE, which are the core of this research, are provided.

2.3 Exact Methods

Initially, simple models with exact methods that are guaranteed to find optimal solutions for small-scale problems were used to solve RCPSPs. However, as their computational complexity increased significantly, they typically became impractical for problems of significant sizes or with large sets of constraints. Some of these methods are described below.

2.3.1 CPM and PERT

The critical path method (CPM) (Kelley Jr and Walker, 1959) and program evaluation and review technique (PERT) (Malcolm et al., 1959) are considered two of the most effective and widely used techniques for solving scheduling problems. The main differences between them are discussed in the following.

The CPM is used in projects with expectable activities, such as construction ones and, when a trade-off is required, it allows project managers to choose which aspect of the project to reduce or increase. Moreover, it is a deterministic tool which provides estimates of the cost and time required to complete the project.

On the other hand, the PERT is used in projects that have activities of uncertain durations such as research/development ones. It employs three estimates of both the cost and time required to complete the project: the optimistic value (O), pessimistic value (P) and most likely value (M). It is a probabilistic tool that utilizes several estimates to determine the completion time of a project and manage the activities involved in order to complete them in a faster time and at a lower cost.

In general, the CPM is suitable for conventional projects with specific durations of activities while, for projects that require long periods of time to be completed and for which it is difficult to estimate the durations of their activities, such as research, the PERT is appropriate.

Both the CPM and PERT are committed to minimizing the makespan of a scheduled project based on two critical assumptions. The first is that the required resources are accessible in sufficient amounts and the second that the precedence constraints between any pair of activities, such as a_m and a_n, denote that activity a_m must be finished before activity a_n can start. Therefore, as the CPM and PERT provide only a resource-unconstrained schedule for a set of precedence-constrained activities with deterministic durations and give the shortest possible critical path time assuming that the resources are infinite, they provide only an approximate estimate of the difficulty of executing a schedule.

Over the years, the assumption of sufficiently accessible resources and particularly the strict precedence assumption of the CPM and PERT have been relaxed, with many research efforts coordinated towards project scheduling with explicit consideration of resource requirements and precedence constraints. Therefore, dynamic variations (Blazewicz et al., 1983) and stochastic variations (Neumann, 1990) have also been developed in an attempt to make the CPM and PERT inclusive/natural of assumptions similar to reality, with strategies incorporating probabilistic evaluations of activity durations.

2.3.2 Integer and linear programming-based methods

Integer programming and linear programming (IP/LP) are complete mathematical methods which are mainly used separately without including other approaches (Garfinkel and Nemhauser, 1972). Using the traditional IP/LP structure, early research concentrated mainly on formulating scheduling problems and solving them as mathematical, mostly 0–1 IP/LP, problems.

IP/LP-based methods which use the IP formulation originally proposed by Pritsker et al. (1969) have also been used by OğUz and Bala (1994). Patterson (1984b) presented an overview of optimal solution methods for project scheduling and Berthold et al. (Berthold et al., 2010, Koné et al., 2013, Damay et al., 2007) the generalization of two existing mixed-integer LP models for the classical RCPSP as well as a novel formulation based on the concept of an event.

Although there are many different models for IP/LP, they are conceptually similar. In them, a large number of binary variables is necessary, with this number growing very quickly for large problems which renders them impractical for those of realistic sizes.

In general, exact methods rely on attributes of the objective function and specific constraint formulations. As Davis (1991) noted, many of the constraints commonly found in real scheduling problems do not lend themselves well to traditional operations research or mathematical programming techniques. Also, as LP formulations typically do not scale well, they can be used for only specific instances or small problems.

In the dynamic programming approach described by Held and Karp (1962), an optimal schedule is progressively developed by constructing one for any two tasks and then extending it by adding tasks until all the tasks are scheduled.

2.3.3 Branch and bound (B&B) algorithms

B&B algorithms are used to explore the search space by constructing a search tree in which each node is either a branch or leaf. When a leaf node is reached, feasible solutions may be found and this node cannot be partitioned or expanded. The search space of a branch node is divided into subsets according to calculations of its lower and upper bounds, and sometimes time-bound adjustments, which is called branching. B&B representations of tree and search spaces are illustrated in Figures 2.3 and 2.4, respectively.

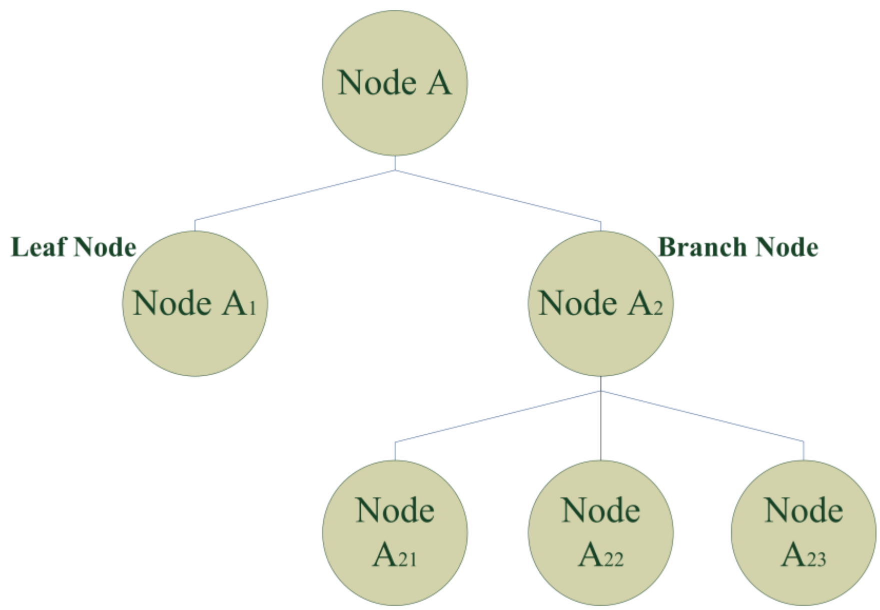


Figure 2.3: Representation of B&B tree space

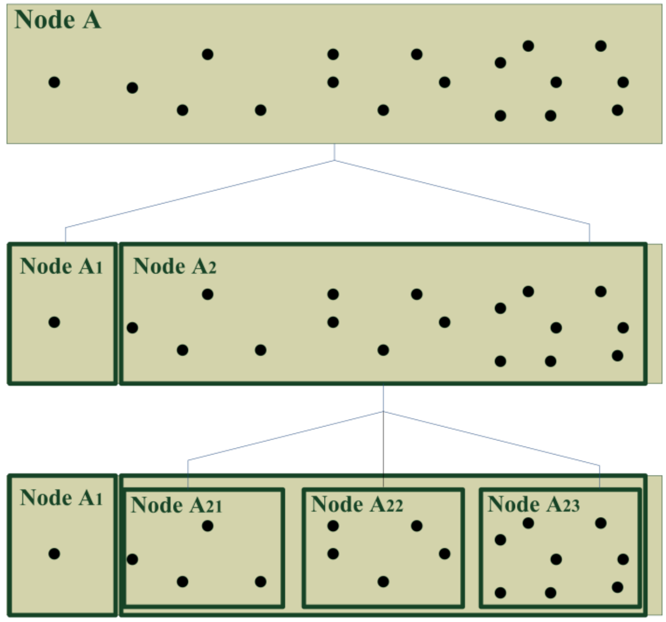


Figure 2.4: Representation of B&B search space

The B&B-based algorithms introduced in Demeulemeester and Herroelen (1992) and Jalilvand et al. (2005) were able to find optimal solutions but their computational complexity increased significantly with increasing numbers of activities. Conversely, there are algorithms that can find good solutions for a problem in a reasonable time, such as priority scheduling (Li et al., 1997) and greedy-based (Lupetti and Zagorodnov, 2006) algorithms, but their shortcoming is their inability to satisfy all constraints. Cheng and Wu (2006) constructed a project scheduling model which includes a time constraint and presented a hybrid algorithm combining a B&B procedure and heuristic. Their simulation results demonstrated that the optimization effect of this algorithm was better than those of other algorithms.

2.4 Heuristics

The term heuristic is used for methods which find solutions from among all possible ones (Gigerenzer and Gaissmaier, 2011) but offer no guarantee of finding the optimal solution. Usually, as heuristics are capable of providing a near-optimal solution, they can be considered approximate algorithms. Typically, they are computationally efficient and require much less time and, in many cases, less space than exact methods.

There are the following two broad categories of search-based heuristics.

1. *Constructive heuristics (single-pass):* in such methods, priorities are assigned to the activities or tasks which are ordered and then scheduled sequentially. These priority values can be assigned statically prior to scheduling or adjusted dynamically during the scheduling procedure (Palpant et al., 2004). Also, different heuristics can be combined in the hope of achieving better performances.

2. *Improvement heuristics (multi-pass):* in these techniques, a heuristic is applied repeatedly until no further improvement is possible. *Improvement algorithms* are heuristics that generally start with a feasible solution and repeatedly try to achieve a better one (Agarwal et al., 2006).

In order to construct a feasible solution, Kelley (1963) proposed a SGS which encouraged several research efforts to introduce many heuristics.

2.4.1 Schedule generation scheme (SGS)

Both serial and parallel SGSs are used to generate a feasible schedule by expanding a *partial schedule* (one with a subset of activities assigned a finishing time). In each stage, the generation scheme forms all activities into two sets, those to be scheduled (a decision set) and those already scheduled (a scheduled set). Subsequently, one or more activities are chosen from the decision set to be scheduled. Both these schemes for project scheduling with minimum time lags have been discussed by Kolisch (1996b) and Brucker et al. (1999). They reported that the serial method has been shown to be better than the parallel one in terms of performance. Both serial and parallel SGS are briefly described below.

2.4.1.1 Serial schedule generation (SSG)

SSG was proposed by Kelley (1963), as cited by Kolisch (1996b). It consists of $e = 1, \dots, Y$ stages with one activity selected and scheduled in each stage in which there are two separate sets, the scheduled set (S_e) and decision set (D_e). The former set contains the activities already scheduled and, thus, belonging to the partial schedule, and the latter the unscheduled activities with every predecessor in the scheduled set. In each stage, one activity from the decision set is selected and scheduled at its earliest precedence and resource-feasible starting time. Then, it is placed in the scheduled set (S_e) and removed from the decision set (D_e). It is also possible that a number of activities move in parallel from the D_e to S_e set since all their predecessors are now scheduled. The algorithm terminates after the final stage equal Y, when all the activities are in the scheduled set (S_e).

2.4.1.2 Parallel schedule generation (PSG)

A PSG scheme iterates over the scheduled time (t_e) of a project instead of selecting activities one-by-one and scheduling them as soon as possible. In each iteration, the activities eligible to be scheduled are added to the scheduled activities set providing sufficient resources are available. For clarification, at each time point (t_e), this scheme selects activities which are eligible to be scheduled and, then according to the priority list, it assigns them scheduling sequences. Then, if there is no resource conflict, the selected activities are scheduled with starting times equal to the first time point (t_1). At the next time point (t_2), which is equal to the earliest finishing time of all the currently active activities, the activities not eligible to be scheduled due to a resource conflict at t_1 become eligible and the process is repeated till all activities are scheduled.

2.4.2 Priority rule-based scheduling methods

The first heuristic methods for scheduling were based on priority rules (Kelley, 1963, Brucker et al., 1999), several of which have been introduced, experimentally tested and compared in terms of their effectiveness relative to one another and an optimal solution (Boctor, 1990, Davis and Patterson, 1975, Patterson, 1976, Thesen, 1976). Because of their easy implementation and low time complexity, in practice, priority-based heuristics are the most widely applied for solving scheduling problems. However, the main problem is finding an efficient priority rule. The common basis of all priority rule-based heuristics can be gathered from the algorithms of (Giffler and Thompson, 1960) and (Storer et al., 1992).

An overall discussion and summary of priority-dispatching rules are provided in Panwalkar and Iskander (1977), Blackstone et al. (1982) and Haupt (1989).

In order to construct a priority rule-based algorithm, a combination of priority rules and SGSs is required. The resultant heuristic method one of two types, that is, single or multiple pass.

Single pass: this method generates a single schedule and, during this process, employs one SGS and one priority rule to produce a single feasible solution. Several examples of such methods can be found in (Thesen, 1976, Whitehouse and Brown, 1979, Lawrence, 1985).

Multiple pass: these techniques generate more than one schedule, with combinations of priority rules and SGSs possibly occurring in several scenarios. The most common are the multi-priority rule, forward-backward scheduling and sampling.

In the *multi-priority rule*, a SGS is used many times with different priority rules each time. Kurtulus and Narula (1985) applied 10 different scheduling rules in order to measure their performances. Kolisch (1996a) introduced an improved RSM (resource scheduling method) priority rule and developed two new priority rules which extended the precedence-based minimum slack priority rule (MSLK) to precedence- and resource-based slack priority rules. Also, Boctor (1990) employed seven different scheduling rules in his suggested multi-heuristic procedures using both parallel and serial rules.

Forward-backward scheduling is an iterative scheduling technique aimed at minimizing the project duration by reducing the project resources, an idea introduced by Li and Willis (1992). In it, one SGS is applied iteratively to schedule the project by switching between forward and backward scheduling. Applications of such methods can be found in Özdamar and Ulusoy (1996).

The *sampling* approach employs one SGS and one priority rule is selected randomly according to a computed selection probability, with the bias in the selection of the priority rules generating different schedules. Kolisch (1996b) distinguished among random sampling, biased random sampling and regret-based biased random sampling based on the method used to compute the selection probability.

2.4.3 Neighborhood search (NS)

NS or local search algorithms belong to a broad class of improvement algorithms. The NS is a technique aimed at finding a good or near-optimal solution starting from an initial given point in the solution space. It repeatedly tries to improve the current point by looking for better ones within the neighborhood points of the current solution/point. Once a better solution is found, it is used as the new starting point, with this process repeated until no better solution than the current one can be found. Then, the current solution/point is adopted as the best solution (Fleszar and Hindi, 2004).

The large-scale NS is an algorithm for use in large spaces that include more neighborhoods. In their survey, Ahuja et al. (2002) reported that, although the quality of the locally optimal solutions and accuracy of the final solutions were improved by using a larger-sized neighborhood, it took longer to search such a neighborhood in each generation.

2.5 Meta-heuristic Methods

Meta-heuristic methods commonly begin with random solutions and no assumptions about the problem being optimized, and can search very large spaces of candidate solutions.

Blum and Roli (2003) summarized the basic characteristic of meta-heuristics as strategies that guide the search process with the aim of efficiently exploring it to find optimal or near-optimal solutions. Meta-heuristic-based algorithms range from simple local search methods to complex learning processes and may integrate procedures to avoid becoming trapped in local solutions in the search space.

As meta-heuristics can be easily adapted to a particular problem or problem class with much less effort than heuristics, they are an appealing choice for implementation in general-purpose software (Ólafsson, 2006). Also, a good meta-heuristic design is likely to obtain near-optimal solutions in reasonable computation times (Ólafsson, 2006).

However, the many drawbacks of using meta-heuristics can be summarized as (Beheshti and Shamsuddin, 2013):

- becoming trapping in local optima;
- requiring long computational times;
- having slow convergence speeds;
- needing to tune multiple search parameters;
- consisting of difficult encoding schemes; and
- providing no guarantee that the best solution found will be the optimal one.

Therefore, improving the performances of existing meta-heuristics or even proposing new ones seems to be an important research task.

According to (Blum and Roli, 2003), meta-heuristics can be divided into trajectory and population-based methods. Examples of trajectory methods are simulated annealing (SA) (Cho and Kim, 1997, Das and Acharyya, 2011b) and the tabu search (TS) (Lee and Kim, 1996), and of population-based ones, evolutionary algorithms (EA) such as a GA (Toklu, 2002) and DE (Damak et al., 2009), and swarm intelligent algorithms such as ant-colony optimization (ACO) (Dorigo, 1992) and particle swarm optimization (PSO) (Kennedy, 2010). In the following sub-sections, the above mentioned algorithms are described.

2.5.1 Trajectory methods

The term *trajectory* is used for methods that work on a single solution at any time (not a population of solutions) and adopt local search-based meta-heuristics. In such an approach, the algorithm starts from an initial point/solution and, as the search process it follows can be described by a trajectory in the search space (Blum and Roli, 2003), the next better solution may or may not be one of the current solution neighborhoods.

2.5.1.1 Simulated annealing (SA)

The SA algorithm was initially inspired by annealing in the minerals industry which encompasses two processes: heating a metal to modify its physical properties by changing its internal structure; and cooling it to fix its new structure. In SA, the heating process is simulated by a variable temperature (t) initially set to be high and then progressively reduced. As, in reality, when t is high, the algorithm can accept any new solution even if it is worse than the current one (change the physical properties), hence its early trapping in any local optimum is avoided and then as it runs, both t and the chances of accepting worse solutions are reduced. Using this mechanism, the algorithm begins with large capabilities to explore the entire search space and then focuses on the exploitation process in the final phases, with the aim of effectively finding optimal or near-optimal solutions, especially when dealing with large instances (Aerts and Heuvelink, 2002).

Boctor (1996) solved non-pre-emptive RCPSPs using a new adaptation of SA in which the initial solution was obtained using a heuristic scheduling technique. Then, SA was applied with reheating and a variable cooling rate which protected the algorithm from becoming stuck in a local optimum solution by intensifying the search process in the neighborhood of this optimum. This algorithm was able to handle single- and multi-modal instances and optimize multiple objective functions. Cho and Kim (1997) proposed a SA using priority rules in which a solution was represented by a vector of numbers called a priority list with each number denoting the priority of each activity to be scheduled. Then, a priority scheduling method was applied to construct a schedule using the given priority list. This algorithm allowed some activities to be delayed with the aim of extending the search space so that solutions could be further improved. The results demonstrated that it achieved good performances compared with those of some other heuristic techniques.

Józefowska et al. (2001) used a SA approach to solve multi-modal RCPSPs. Their algorithm was implemented based on a precedence-feasible list of activities and modal assignment. They considered SA with and without a penalty function and three different neighborhood generation mechanisms applied to both versions. According to their results, the version of SA with the penalty function performed better and, moreover, the proposed algorithm showed an improved performance for large problems.

2.5.1.2 Tabu search (TS)

Classical local search methods set a candidate solution for a problem and begin to explore its neighbors to find better ones and, in most cases, become trapped in a local optimum solution. The idea of the TS was initiated by (Glover, 1989, Glover, 1990a) as a technique for solving combinatorial optimization problems (Icmeli and Erenguc, 1994) which could improve the performances of local search methods by directing the search away from local optima through accepting worse solutions if no improved ones were available.

A TS starts with an initial solution which may be feasible or infeasible. Then, its neighboring points are explored using a suitable local search that aims to find a better solution, after which the previously best solution moves from the current solution to its better neighbor solution. The movement from one point/solution to another is retained in some sort of tabu status and, if no improvement is achieved, which means that the search is stuck in a local solution, TS does not allow movement back to the visited points for a certain number of iterations. Therefore, it avoids cycling but, if a better solution is found, the best solution will move to such a point regardless of its tabu status. This cycle continuing until some pre-defined stopping criteria are satisfied (Glover, 1990b). Thomas and Salhi (1998) introduced an enhanced TS technique that uses well-defined move strategies and a structured neighborhood search while defining a suitable tabu status.

2.5.2 Population-based methods

Population-based or evolutionary computation (EC) methods have been utilized to solve complex optimization problems. In them, a population of solutions rather than a single vector of decision variables is used. As they are direct search methods which have no assumptions about the problem being optimized and can search very large spaces of candidate solutions, they appear to be effective tools for the optimization of management schedules (Pukkala, 2009).

2.5.2.1 Swarm intelligence (SI) algorithms

The expression SI was introduced by Beni and Wang (1993) in the context of cellular robotic systems. It can be described as the collective behavior emerging from the processes of social insects acting subject to very few rules, with self-organization the main feature for agents interacting within limitations (Kennedy et al., 2001). In other words, a SI system consists of a population of agents interacting locally with one another and their environment. Several examples of SI were inspired by the world of animals, such as flocks of birds, schools of fish and colonies of ants. Since more information is gathered from the whole swarm by social interactions among the agents, the environment or search space can be explored more efficiently. Ant colony optimization (ACO) and PSO, which are examples of IS algorithms, are discussed in the following sub-sections.

A. Ant colony optimization (ACO)

ACO was introduced by Dorigo (1992) and Dorigo et al. (1996) as a novel nature-inspired meta-heuristic for solving hard combinatorial optimization problems. As it is an approximate algorithm used to obtain good solutions to complicated optimization problems in a reasonable amount of computational time, it is included in the class of meta-heuristics (Blum and Roli, 2003).

ACO simulates the behavior of ants in the search for sustenance exhibited when they attempt to find the shortest paths between their nests and food sources (Deneubourg et al., 1990). Initially, they walk randomly along different paths looking for good food sources (solutions) and a special substance called *pheromone* produced by the explorer ant is deposited on the paths explored to guide others. The concentration of *pheromone* indicates the direction to be taken; the stronger the concentration, the higher the probability that the ants will follow this path. The framework of a basic ACO algorithm is given in (Dorigo and Blum, 2005).

ACO is an iterative distributed algorithm. In each generation, a set of artificial ants/agents constructs solutions by walking from vertex to vertex, with each agent not allowed to revisit any vertex during its walk. An ant selects an initial solution (i) and then chooses the successor vertex (v) to be visited according to a stochastic technique built on the basis of the pheromone. If v has not been visited before, the probability of selecting v depends on the pheromone associated with edge (i,v) which is the path between i and v. To improve the quality of solutions produced by ACO over iterations, at the end of each iteration, the pheromone values are altered based on the quality of solutions constructed so far to bias the ants towards constructing similar solutions to the best ones previously constructed in future iterations (Dorigo et al., 1999, Yaseen and Al-Slamy, 2008).

An ACO-based methodology for solving a multi-modal RCPSP (MRCPSP) was introduced by Zhang (2011). They proposed two levels of pheromones for each ant in terms of the sequence and modal selection of activities for directing the algorithm's search process. Based on their conclusions, this algorithm was an effective alternative methodology for solving a MRCPSP.

B. Particle swarm optimization (PSO)

PSO was inspired by the natural movements of a flock of birds and school of fish. It is widely used to solve optimization problems because of its easy implementation and good progress towards optimality (Bai, 2010).

A PSO algorithm starts with a population of candidate solutions called particles. Each particle evaluates the objective function at its current location in the search space and searches for better solutions. The movement of each particle in the swarm through the search space is determined according to its best position (the history of its own current and best locations) or the swarm's best position. The next iteration takes place after all particles have been moved. Eventually, the swarm as a whole, like a flock of birds collectively foraging for food, is likely to move close to an optimum value of the fitness function (Bai, 2010, Poli et al., 2007).

Chen et al. (2010) proposed two rules called the 'delay local search rule' and 'bidirectional scheduling rule' for PSO to solve scheduling problems. The former enables some activities to be delayed by altering the previously determined start of the processing time and is also capable of escaping from local solutions. The latter combines forward and backward scheduling to expand the search area in the solution space to obtain a potential optimal solution. To speed up the production of the feasible solution, in that study, a critical path was adopted and, based on the results obtained, the algorithm was efficient.

2.5.2.2 Evolutionary Algorithms (EAs)

EAs were inspired by the biological model of evolution and natural selection initiated by Darwin (1859) and have a long history of successfully solving RCPSPs. In the next sub-sections, EAs, such as GAs and DE which are later used as basic techniques in this thesis, are discussed.

The basic outlines of all EAs are broadly similar in that they iteratively evolve a population of candidate solutions over several generations, but have some variations in the order of their evolutionary operations and ways of generating an initial population of individuals. Many encoding types, such as real-value, integer and string, can be used to represent each solution. Each individual is evaluated by a pre-defined fitness function which determines how close it is to the desired value (fitness value), with the selection strategy always favoring solutions with higher fitness values. Then, the concept of natural selection in biology is mimicked by allowing some individuals to survive from generation to generation according to their fitness values. New candidates (offspring) are reproduced by performing recombination (crossover) and/or mutation operations. In a recombination operator, two or more selected parents produce one or more offspring. In contrast, a mutation operator is applied to one individual by changing a single element in order to generate a modified one in the hope of maintaining diversity. Over time, the quality of the solutions/individuals in the population should be improved. Finally, the evolution can be terminated once the algorithm has found a solution that is sufficiently good.

A. Genetic algorithms (GAs)

A GA was first introduced by Holland (1975) and Goldberg et al. (1989) developed it as a computational approach for solving hard problems. It mimics the principles of biological evolution such that a population of candidate solutions (called individuals or phenotypes) to an optimization problem is evolved towards better solutions by its set of properties or genes (its chromosomes or genotype) being mutated and altered. As the search for better solutions in a GA-based approach is largely independent of context, it can be readily applied across a variety of situations.

```
Generate_initial_population
Evaluate_Population()
While not (terminating condition) do
    Selection_population()
    Cross_population()
    Mutate_population()
    Evaluate_population()
End While
```

Figure 2.5: General procedure for GA

An implementation of this algorithm (Figure 2.5) begins with a random population of chromosomes, each of which represents a possible solution, from which a selection operator chooses two or more from a generation by comparing their fitness values. A crossover operator replaces the subsequence before a pre-determined position (usually selected randomly) by that after the crossover point between two parent chromosomes to produce the offspring depending on the value of the crossover probability (p_c). Then, a mutation operator selects a random position of this chromosome and randomly modifies its value to a new one depending on the value of the mutation probability (p_m) which helps to avoid local minima as it tries to enable new regions in the search space to be explored.

In this section, a description of different representations of solutions, and crossover and mutation operators for a GA are given.

i. Genetic representations

To solve a problem using a GA, candidate solutions must be encoded in an appropriate form and, traditionally, are represented by a binary array of bits called '*chromosomes*'. Arrays of other types and structures can be utilized in the same way as different types of problems require different genetic representations/encoding, such as binary, permutation and real-value encodings introduced to represent individuals by Ronald (1997).

In *binary representation*, every chromosome is a string of bits of 0 or 1. This encoding offers many possible solutions/chromosomes and has been found to be an efficient search technique which avoids local optimum solutions (Haupt and Haupt, 2004). However, it is often not suitable for many problems, sometimes corrections must be made after crossover and/or mutation and also its computational cost is usually higher than those of deterministic optimization techniques (Haupt and Haupt, 2004); for example, Whitley et al. (1989) confirmed that binary representation was not considered very suitable for the traveling salesman problem (TSP).

In *permutation representation*, every chromosome is a string of numbers representing numbers in a sequence. Although its encoding may be the most natural way of representing activity/task sequences, not all permutations of tasks in a project represent feasible schedules because of the existence of precedence constraints among the tasks (Golmakani and Namazi, 2012). It is only useful for ordering problems, such as the TSP or task ordering (Mohebifar, 2006).

In *real-value representation*, every chromosome is represented as a string of some real values and it is usually used for problems in continuous domains. Haupt and Haupt (1998) mentioned that real-number representation in a GA is more convenient with other optimization algorithms so that they can be easily hybridized or combined.

ii. Selection

As a GA can explore a large search space, which is containing feasible solutions, strong individuals within the population are selected to survive longer than weak ones. The strength of an individual (strong/weak) is determined according to its fitness value which is calculated using a pre-defined fitness function. Based on this, weak individuals are excluded and the fittest selected to reproduce themselves by using crossover and mutation operators. Several selection methods/operators, such as tournament, roulette wheel, rank selection and elitism, are considered (Chudasama et al., 2011, Sarker et al., 2003).

In *tournament selection*, an individual is selected from the population by running several ‘competitions’ among a few individuals randomly chosen from the population, as the individuals with the best fitness values (the winners of each competition) selected for crossover (Blickle and Thiele, 1995b). In this method, the selection pressure can easily be adjusted by changing the tournament size ($TSize$) (i.e., if $TSize$ is large, weak individuals have fewer chances of being selected) (Miller et al., 1995).

In *roulette wheel selection*, also called fitness proportionate selection, individuals are selected according to their fitness values and, as in nature, strong ones have more opportunities to survive than weak ones. This process is repeated until the desired number of individuals is obtained in a so-called mating population (Baker, 1987).

In *rank selection*, individuals are sorted based on their fitness values, with 1 assigned to the worst and M to the best individuals, and selection probabilities assigned to them according to their rankings which, in linear ranking selection, are linearly assigned (Blickle and Thiele, 1995a).

In *elitism*, a small proportion of the fittest candidates is copied, without any changes, into the next generation. Using this method, a GA does not waste time re-searching previously explored partial solutions which, in turn, would affect its performance. Candidate solutions that are kept unchanged for the next generation can be selected as parents to produce offspring through the reproduction operators in the next generation (Ahn and Ramakrishna, 2003, Chudasama et al., 2011).

iii. Crossover

Crossover is considered the key alteration operator in a GA's evolutionary process. It merges the genetic information of two individuals previously selected as parents to create new offspring. Several crossover operators have been proposed during the last few decades, with single-point, two-point (multi-point) and uniform (Michalewicz, 1992, Magalhães-Mendes, 2013) more appropriate for discrete problems and heuristic (Wright, 1991), flat (Radcliffe, 1991), simulated binary (SBX) (Agrawal et al., 1995), simplex (SPX) (Tsutsui et al., 1999), parent-centric (PCX) (Deb et al., 2002) and triangular crossovers (Elfeky et al., 2008) for continuous problems. Each operator has its advantages and disadvantages when applied to evolutionary problems.

In *single-point crossover*, two individuals are randomly selected as parents, a crossover point chosen uniformly at random between 1 and the chromosome length, and two new chromosomes created for the two parents. This crossover point divides each individual into two sub-schedules (left and right) and then the right (or left) sub-schedules of the two individuals are swapped; for example, considering two parents ($P1$ and $P2$), $OS1$ and $OS2$ are the offspring produced and the crossover point occurs after the sixth bit as shown in Figure 2.6.

In *two-point crossover*, a swapping process similar to that of single-point crossover occurs except that two crossover points ($CP1$ and $CP2$) instead of one are randomly selected (Figure 2.7).

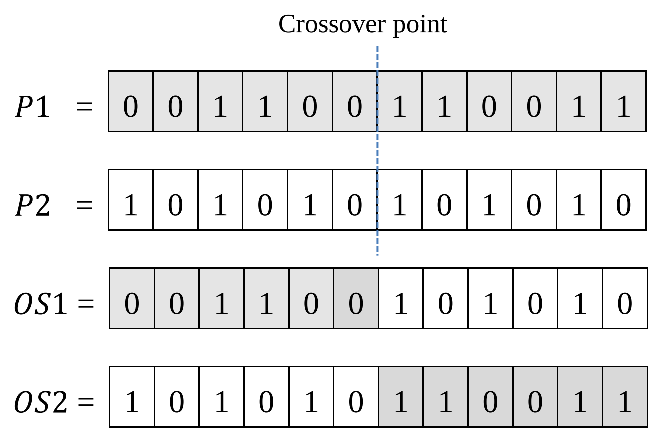


Figure 2.6: Single-point crossover

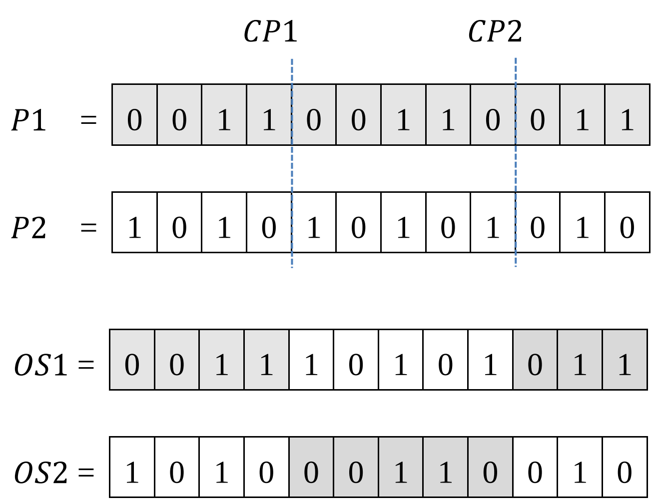


Figure 2.7: Two-point crossover

In *uniform crossover*, a vector of random numbers is generated and each bit/gene value of the first parent ($P1$) assigned to the first offspring ($OS1$) if the corresponding random number of each gene (R_n) is less than the crossover probability (p_c) or otherwise to the second offspring ($OS2$). In the following example, p_c is equal to 0.4.

$\boldsymbol{P1}$	0	0	1	1	0	0	1	1	0	0	1	1
$\boldsymbol{P2}$	1	0	1	0	1	0	1	0	1	0	1	0
$\boldsymbol{R_n}$	0.2	0.7	0.3	0.5	0.9	0.8	0.1	0.3	0.2	0.6	0.1	0.7
$\boldsymbol{OS1}$	0	0	1	0	1	0	1	1	0	0	1	0
$\boldsymbol{OS2}$	1	0	1	1	0	0	1	0	1	0	1	1

Figure 2.8: Uniform crossover

iv. Mutation

Mutation is considered a key operator that increases the diversity of the population and enables GAs to explore promising areas of the search space (Korejo et al., 2010). As it is a genetic operator, it alters a few random bits of a chromosome/individual and maintains genetic diversity, i.e., variations in a population’s gene pool from one generation to another, with the aim of preventing convergence towards a local optimum. Normally, after offspring are produced by recombination/crossover, mutation is applied to their variables according to a low probability called the mutation rate. Several types of mutation, such as flip bit, boundary, uniform and non-uniform, have been introduced.

In *flip bit mutation*, randomly selected bits are changed (or flipped). Considering a modified parent (produced from crossover), the flipping of a bit involves inverting 0 to 1 and 1 to 0 (Sivanandam and Deepa, 2007), a bit string mutation commonly used in binary encoding.

In *boundary mutation*, randomly selected bits are replaced by either their lower or upper (randomly chosen) bounds, a mutation that can be used for floating and integer bits.

In *uniform mutation*, the value of the selected bit is replaced by a uniform random value between its upper and lower bounds pre-assigned for that bit or gene. It is used for integer and floating bits.

Non-uniform mutation was proposed by Michalewicz (1996) with the aim of reducing the disadvantages of random mutation in a real-coded GA. By applying a non-uniform mutation operator, the probability that the amount of mutation will be close to 0 in the next generation is increased. It performs well for problems for which a solution only needs to be refined during the later stages of an algorithm's execution.

B. Differential evolution (DE)

Storn and Price (1997) proposed DE. In EC, DE is a method that optimizes a problem by iteratively trying to improve a candidate solution. The quality of which can be measured by a given fitness function. DE is a stochastic population-based search technique which uses mutation, crossover and selection operators to guide the search to find (near-) optimal solutions and, among existing EAs, is considered a powerful tool for solving optimization problems. In it, an initial population with a pre-determined size (PS) is generated and then each individual ($\vec{x}_i$), which consists of n variables, is evolved using the three evolutionary operators.

The three major operations used in the iteration phase, mutation, crossover and selection, are discussed in the following paragraphs.

i. Mutation operation

For each target vector ($x_{i,g}$), a mutant vector is generated which, in its simplest form (Storn and Price, 1995) is calculated as:

$$\vec{v}_{i,g+1} = \vec{x}_{r_1,g} + F \times \left(\vec{x}_{r_2,g} - \vec{x}_{r_3,g}\right) \quad , r_1 \neq r_2 \neq r_3 \neq i \tag{2.5}$$

where F is the mutation parameter, or weighting factor, that controls the amplification of the differential variation $(\vec{x}_{r_2,g} - \vec{x}_{r_3,g})$ and generally lies within the range of [0, 2] (Chandra and Chattopadhyay, 2012), and $\vec{x}_{r_1,g}$, $\vec{x}_{r_2,g}$, $\vec{x}_{r_3,g}$ three randomly chosen vectors which are not equal to each other or the target vector $(\vec{x}_{i,g})$.

This operation enables DE to explore the search space and maintain diversity. Depending on the way in which the mutant vectors are generated from the target ones, there may be different variations of DE mutation strategies.

The mutant vector of any generation can be produced by incorporating the best target vector of that generation. This DE mutation scheme can be applied through DE/best/1 and DE/best/2 strategies, the only difference between which is the number of target vectors used to generate the mutant vector, as shown in equations (2.6) and (2.7), respectively (Das and Suganthan, 2011, Storn, 1996).

$$\vec{v}_{i,g+1} = \vec{x}_{best,g} + F \times (\vec{x}_{r_1,g} - \vec{x}_{r_2,g}) \tag{2.6}$$

$$\vec{v}_{i,g+1} = \vec{x}_{best,g} + F \times \left(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}\right) + F \times (\vec{x}_{r_3,g} - \vec{x}_{r_4,g}) \tag{2.7}$$

where $\vec{x}_{best,g}$ is the best individual vector in generation g.

The DE/current-to-best/1 scheme involves another control parameter (λ) in addition to the weighting factor and includes the best target vector of the current generation as well as two different target vectors and the current one $(\vec{x}_{i,g})$ with the aim of generating a mutant vector for the next generation $(\vec{v}_{i,g+1})$ according to (Das et al., 2008):

$$\vec{v}_{i,g+1} = \vec{x}_{i,g} + \lambda \times \left(\vec{x}_{best,g} - \vec{x}_{i,g}\right) + F \times \left(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}\right) \tag{2.8}$$

In DE/rand-to-best/1 (Price et al., 2005), the best and current vectors as well as two randomly selected target vectors are used to generate the mutant vector according to:

$$\vec{v}_{i,g+1} = \vec{x}_{r_1,g} + F \times \left(\vec{x}_{best,g} - \vec{x}_{i,g}\right) + F \times \left(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}\right) \tag{2.9}$$

ii. Crossover operation

In a crossover operation, trail vectors are generated by combining the target and mutant vectors (offspring) according to a pre-defined possibility. Binomial and exponential crossovers are the two most well-known types of crossover in the literature.

In a *binomial crossover operation*, the trial vectors are generated according to:

$$u_{i,g+1}^{j} = \begin{cases} v_{i,g+1}^{j} & rand(j) \le CR \ or \ j = a_j \\ x_{i,g}^{j} & otherwise \end{cases} \tag{2.10}$$

where $j = 1,2,\dots,n$; $i = 1,2,\dots,PS$; CR is the crossover possibility in the range of [0,1], $rand(j)$ the jth evaluation of a uniform random number generator within [0,1] and a_j a randomly selected dimension to ensure that at least one element of $u_{i,g+1}$ is chosen from the mutant vectors (Storn and Price, 1997, Price et al., 2006).

An *exponential crossover operation* acts like a two-point crossover in which the first ($CP1$) and second ($CP2$) cut points randomly selected from $\{1,\dots,n\}$ and $\{1,\dots,L\}$, respectively, where L denote the number of consecutive components (counted in a circular manner) taken from the mutant vector. In this strategy, trial vectors are generated according to:

$$u_{i,g+1}^{j} = \begin{cases} v_{i,g+1}^{j} & j = mod(CP1 - CP2 + 1, n) \\ x_{i,g}^{j} & otherwise \end{cases} \tag{2.11}$$

iii. Selection operation

In a selection operation, a comparison of each trial vector and its corresponding target vector is used to determine whether either the trail or target vectors can survive into the next generation ($g + 1$) by adopting the greedy selection strategy, the formula for which is:

$$x_{i,g+1} = \begin{cases} u_{i,g+1}, & f(u_{i,g+1}) < f(x_{i,g}) \\ x_{i,g}, & otherwise \end{cases} \tag{2.12}$$

As the processes of the evolutionary operations (i.e., mutation, crossover and selection) continue as long as the averaged cost function is more than a pre-defined cost value, termination of the DE technique can be determined by the maximum allowable value of the averaged cost function (the desired value).

C. Evolutionary strategy

The evolutionary strategy (ES) was initiated by Rechenberg (1984) and Schwefel (1993) as an optimization technique. Similar to all EAs (e.g., GAs), it tries to evolve better solutions through crossover (recombination), mutation and survival of the fittest. It encodes solutions/parameters as floating point numbers and uses arithmetic operators to modify them whereas a GAs encodes parameters as bit strings and uses logical operators to manipulate them. Therefore, ES is considered an effective tool for optimizing continuous functions while, in contrast, GAs are more appropriate for combinatorial optimization (Zhang et al., 2005a, Dianati et al., 2002).

In the process of ES, the algorithm begins with a population of PS vectors as parents, with an offspring (or child) population of λ vectors, where $\lambda \geq PS$, produced by recombining randomly selected parent vectors. According to Bäck et al. (1997), there are two types of recombination: (1) discrete, in which some offspring genes are from one parent and the rest from the other; and (2) intermediate, which is the average of the genes of both parents. After applying the recombination process to the parents, the individuals produced are then mutated by altering a randomly chosen single bit/gene. After all λ offspring are mutated and evaluated, a greedy selection to determine the individuals to survive to the next generation can be performed in two ways: (1) in (PS, λ)-ES, the best PS children are selected to be parents in the next generation; and (2) in $(PS + \lambda)$-ES, the best PS children selected are chosen from a combination of the parent and offspring populations.

Beyer and Schwefel (2002) developed two general rules for designing and evaluating ES experiments. Firstly, performing slight modifications to all variables at one time randomly which sounds similar to mutation. Secondly, preserving the offspring produced as the new set of variables/solutions after the evolutionary process, in case the quality of solutions is improved or at least remains the same, or otherwise returning these variables to their old status which is typically applied in ES as survival of the fittest.

D. Evolutionary programming (EP)

EP is a stochastic optimization strategy originally initiated by Fogel et al. (1966), with its motivation to generate an alternative approach to artificial intelligence (Fogel and Fogel, 1996). EP, ES and GA techniques are broadly similar as each generates a population of candidate solutions which are evolved subject to random modifications and compete to survive to the next generation.

In an EP process, an initial population of candidate solutions is randomly generated. Then, new individuals (offspring) are reproduced by applying a mutation strategy to each individual/solution in the population according to the distribution of mutation types. The severity of a mutation is judged according to the functional change imposed on the parents. In order to evaluate each offspring solution, its fitness is computed and then a stochastic tournament selection used to choose the fittest solutions to be retained to construct the new population of solutions.

Although EP has several applications in different areas, such as artificial neural networks (Yao and Liu, 1997) and real continuous function optimization (Xin et al., 1999), it has slow convergence to good near-optimal solutions when solving some multi-modal optimization problems (Yao et al., 1999). Moreover, because of the low diversity in the population, its performance progressively decreases (Ji and Wang, 2008).

E. Genetic Programming (GP)

Over a number of decades, automatic programming has been considered a tractable point of study to many researchers aiming at getting computers to automatically solve a problem. Among all artificial intelligence (AI) techniques, which are software technologies that make computers, or robots, have similar, or better, performance to the human computational ability in speed, accuracy and capacity (Mccarthy, 1989), genetic programming (GP) is considered the most potential way for automatically writing computer programs (Walker, 2001).

As GP is one of evolutionary algorithms, it can be paraphrased as "survival of the fittest". In GP, a population of computer programs to solve a problem, as individuals or artificial chromosomes, is iteratively transformed into a new generation of programs by applying the genetic operations which include crossover (sexual recombination), mutation, reproduction, gene duplication, and gene deletion. Over time, the best individuals survive and eventually evolve to tackle the given problem (Walker, 2001).

Although, GP is an extension of the GA (Holland, 1975), GA encodes candidate solutions to a problem in the population, but, in GP, the execution of several programs are the candidate solutions to the problem (Koza and Poli, 2005). Programs are presented in GP as syntax trees instead of lines of code and each tree contains nodes/points and links. The nodes show the instructions of execution while the links indicate the parameters/arguments for each instruction (Koza and Poli, 2005) .

Generally, for any problem domain, GP could be used to evolve computer program solutions, if, and only if, individual solutions can be compared and ranked. However, GP requires massive computing resources before solving any real-world problem (Walker, 2001).

2.6 Approaches for Representation of RCPSPs

For EC problems, Ashlock (2006) defined chromosome representation as the choice of the data structure that represents a solution and the variation operators that act on that structure. Choosing a good representation for a difficult problem can have a great impact on an EA's performance. Although there could have been many rewards obtained from searching various representations, the author mentioned many reasons for not doing this, such as the substantial cost of implementing and running well-designed experiments for each representation and then determining an approach for comparing solutions.

The following are the two basic approaches for chromosome representation.

- *Direct approach*: a solution is encoded as a vector of numbers, where each number represents an activity or gene, and it is the most straightforward technique.

- *Indirect approach*: each chromosome is represented by a sequence of rules for task assignments, which is not the original schedule (solution), and then the chromosomes are evolved by an EA to determine a better sequence of rules (Robbins, 2008) which is then used to construct a schedule.

Over the years, many schedule representation schemes have been introduced, such as natural data variables, list SGS, set-based and resource flow network (Artigues et al., 2013), and activity-list, random-key, priority rule, shift vector and schedule scheme representations (Kolisch and Hartmann, 1999). Brief descriptions of some of these representations are given in the following sub-sections.

2.6.1 Natural data variables representation

In this representation, a RCPSP selects variables to represent either the starting or finishing/completion times of activities. It is the simplest formulation of a RCPSP since the makespan criterion is considered to be in a non-pre-emptive environment (Artigues et al., 2013).

2.6.2 List SGS representation

Since selecting a solution among alternatives is a heuristic process, the priority list can be used to organize the scheduling process. A combination of a schedule generator (heuristic) and priority list (as a decision-maker) can be considered to provide a solution for RCPSPs as the list produced from a heuristic is considered an encoding of the solution. Also, this representation is the basic encoding of a solution applied by numerous greedy heuristics, such as serial and parallel SGSs (Artigues et al., 2013).

2.6.3 Activity list representation

An activity list (AL) representation of a schedule/solution for a RCPSP is a precedence-feasible list of activities (i.e., a permutation vector of activities in which each activity satisfies the precedence constraints) as each activity scheduled after all its predecessors in the list. A SGS is applied to decode the AL to obtain the corresponding schedule by selecting the activities according to their order in the list and scheduling them at their earliest starting times. For clarification, any list in which all activities satisfy the precedence constraints is called an activity list representation of the schedule and vice versa; for instance, considering the RCPSP example presented in Figure 2.1, the vector h = (1, 3, 2, 5, 4, 7, 6, 8, 10, 9, 11) is an activity list representation of the schedule while $h`$ = (1, 3, 4, 2, 5, 7, 6, 8, 10, 9, 11) cannot be one as it does not satisfy the precedence constraints (i.e., activity 2 must be listed after activity 4).

An AL representation has been widely used in many research works (Alcaraz and Maroto, 2001, Nonobe and Ibaraki, 2002, Bouleimen and Lecocq, 2003, Valls et al., 2008) because: (1) it is decoded easily and rapidly; (2) it always produces a feasible schedule/solution; (3) the form of its list can be easily modified and manipulated which increases the number of opportunities for finding an optimal solution (Moumene and Ferland, 2009).

2.6.4 Random key representation

The random key (or priority value) representation was first proposed by Bean (1994) and then Norman and Bean (1995) and Norman (1995) generalized it to solve the job scheduling problem (JSP). It encodes a solution as a vector of n (real-valued) numbers which assigns a number to each activity (j), where the j^{th} number relates to the corresponding j. Initially, the random keys are generated either randomly (Lee and Kim, 1996) or according to some priority rules (Leon and Balakrishnan, 1995) and then the array of them is transformed into a schedule using a serial or parallel SGS which schedules the activity with the highest random key of the eligible activities which satisfy all the constraints. Therefore, it can be said that random keys play the role of priority values (Kolisch and Hartmann, 1999).

For clarification, considering the RCPSP example in Figure 2.1, the optimal schedule in Figure 2.2 can be produced by applying either the serial or parallel SGS from the following random key array for 11 actual activities.

(0.9, 0.85, 0.72, 0.69, 0.61, 0.58, 0.55, 0.51, 0.4, 0.45, 0.1)

Random key representations are used to encode solutions for RCPSPs in many research works (Mendes et al., 2009, Sebt and Alipouri, 2012, Zhang et al., 2006) and also applied to represent chromosomes for multi-modal RCPSPs (Gonçalves et al., 2008).

The significant advantage of random keys is that all the offspring created by crossover are feasible which is achieved by moving much of the feasibility feature into the objective function evaluation, that is, any crossover vector will be feasible if any random key array/vector is considered a feasible solution (Mendes et al., 2009).

2.6.5 Priority rule representation

Priority rule-based representations, in which chromosomes are encoded as sequences of dispatching rules that are then evaluated by any EA to detect those with better sequences, have been proposed (Dorndorf and Pesch, 1995). However, the main problem is finding an efficient priority rule.

The common basis of all priority rule-based heuristics is discussed in the work of Giffler and Thompson (1960) and Storer et al. (1992) and more details are provided in (Panwalkar and Iskander, 1977), (Blackstone et al., 1982) and (Haupt, 1989).

Because of their easy implementation and low time complexity, priority dispatching rules are very popular for solving optimization problems.

2.6.6 Shift vector representation

Shift vector representations for RCPSPs, which are arrays of non-negative integer values used to represent solutions, were initiated by Sampson and Weiss (1993). An extension of the traditional forward recursion is used for decoding in which the starting time of an activity (j) is computed as the sum of the maximum finishing time of its predecessors and the jth shift value. In other words, the shift non-negative value in a certain position determines how many periods the corresponding activity will be shifted to be scheduled after its early starting time.

However, as resource constraints are not considered in the initial representation, which means that the shift vector may be an infeasible solution, a penalty function or any constraint-handling mechanism can be included to handle them (Demeulemeester and Herroelen, 2002).

2.6.7 Schedule scheme representation

Schedule scheme representation, which is based on the binary version of the schedule scheme approach for a B&B algorithm, was initiated by Brucker and Knust (1999). In it, the relationship between each pair of activities (e.g., (i, j)) is described by one of four disjoint relationships defined as a schedule scheme (C, D, N, F) (Demeulemeester and Herroelen, 2002) as follows.

1) $(i, j) \in C$ denotes that a conjunctive relationship exists between i and j, with i required to be finished before j starts, and is symbolized by '$i \prec j$'.
2) $(i, j) \in D$ denotes that a disjunctive relationship is assigned between i and j, which cannot be processed in parallel (not overlap), and is symbolized by '$i - j$'.
3) $(i, j) \in N$ denotes that a parallelity relationship is assigned between i and j, which are required to be processed in parallel, and is symbolized by '$i \parallel j$'.
4) $(i, j) \in F$ denotes that none of the above relationships is allocated between i and j (flexibility relationship).

The schedule scheme then represents a schedule in a way that satisfies the corresponding relationships. However, as this schedule may not be feasible (Demeulemeester and Herroelen, 2002), Baar et al. (1999) proposed a heuristic for converting it to a feasible one.

2.6.8 Solution representations in EAs for RCPSPs

In a RCPSP, it is necessary to encode a chromosome to simplify the crossover and mutation operators for a specific problem. Ozdamar (1999) represented a chromosome using indirect encoding. Storer et al. (1992) and Uckun et al. (1993) represented it by a serious of priority rules used in an iterative scheduling to produce a solution from the chromosome. In it, each chromosome is represented by a number of genes equal to twice the number of activities in the project, with two genes in each position. The first gene determines the execution mode and the second specifies the priority rule for selecting the candidate activities to be scheduled. In (Ozdamar, 1999), there are 12 activities, each with from one to three execution modes and their corresponding durations as well as renewable and non-renewable resources. In the author's study, he selected nine priority rules based on the results reported in previous articles (Alvarez-Valdes and Tamarit, 1989, Ulusoy and Özdamar, 1989, Ulusoy and Özdamar, 1994).

A valid chromosome for this problem is represented as:

Mode Assignment: (1, 1, 3, 1, 1, 2, 2, 1, 3, 1, 3, 1)

Priority Assignment: (1, 3, 2, 5, 9, 6, 2, 7, 7, 9, 6, 8)

The first set of genes represents the modes assigned to 1, 2,…, n activities, where n is the number of activities in the project. The second set can be read as the first scheduling decision should be carried out by sorting the schedulable activities according to a specific priority rule and selecting the highest priority activity, and the second be directed by another priority rule; and so on.

Cho and Kim (1997) modified the SA approach proposed by (Lee and Kim, 1996) by extending the random key representation to allow some activities to be delayed, with the aim of extending the search space. (Baar et al., 1999) introduced two versions of the TS algorithm. The first uses an activity list for solution representation and a serial SGS as a decoding procedure, and the second is based on a schedule scheme representation, where the neighbors are investigated by either placing the activities in parallel or deleting the parallelity relationship.

Hartmann and Kolisch (2000) compared the experimental results of existing heuristic methods, such as single-pass heuristics, with serial and parallel SGS, random and adaptive sampling approaches and some meta-heuristic methods such as GA and TS. They concluded that, generally, techniques which utilize activity list representations produce superior results to other representations for RCPSPs. Having several different representations for a single schedule is the main reason that a random key representation has inferior performance (Debels et al., 2006).

Hartmann (1998) developed a GA which utilizes an activity list for solution representation and compared it with two other GAs, one using a random key representation and the other a priority rule one. The serial SGS and two-point crossover are employed for the three algorithms according to the representation scheme used in each. In the proposed activity list-based GA, the initial population is determined using a random sampling method and the author reported that, based on a computational study, it outperforms the other two GAs as well as seven state-of-the-art heuristic approaches.

Afshar-Nadjafi et al. (2015) proposed a DE for solving RCPSPs. In it, a priority value representation is utilized to encode a project schedule and a serial SGS to obtain the schedule. Their proposed DE combines a local search and learning module in order to improve its quality. Then, its performance is evaluated by statistically comparing the solutions it obtains for various test problems in terms of the objective function (makespan) and computational times.

2.7 Existing Approaches for RCPSPs

As a RCPSP has been proven to be NP-hard, especially for large-scale scheduling problems (Blazewicz et al., 1983), many exact methods have been proposed for solving it. B&B-based algorithms are introduced in (Demeulemeester and Herroelen, 1992, Jalilvand et al., 2005), with their results showing their capability to find optimal solutions. However, their computational complexity significantly increases with increasing numbers of activities. Although there are fast algorithms, such as priority scheduling (Li et al., 1997) and greedy-based (Lupetti and Zagorodnov, 2006) ones, they may not satisfy all constraints. Cheng and Wu (2006) constructed a project scheduling model which includes a time constraint and presented a hybrid algorithm that combines B&B and heuristic algorithms. Their simulation results show that their proposed algorithm is better than the others.

Kolisch and Drexl (1996) proposed a new heuristic technique, which is a hybrid of priority rule and random search techniques, that employs two types of adaptations in order to determine the solution space. The results from their evaluations, which compare it with other proposed heuristics showed that it could be usefully applied to solve different hard problems in the project scheduling field.

As previously mentioned, several meta-heuristic algorithms have been proposed for solving RCPSPs, such as SA (Cho and Kim, 1997), TS (Lee and Kim, 1996), DE (Damak et al., 2009) and GA (Toklu, 2002). Bouleimen and Lecocq (2003) presented new SA algorithms for a RCPSP and its multi-modal version in which the objective function minimizes the makespan by replacing a traditional SA search scheme with a new design that considers the specificity and characteristics of the solution space of a project scheduling problem.

Merkle et al. (2002) proposed several new features for an ACO algorithm by combining two pheromone evaluation methods of ants to find new solutions. Their results show that, on average, the algorithm performs better than several other heuristics for RCPSPs. In several works (Zhang et al., 2006, Zhang et al., 2005b, Chen, 2011), PSO has been used to solve RCPSPs, with the authors concluding that a PSO-based approach could provide an efficient and easy-to-implement alternative for analyzing and solving RCPSPs. Kochetov and Stolyar (2003) introduced an EA based on a path re-linking strategy and a TS with a variable neighborhood. Nonobe and Ibaraki (2002) extended the definition of a RCPSP further to include various complicated constraints and objective functions and then developed a TS-based heuristic algorithm containing elaborations in terms of representing solutions and constructing a neighborhood.

Hartmann (1998) and Hartmann (2002) proposed two GA variants, the first a permutation-based GA in which SA, TS and GAs are used and the second a self-adaptive GA for constructing scheduling with or without constraints. Bettemir and Sonmez (2015) integrated the capabilities of the parallel search in GAs and fine tuning of the SA technique to propose a hybrid strategy, with the aim of producing an efficient algorithm for RCPSPs. Valls et al. (2008) presented a hybrid GA (HGA) involving several changes in the GA paradigm, in which a new crossover, a local improvement operator, a new way of selecting the parents to be combined and a two-phase strategy to re-start the evolution, are used. Its results show that it is fast and produces high-quality solutions better than those of state-of-the-art algorithms.

Zamani (2013) used a GA to solve a problem. He developed a magnet-based crossover operator that can preserve up to two contiguous parts from the receiver and one from the donor genotype, with the results showing better performances than the classical two-point crossover. Gonçalves et al. (2008) and Mendes et al. (2009) proposed a solution method that constructs schedules using a heuristic which builds parameterized active schedules based on priorities, delay times and release dates, respectively. They represented the solution using random keys and a GA as a search method, with the results demonstrating the effectiveness of the proposed algorithm. Agarwal et al. (2011) proposed a neurogenetic (NG) approach which is a hybrid of GA and neural network (NN) approaches and is proven to be independently superior to both NN and GA.

Debels et al. (2006) proposed a new meta-heuristic combining elements from a scatter search (SS), generic population-based search method and optimization heuristic, called electromagnetism (EM), for the optimization of unconstrained continuous functions based on an analogy with the electromagnetism theory. It is able to provide near-optimal solutions for relatively large RCPSP instances. Also, Debels and Vanhoucke (2007) extended a GA procedure by a so-called decomposition-based GA (DBGA) that iteratively solves sub-parts of a RCPSP's project. Chen and Shahandashti (2009) proposed a hybrid algorithm of a GA and SA (GA-SA) to produce a generic search method and compared its performance with that of a modified SA method (MSA) and several heuristic rules-based techniques, with the results showing that it performs better than all the other approaches.

Damak et al. (2009) proposed a DE algorithm for solving RCPSPs in a reasonable time, obtaining better results than those from other algorithms in the literature. In order to achieve a higher computational efficiency for RCPSPs, Yan et al. (2014) proposed a modified DE that uses two parallel mutation operations to improve its search capability, with the best individuals chosen from one target vector and two trail vectors by the selection operation. Although their modified DE performs better than GA and SA, there is no indication of its computational time.

Chen et al. (2010) combined a local search strategy, ACO and SS in an iterative process to produce an efficient hybrid algorithm (ACOSS) for solving RCPSPs, with the results showing that it produces good solutions for small instances and slightly better ones for large instances. Fang and Wang (2012) proposed a heuristic based on the framework of the shuffled frog-leaping algorithm (SFLA) for solving RCPSPs by combining a permutation-based local search (PBLS) and forward‑backward improvement (FBI) procedure to enhance its exploitation capability. Although the results show that the SFLA performs well for solving large instances, it obtains high values of deviation from the optimal solution for small ones. Fahmy et al. (2014) defined the justification technique as "*a simple technique which was presented for improving the quality of schedules generated with algorithms for solving RCPSPs*", implemented it with meta-heuristics and showed that embedding it with meta-heuristics improved performance.

2.8 Brief Discussion of Existing Approaches

Based on the literature reviewed in this chapter, it was noted that enumeration or exact methods, such as mathematical programming or B&B, are only applicable for solving small project instances (Demeulemeester and Herroelen, 1992, Jalilvand et al., 2005) as they are not computationally practical for large ones. Consequently, *heuristics* emerged.

Heuristic methods can find near-optimal solutions at an acceptable computational cost as they usually require less time and memory than exact approaches and, moreover, can be applied to a wide range of problems. However, they do not guarantee optimal results (Abdolshah, 2014) and also perform poorly with respect to the average deviation from the optimal objective function value (Brucker et al., 1999).

Meta-heuristics are used to develop a specific heuristic method as they can be easily adapted to a particular problem or problem class with much less effort than heuristics which makes them an appealing choice for implementation in general-purpose software (Ólafsson, 2006). However, as they have many drawbacks, as previously discussed, improving the performances of existing ones or developing new ones appears to be necessary, and their limitations motivated the emergence of hybrid methods in which several components, such as EAs and a local search or EAs and a heuristic are incorporated.

Of all the approaches discussed in this chapter, hybrid methods are almost the best meta-heuristics for a set of RCPSP instances. However, conducting comparisons among meta-heuristic algorithms in order to determine the best is a challenging issue for two reasons: (1) fine tuning of all the parameters of all the algorithms is required; and (2) the quality of a solution obtained by a metaheuristic depends on the available computing time. Also, focusing on individual aspects and components of heuristic and meta-heuristic methods is necessary to provide a better understanding of the performance of each of their components.

As there is no one algorithm capable of consistently solving a wide range of test problems or suitable for all problem classes, the concept of using multi-method and multi-operator algorithms for solving RCPSPs will be beneficial.

2.9 Chapter Summary

In this chapter, the importance of project scheduling in real-life applications was described and a very well-known model of it, a RCPSP's conceptual model and its complexity, discussed. Also, different exact, heuristic and meta-heuristic techniques which endeavor to achieve the optimality of RCPSPs were presented.

Based on the literature reviewed in this chapter, it is clear that hybrid algorithms offer promising potential for real-life problems as no single algorithm has been proven to be superior to any other over a wide range of test problems. Therefore, suggestions of hybrid heuristics and multiple approaches have recently been raised. However, a procedure for selecting the algorithms that should be used to design them has not been well studied. Based on this motivation, experimental analyses of different hybrid approaches, such as a GA and local search, DE and heuristic, and multi-EAs and heuristic are introduced in Chapters 3, 4 and 5, respectively.

Chapter 3

Genetic Algorithms for solving RCPSPs

The main aim of this chapter is to introduce a new GA for solving RCPSPs. Firstly, its components are described and then it is implemented to solve a number of test problems of different sizes. The experimental results for different RCPSPs are elaborated and the effects of the algorithm's components on its performance discussed. Finally, comparisons with state-of-art-algorithms are presented.

3.1 Introduction

In classical RCPSPs, a project consists of a set of activities, each of which has a known deterministic duration and is allowed to be executed only once. These activities have to be scheduled in a way that minimizes the makespan of the project, that is, its total duration. As discussed in Chapter 2, such optimization problems are considered NP-hard ones.

Over the years, many exact, heuristic and meta-heuristic algorithms for solving RCPSPs have been introduced. In exact ones, such as integer programming, dynamic programming and the branch and bound (B&B) algorithm, the search for an optimal solution in specific real-world applications can be complex for many reasons, such as the problem being large, the data and parameters dynamic or too complex to express mathematically and the existence of contradictory objectives. Due to such complexity, an exact approach is much slower than a heuristic one and, therefore, its computational costs are higher (Widmer et al., 2010).

On the other hand, heuristic methods can be easily amended or combined with other techniques to construct hybrid algorithms, a flexibility that allows them to be applied in a wide range of problems. However, they are still not sufficiently good in terms of their average deviations from the optimal objective function value (Brucker et al., 1999). Several meta-heuristic methods for solving RCPSPs, such as simulated annealing (SA) (Cho and Kim, 1997), a tabu search (TS) (Lee and Kim, 1996), differential evolution (DE) (Damak et al., 2009) and a genetic algorithm (GA) (Toklu, 2002), have been introduced. Although they have constantly been among the most popular for handling combinatorial optimization problems, they are much more expensive in terms of computational time and further research is required to improve their effectiveness (Widmer et al., 2010).

Also, a few number of memetic algorithms (MAs) which belong to the class of stochastic global search heuristics in which evolutionary algorithm-based approaches are combined with problem-specific solvers, such as local search heuristic techniques and approximation algorithms (Neri and Cotta, 2012) have been introduced for solving RCPSP. A MA method is based on a population of agents and has proven to be successful in a variety of problem domains, in particular, for obtaining approximate solutions to NP-hard optimization problems (Moscato and Cotta, 2010). However, a good definition and analyzes of the different components of such algorithms is required.

Motivated by the research gaps mentioned above, in this chapter, a new MA consisting of a GA and multiple local search techniques is proposed. A new repairing heuristic procedure for generating feasible solutions in the initial population is also introduced. These solutions, including the repaired ones are then evolved using GA operators and, to increase the convergence speed, an ensemble of two local search methods is adopted.

The algorithm is tested by solving 60 standard benchmark problems chosen from the well-known test set, the project scheduling problems library (PSPLIB) created by Kolisch et al. (1999). The effect of the algorithm’s components, such as its: (1) validation rate, which is the number of solutions repaired to be feasible in the initial population; (2) local search rate; (3) mutation rate; and (4) population size, are discussed. The final variant of the proposed algorithm is compared with: (1) variants of the classical GA with different mutation rates; (3) other state-of-the-art algorithms.

The rest of this chapter is organized as follows. In section 3.2, the methodology of the proposed MA and its components are explained. The experimental study and analyses of its different components are discussed in section 3.3. Finally, section 3.4 provides a summary of this chapter.

3.2 Methodology

In this section, the framework and components of the proposed MA are discussed. In its process, an initial population is randomly generated and then some infeasible individuals are selected to undergo the repairing process to convert them to feasible ones, the best of which are sorted according to their fitness values and constraint violations. A tournament selection is used to select the best individuals to participate as parents, a crossover applied to generate a new offspring and a mutation with an adaptive rate applied to alter the offspring.

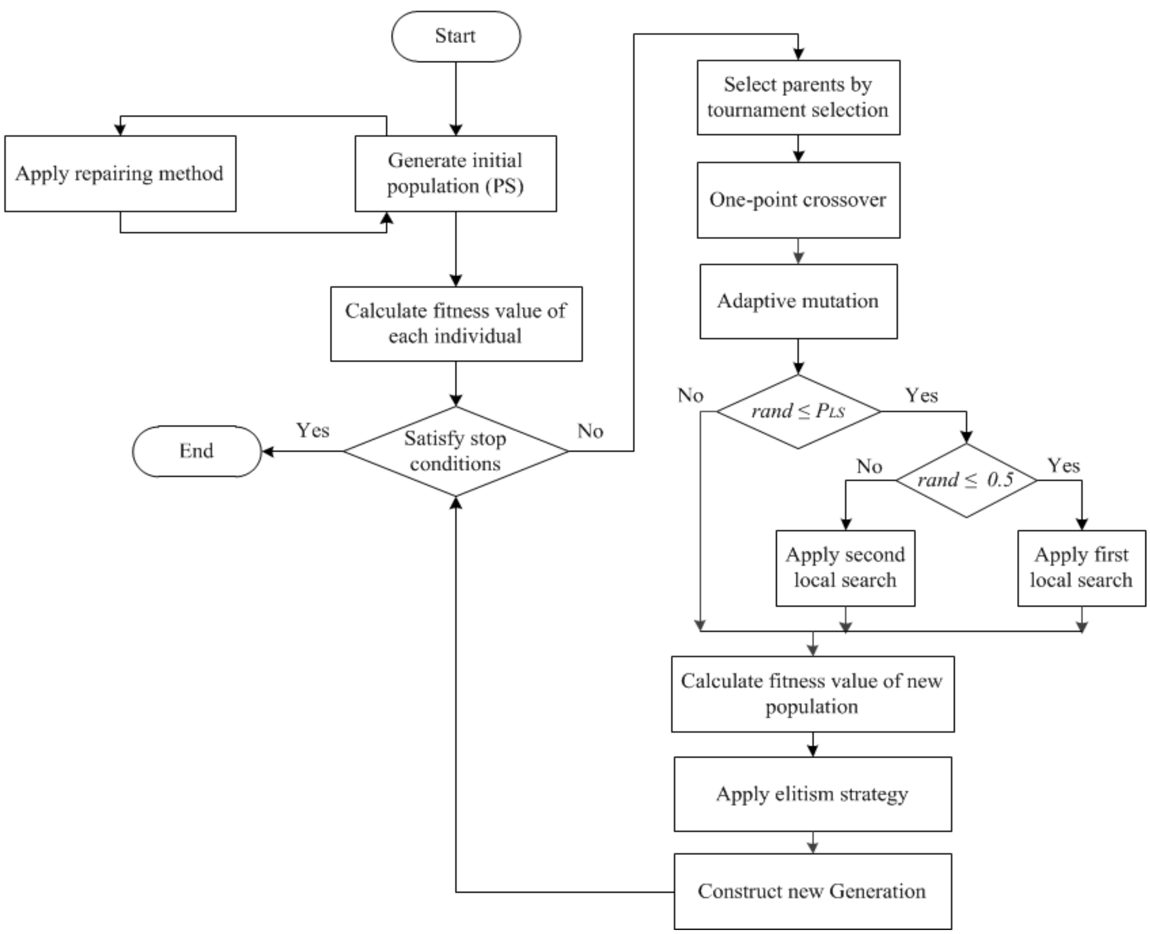


Figure 3.1: General framework of proposed algorithm

As it is very common for a RCPSP to become stuck in a local optimum, involving more variation operators helps an algorithm to avoid this situation and move to a more promising search region. This is achieved in the proposed MA by adopting two local search techniques, one of which, based on a probability of 0.5, is used to enable more exploitation (the refinement of existing solutions) to the best solution found so far in the hope of accelerating the convergence rate and another to explore more solutions. Finally, an elitism strategy is applied which, in the context of a GA, means that the best solution found so far during the search constantly survives to the next generation. The operations of the proposed MA continue until the pre-defined stopping conditions are met.

Figure 3.1 shows a flowchart of the process and each of the algorithm's components is discussed in the following sub-sections.

3.2.1 Representation

Every solution in the population is represented by a vector/chromosome, the length of which is equal to the number of activities in the project, with integer values used to represent the activities as:

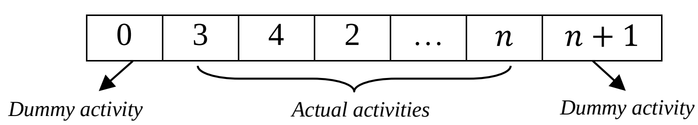
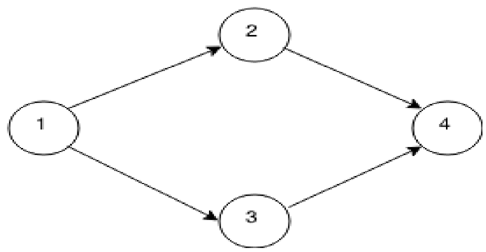


Figure 3.2: Chromosome representation

As, in RCPSPs, an activity cannot start unless all its predecessors have been finished, it is necessary to identify the precedence relationships among all the activities. In this algorithm, an incidence (binary) matrix is generated to represent the predecessor-successor relationship for each activity in the project; for instance, that in Figure 3.3 shows the precedence constraints among four activities, where Figure 3.3 (a) shows a network graph that indicates the dependency constraints. In Figure 3.3 (b), each row represents the predecessor activities of (the row number)th activity and each column the successor activities of (the column number)th activity; for example, activity 1 does not depend on any activity as it is the start node, activities 2 and 3 depend on activity 1 while activity 4 depends on both activities 2 and 3 and, as it is the last node, has no successor activities.



(a)

Predecessors →

Successors ↓	1	2	3	4
1	0	0	0	0
2	1	0	0	0
3	1	0	0	0
4	0	1	1	0

(b)

Figure 3.3: Representation of predecessor-successor relationship

3.2.2 Fitness evaluations

In the beginning, a number of chromosomes, equal to the population size (PS), is randomly generated to form the initial population and, generally, the fitness value (makespan) and/or constraint violations are used to evaluate the acceptability of any solution.

For each schedule, an improved serial SGS (described in Chapter 2) is applied to construct solutions. In it, each activity in a schedule is processed under the restriction of the resource availability constraints as the activities of a candidate solution are scheduled by their appearances in the generated schedule and each activity can be processed if, and only if, its required number of resources does not exceed the available amount of resources at a specific time. Therefore, the schedule produced is guaranteed to satisfy the resource availability constraints.

As the resource availability and precedence relationships are the two main constraints that must be satisfied in RCPSPs and, given that the algorithm produces a resource constraint-satisfied schedule as described above, the violation of the precedence constraint is considered a measure of solution feasibility. As the violation value of each solution can be determined by calculating the number of violations of the precedence constraint for every activity within that schedule, any solution can be considered feasible if its violation value is less than 1.

The fitness value is determined by calculating the total duration of a project to completion and a schedule is constructed by adding one activity after another. After each new activity is inserted, its finishing time is set as the makespan.

3.2.3 New repairing method

It is noted that, as RCPSPs are complex optimization problems, the lack of feasible solutions in the initial population affects the quality of the evolutionary process. Therefore, a repairing heuristic method for obtaining feasible solutions in this population is proposed.

In this process, some infeasible solutions are selected from the initial population and then modified by reordering the positions of each of their activities by satisfying their predecessor and successor constraints. The steps in the proposed repairing and violation calculation method are shown in Figure 3.4.

It is worth mentioning that, to maintain diversity, the repairing method is applied to a certain percentage (R_m) of the population size (PS).

Set $i = 1$; $feas_{count} = 1$; $violation(S_i) = 0$;
While $i < PS$ **do**
    Generate a random solution S_i
    While $feas_{count} < R_m$ **do**
        Find Pre_j of each gene j in S_i
        If all (Pre_j) is already scheduled, **then**
            j is feasible
        Else
            $violation(S_i) = violation(S_i) + 1$
            Rearrange activities positions in Pre_j
            Add activity j to the schedule
            $feas_{count} = feas_{count} + 1$
        End if
    End while
    $i = i + 1$
End while

Figure 3.4: Proposed repairing method

3.2.4 Selection

In this phase, the individuals in the population are sorted according to their fitness and violation values, with the best having the minimum fitness value (makespan) and a zero or minimum violation. Then, a tournament method is applied to select individuals as parents to generate offspring for the next generation by running several ‘competitions’ among a few individuals chosen randomly from the population. As solutions are either feasible or infeasible, three rules can be used to manage this selection process: (**1**) of two feasible solutions, the better one (according to their fitness values) is selected; (**2**) a feasible solution is always better than an infeasible one; and (**3**) of two infeasible solutions, the one with the smaller sum of constraint violations (or lowest violation value) is chosen.

3.2.5 Crossover

Crossover is considered the key operator in a GA. In the proposed algorithm, it is applied according to a crossover search parameter with a probability (p_c). As suggested by Hartmann (1998), a one-point crossover, which has the advantage that, if the parents are feasible, it guarantees that the offspring produced are also feasible, is used. In this process, two individuals are randomly selected as parents (say, P_1 and P_2), as described in section 3.2.4. A random integer (q) is chosen as the crossover point, where $1 \leq q \leq n$ and n is the last actual activity in the schedule. The offspring (OS) is generated by inheriting the activities in positions 1 to q from P_1, that is, $OS_j = P_{1_j}\ \forall\, j \leq q$, and the remaining positions ($q+1, \dots, n$) taken from P_2, with the activities taken from P_1 not considered again. Considering the project illustrated in Figure 2.1 in Chapter 2 as a simple example, in Figure 3.5, two individuals are selected as parents (P_1 and P_2) and q set to 5.

P_1	1	3	2	6	9	5	4	7	8	10	11
P_2	1	2	3	5	4	6	8	10	7	9	11
OS	1	3	2	6	9	5	4	8	10	7	11

Figure 3.5: One-point crossover

In Figure 3.5, the activities from 1 to q are copied from P_1 to OS while those from $q+1$ to n in OS are taken from P_2 after the absence of any redundancy in OS is confirmed. Therefore, as the first three activities as well as 6 and 9 in P_2 cannot be copied to OS because they are already listed, activities 4, 5, 7, 8, 10 and 11 are taken. From this figure, it is clear that, as both P_1 and P_2 satisfy the precedence constraint, the offspring produced is also feasible. The pseudo-code of the one-point crossover is presented in Figure 3.6.

Generate random number ($rand$) within the rang [0, 1]
If $rand \leq p_c$ **then**
    Select P_1 and P_2 as parents using the tournament selection scheme (section 3.2.4)
    Select $q \leftarrow [1, n]$
    For j=1 to q **do**
        $OS(j) \leftarrow P_1(j)$
    End for
    $t \leftarrow q + 1$
    For s=1 to n **do**
        If $P_2(s)$ not exist in $OS\ [1:q]$ **then**
            $OS(t) \leftarrow P_2(s)$
            $t \leftarrow t + 1;$
        End if
    End for
End if

Figure 3.6: One-point crossover scheme

3.2.6 Mutation

In a GA, a mutation operator is used to increase the diversity of the population by making a small change which enables the GA to explore promising areas of the search space (Korejo et al., 2010). In this algorithm, a random number is generated within the range of [0, 1] and, if it is less than or equal to the mutation rate (p_m), the chromosome is modified by choosing one of the activities at random and swapping it with its adjacent gene after ensuring that there are no precedence relationships between them.

Based on the example in Figure 2.1, Chapter 2, in the chromosome presented in Figure 3.7, activities 6 and 8 are swapped to produce a new feasible offspring.

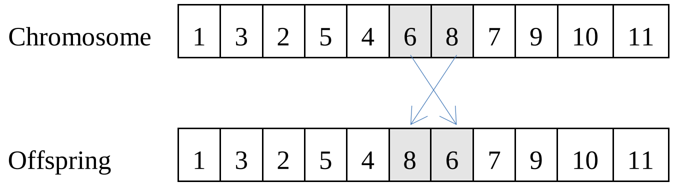


Figure 3.7: Example of mutation

3.2.7 Local search

After obtaining a new set of individuals from the different GA processes, one of two different local search procedures is applied according to a pre-defined probability (P_{LS}). The first is applied to the best 10% of the current population with the purpose of obtaining near-optimal solutions and the second to the best solution obtained so far in the current population in a bid to achieve the optimal solution. The use of multiple local searches provides more variation than mutation as it probes a more promising region of the search space in the hope of improving the promising current best solutions by refining them.

```
Generate a random number rand ∈ [0 − 1]
If rand ≤ P_LS, then
    Generate u ∈ [0 − 1]
    If u ≤ 0.5, then
        Apply the first local search; else
        Apply the second local search
    End if
End if
```

Figure 3.8: Mechanism for selection of two local searches

In its selection mechanism (Figure 3.8), a random number (*rand*) is generated and, if it is less than 0.5, the first local search is applied; otherwise the second one is used

The first local search is applied to the best individual in the entire population and begins by changing a single gene of the chromosome through swapping two consecutive activities. This process is similar to that of the mutation operator except that the feasibility status of the solution is checked before any movement (swapping) is conducted. If the swapping does not cause a violation of the precedence constraint, the movement is committed and the fitness value of the new individual calculated. If this value is better than the old one, the new individual is accepted as the best solution; otherwise the same process is applied to two other randomly chosen consecutive activities. The pseudo-code of the first local search is shown in Figure 3.9.

```
best_ind ← the best individual in the current population
old_fit ← the fitness value of the best_ind
For j = 1 to n do
    If j does not exist in the predecessors of j + 1 then
        swap(j, j + 1)
        new_fit ← the fitness value of the best_ind after swapping activities
        If new_fit < old_fit then
            The best individual in the current population ← best_ind
        End if
    End if
End for
```

Figure 3.9: First local search

The second local search is applied to the first λ individuals, where $\lambda \in [1 - PS]$. In it, a random activity (l) in the schedule is selected and its predecessor and successor activities are identified. Then, l is swapped with another random activity (l_2) within a specified range by a circular displacement (Bouleimen and Lecocq, 2003). This range is determined by two bounds, the last predecessor (l_{pred}) and first successor (f_{succ}) activities of l,. Two ranges can be defined by these bounders, a range from l_{pred} to l and another from l to f_{succ}. l_2 is randomly selected according to the one that contains more activities or has the largest difference from l. In order to fulfil the predecessor-successor relationships, it is checked that the activities in the range between l and l_2 are not located in the predecessor or successor activities of l_2. The pseudo-code of the second local search is shown in Figure 3.12.

As a further illustration, the project in Figure 2.1 in Chapter 2 is again considered and the following individual chosen to be altered by the second local search as:

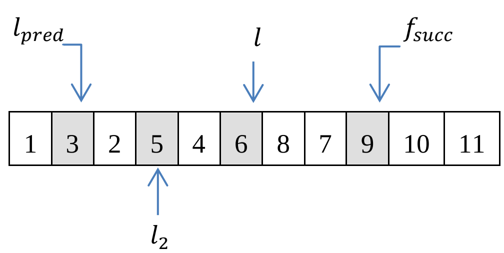


Figure 3.10: Individual before local search

As the difference between l and l_{pred} is more than that of the other side, l_2 is randomly selected from the range of $[l_{pred},\ l]$ (Figure 3.10) and then all activities in the range between l and l_2 are checked to ensure that they are not in the successor activities of l_2. Thereafter, l takes the place of l_2 and a circular displacement is performed to produce the new individual, as shown in Figure 3.11.

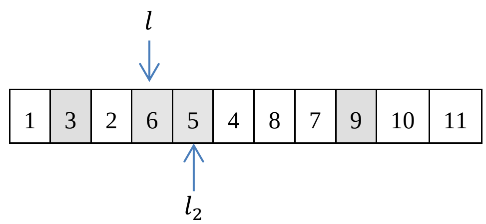


Figure 3.11: New individual after local search

Again, the fitness value is calculated for the generated individual and compared with that of the old one, with the better one is selected.

For each individual (ind) in the first γ individuals **do**
  old_{fit} ← the fitness value of the ind
  For each activity (j) **do**
    $last\_pred$ ← the last predecessor activity of j
    $first\_succ$ ← the first successor activity of j
    $pred_{dist} \leftarrow j - last\_pred$
    $succ_{dist} \leftarrow first\_succ - j$
    If $pred_{dist} \leq succ_{dist}$ **then**
      $rand_g$ ← select random number within the range $[last\_pred, j]$, **else**
      $rand_g$ ← select random number within the range $[j, first\_succ]$; **End if**
    If $rand_g$ does not exist in the predecessors, or the successors, of j **then**
      Update the position of j to be at the position of $rand_g$
      Update the position of $rand_g$ to be at the position of $rand_g + 1$, and so on in a circular displacement
      new_{fit} ← the new fitness value of ind
      If $new_{fit} < old_{fit}$ **then**
        Replace individual of old_{fit} with individual of new_{fit}
      End if
    End if
  End for
End for

Figure 3.12: Second local search

3.2.8 Elitism

At the end of each generation, the newly generated individuals are moved to the next generation. In the traditional random immigrant method which transfers individuals through generations, some random individuals are injected to replace some from the new population. Using this method, the population diversity level may be increased and, as a consequence, the GA's performance improved in uncertain/dynamic environments (Jourdan et al., 2001). However, randomly displacing individuals in the next generation may produce solutions which are worse than the existing ones. Also, if the population has either not, or only slightly, changed, the random immigrants may distract the optimization ability of the GA.

Therefore, in the MA, the best two or three individuals are randomly selected to be moved directly to the next generation in order to maintain good solutions through generations.

3.3 Experimental Study

In this section, a brief description of the benchmark problem is provided, the results obtained by the proposed MA reported, some parametric analyses of the proposed algorithm conducted and, finally, comparisons with some state-of-the-art algorithms discussed.

3.3.1 Benchmark problems

Standard benchmark problem sets from the PSPLIB created by (Kolisch et al., 1999, Kolisch and Sprecher, 1997) are used. The investigated algorithms have been applied on the single mode data sets cases consisting of four sets of J30 (30 activities), J60 (60 activities), J90 (90 activities) and J120 (120 activities), each of which had four resource types. Based on three complexity factors, the J30, J60 and J90 instances were grouped into 48 instances and J120 into 60, with each instance containing 10 different problems. The three complexity factors are defined as the resource factor (RF), network complexity (NC) and resource strength (RS) (Kolisch and Sprecher, 1997).

The RF, which was scaled in the range of [0, 1], defines the average proportion of resources required for each task; for instance, if RF=1, each job required the use of all resources whereas, if RF=0, no resources were required by any job. The NC reflects the average number of non-redundant precedence relationships for each activity, including dummy ones. The RS scales the proportion of resource usage and availability and was also selected from the interval [0, 1], where RS=0 means that the problem was highly resource constrained and RS=1 that it was resource unconstrained.

For the J30, J60 and J90 instances, the parameter levels were set as $NC \in \{1.5, 1.8, 2.1\}$, $RF \in \{0.25, 0.5, 0.75, 1\}$ and $RS \in \{0.2, 0.5, 0.7, 1\}$ and, for the J120 ones, $NC \in \{1.5, 1.8, 2.1\}$, $RF \in \{0.25, 0.5, 0.75, 1\}$ and $RS \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$.

In order to show how the different values of the three complexity factors determine the complexity of each instance, Figure 3.13 is introduced. In it, the x-axis represents the instance number of J30 and the y-axis represents the complexity level of each instance which can be calculated according to:

$$C_{ins} = \frac{(NC_{max} - NC_{ins}) + RF_{ins} + (1 - RS_{ins})}{3} \quad (3.1)$$

where C is the complexity level of each instance (ins) and NC_{max} is the maximum value of NC complexity factor.

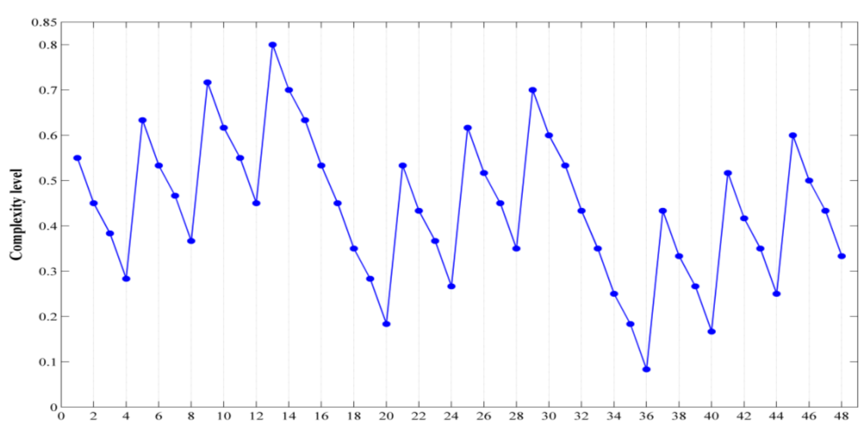


Figure 3.13: The complexity levels of 48 instances of J30

In this chapter, in an initial step to evaluate the proposed MA, 16 problems of J30 instance and 15 from each of J60, J90 and J120 ones, that is, a total of 61 problems, were randomly selected from three different instances for testing MA in different complexity levels. The test problems had the same values of both NC and RF but three different ones of RS, that is, RS= {0.20, 0.50, 0.70} respectively. According to Kolisch et al. (1995), the complexity of a RCPSP increases with an increasing RF value and decreasing NC and RS values. Therefore, the data set can be gathered into three groups: (1) problems with RS=0.20 which are then the most difficult group followed by (2) those with RS=0.50 while (3) those with RS=0.70 are the easiest.

3.3.2 Parameter settings

The proposed MA was coded using Matlab and implemented on a PC with a 3.4 GHz processor, 16G RAM and Windows 7. To judge its performance, the average percentage deviations ($AvgDev(\%)$) from optimal solutions for J30 instances or from the critical path lower bounds for J60, J90 and J120 as reported by Stinson et al. (1978) were used, where the average deviation can be calculated according to:

$$AvDev(\%) = \left(\frac{1}{S}\sum_{s=1}^{S}\frac{BS_s - LB_s}{LB_s}\right) \times 100 \tag{3.2}$$

where S is the total number of instances used, BS_S the best solution achieved by an algorithm for S instances and LB_S the pre-known lower bound of a s instance.

The parameters were set as PS = 100, p_c = 1 and p_m adaptively calculated according to:

$$p_m = Max\left(\partial_{min}, \partial_{max} - (\partial_{max} - \partial_{min}) \times \left(\frac{cfe}{Fit_{max}}\right)\right) \tag{3.3}$$

where ∂ is the lower limit of the mutation rate, ∂_{max} the initial value of the mutation rate, cfe the number of fitness evaluations and Fit_{max} the maximum number of cfe, with ∂_{min}=0.05 and ∂_{max}=0.2.

The local search parameter (P_{LS}) was set to 0.2 and the tournament pool size randomly selected as 2 or 3, with the number of individuals (R_m) in the initial population that must be feasible set to 25% of PS. Justifications of a selection of thesis parameters are described in Section 3.3.4.

3.3.3 Computational results

For each test instance, 30 independent runs were executed. There were two stopping criteria: (1) run the algorithm for up to $Fit_{Max} = n \times$10,000 fitness evaluations; or (2) run it until no improvement in the fitness value during 150 consecutive generations was found. In Table 3.1, the $AvgDev(\%)$ from the optimal solutions ($AD\%$) for J30 instances and from the critical path lower bound for the J60, J90 and J120 ones, and their average CPU times in seconds (t) for the three groups defined in Section 3.3.1 are presented.

From Table 3.1, it has been observed that the quality of the obtained solutions increases and the average computational time of MA reduces with increasing RS values from 0.20 to 0.70. The results show also that the proposed MA has achieved the optimal solutions for all J30 test problems with deviation values from the optimal solution equal to zero. For J60, the proposed algorithm obtained the optimal solutions for 54% of test problems and 67% of J90. Finally, although, no optimal solution achieved for test problems of J120, the algorithm showed a very low average deviation values.

Group	RS	J30		J60		J90		J120	
		$AD\%$	t	$AD\%$	t	$AD\%$	t	$AD\%$	t
1	**0.20**	0	3.09	7.84	16.46	17.34	23.29	51.50	32.05
2	**0.50**	0	1.48	0.64	14.05	0.00	13.06	35.30	30.25
3	**0.70**	0	0.13	0.59	13.818	0.00	9.69	12.01	24.17
Average		0.00	1.48	3.02	14.77	5.78	12.68	32.94	28.82

Table 3.1: Results from proposed MA with different RS values

To provide an indication of the influence of RS values in the convergence of the proposed MA, a few plots of some J60 instances with different RS values are shown in Figure 3.14.

In this figure, it can be noted that the MA converged quickly towards the optimal solution for problems with RS=0.70 (J60.3-3) and took some time to converge for others with RS=0.50 (J60.2-3), while it was taking longer time to solve problems with RS=0.20 (J60.1-1). Therefore, it is clear that this discrepancy was due to variations in the RS complexity factor discussed in section 3.3.1 that managed the degree of difficulty of each problem.

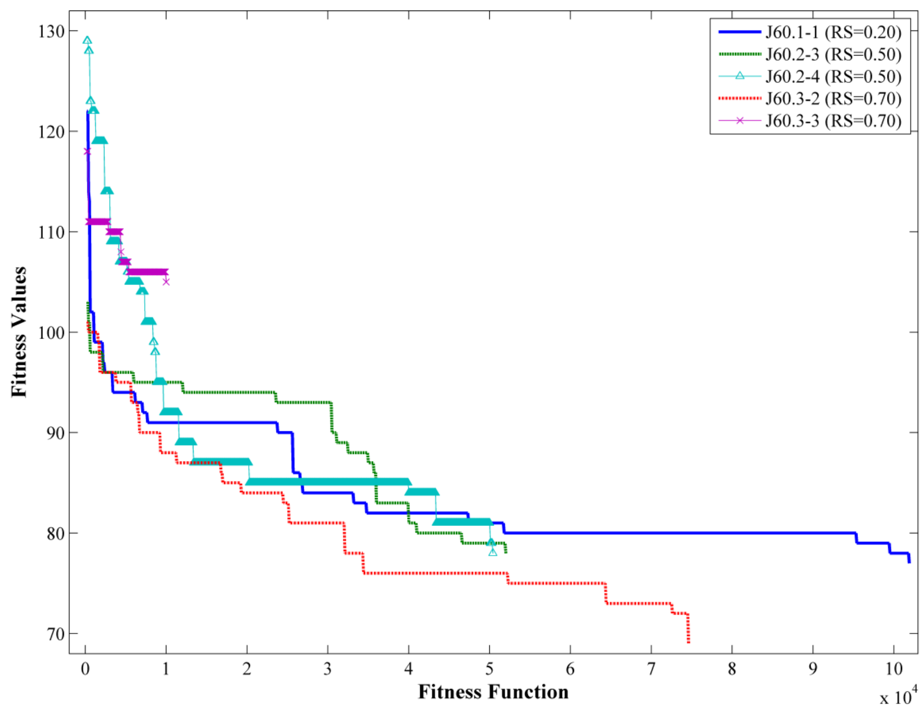


Figure 3.14: Convergence plots of MA for some problems of J60

3.3.4 Parametric analysis

In this section, five sets of experiments designed to analyze the effects of: 1) R_m, number of repaired solutions; 2) P_{LS}; 3) p_m; 4) PS; and (5) using more than one local search; with 15 test instances of J60 are discussed. However, to investigate the influence of each parameter, one problem from each of three instances with complexity values [NC, RF, RS] = [1.50, 0.25, 0.2-0.7] was used. Convergence plots of the three problems, J60.1-1, J60.2-4 and J60.3-5, are given for each parameter. In all the experiments, the MA was run up to 25 times with 5000 generations.

Note that, the selection of the parameters is done in a sequential manner in which the best parameter found in an experiment is fixed in the subsequent ones.

3.3.4.1 Effect of R_m

The MA was run with different values of R_m of (1) R_m= 0% of PS, (2) R_m= 25% of PS, (3) R_m= 50% of PS and (4) R_m= 100% of PS. All the other parameters were the same as those described in section 3.3.2.

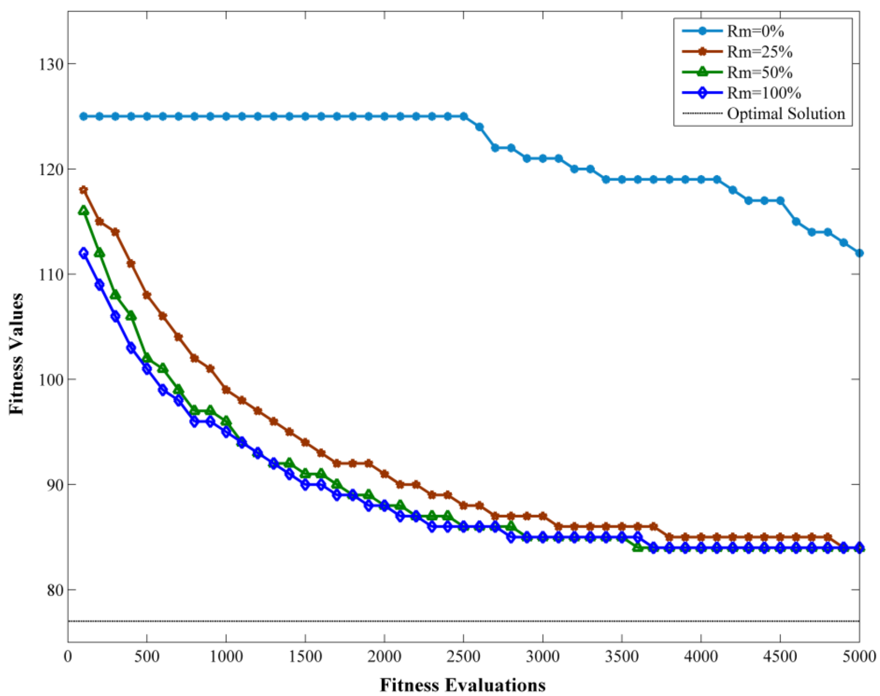


Figure 3.15: Convergence plots of J60.1-1 with different R_m values

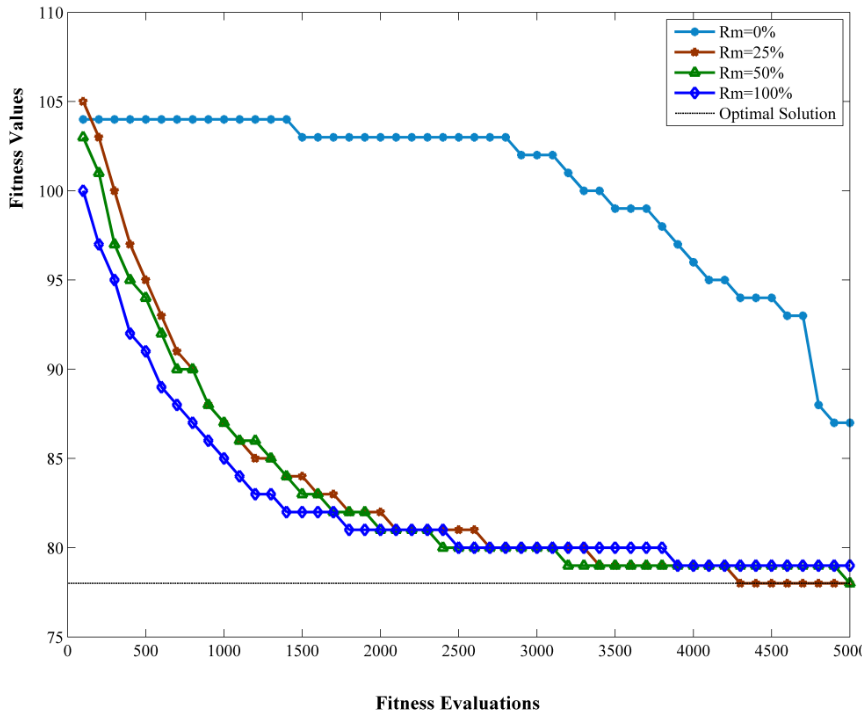


Figure 3.16: Convergence plots of J60.2-4 with different R_m values

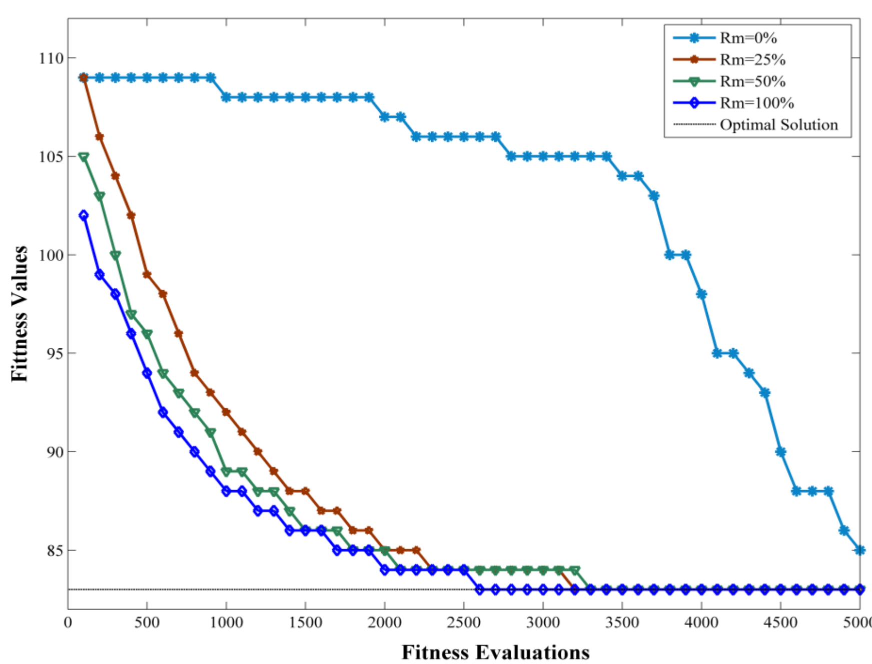


Figure 3.17: Convergence plots of J60.3-5 with different R_m values

Figures 3.15, 3.16 and 3.17 show the convergence plots of the J60.1-1, J60.2-4 and J60.3-5 problems, respectively, produced from the first experiment. It is clear that applying the repairing method had a significant effect on the performance of the algorithm. However, when R_m was further increased from 25% to either 50% or 100%, its performance might have degraded, as shown in Figure 3.16, which may have been due to the lack of diversity in the initial population.

In Table 3.2, the $AvgDev(\%)$ and computational times of the solved problems are shown. Although the computational times of R_m=50% and R_m=100% were greater than those of both R_m=0% and R_m=25%, it is clear that the quality of solutions slightly increased with an increasing R_m value. In order to achieve a good balance between the solution quality and time cost, these figures suggest that R_m should be 25%.

$\boldsymbol{R_m}$	**0%**	**25%**	**50%**	**100%**
$\boldsymbol{AvgDev}(\%)$	17.37	3.43	3.45	**3.36**
CPU time	**2.72**	3.16	3.22	3.27

Table 3.2: Average deviations and CPU times of variants of $\boldsymbol{R\_m}$

3.3.4.2 Effect of $\boldsymbol{P_{LS}}$

In this experiment, the effect of the local search was studied by running the MA using different values of P_{LS}: (1) P_{LS}=0, where the local search was switched off; (2) P_{LS}= 0.2; and (3) P_{LS}= 0.5 with the best variant found in the previous subsection.

Again, convergence plots of the J60.1-1, J60.2-4 and J60.3-5 problems are shown in Figures 3.18, 3.19 and 3.20, respectively. When the local search was switched off, the algorithm was not able to converge quickly in the complex J60.1-1 problem. Also, by increasing the probability of using the local search, the convergence of the algorithm might have degraded because of more emphasis being placed on refining existing solutions to improve their quality rather than exploring new ones (Garg and Mittal, 2014).

Table 3.3 shows the $AvgDev(\%)$ and computational times using the local search with 0, 0.2, and 0.5 in which it can be seen that P_{LS} with 0.5 outperformed the other two values in terms of $AvgDev(\%)$ but took more computational time than both of them.

Figures 3.18, 3.19 and 3.20 and Table 3.4 suggest that applying a 25% multiple local search was effective.

P_{LS}	**0**	**0.2**	**0.5**
$AvgDev(\%)$	5.27	3.43	**3.40**
CPU time	2.82	**2.25**	3.29

Table 3.3: Average deviations and CPU times of variants of P_{LS}

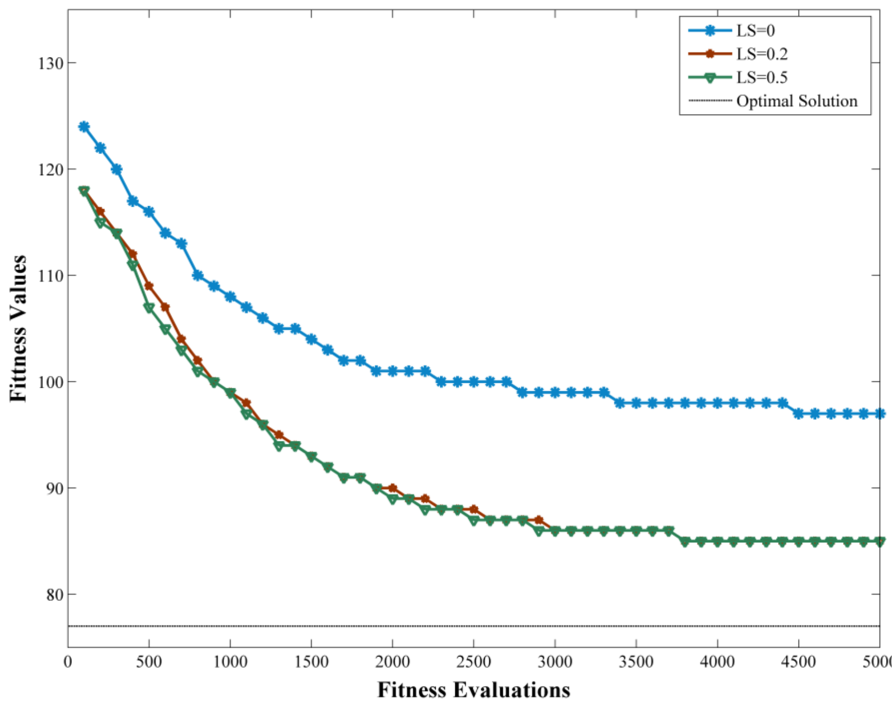


Figure 3.18: Convergence plots of J60.1-1 with different P_{LS} values

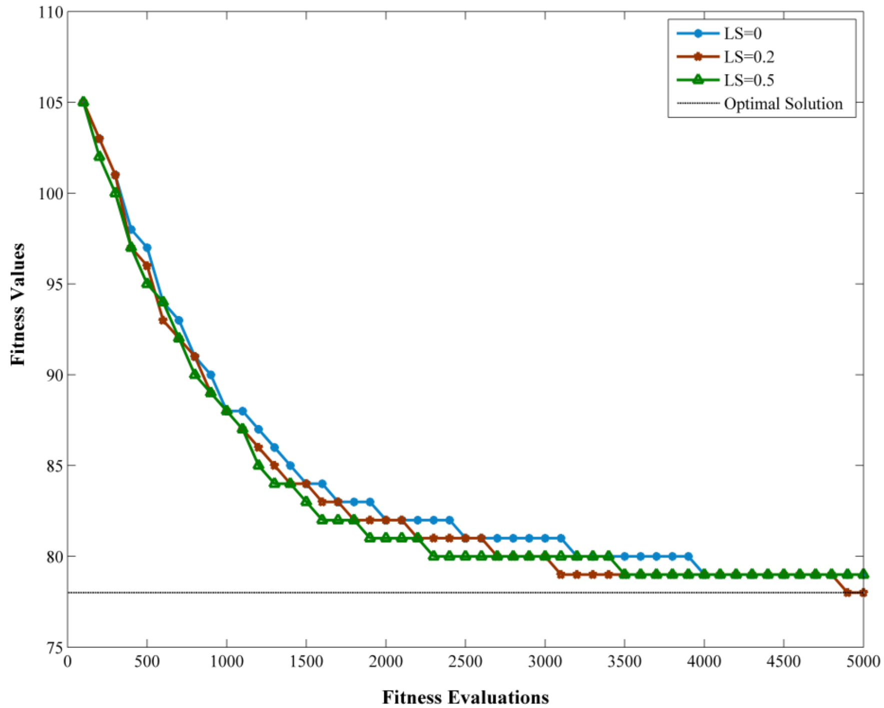


Figure 3.19: Convergence plots of J60.2-4 with different P_{LS} values

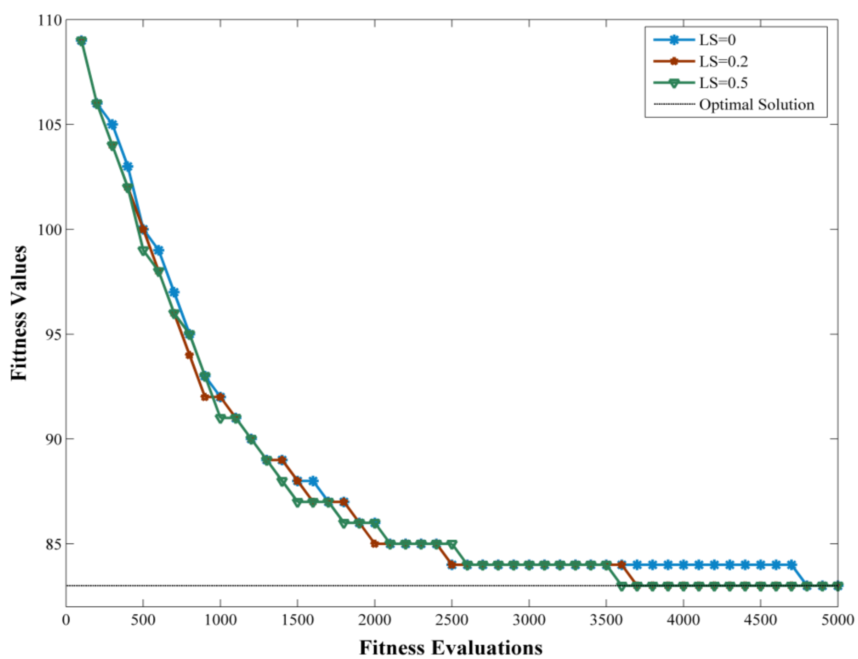


Figure 3.20: Convergence plots of J60.3-5 with different P_{LS} values

3.3.4.3 Effect of $\boldsymbol{p_m}$

To further analyze the influence of the mutation rate on the performance of the proposed MA, the same set of test problems was solved using different values of p_m: (1) p_m=0.05 (MA$_{p_{m=0.05}}$); (2) p_m=0.2 (MA$_{p_{m=0.2}}$); and (3) p_m adaptively calculated in the range of [0.05, 0.2] (MA$_{adaptive\ p_m}$).

Figures 3.21, 3.22 and 3.23 demonstrate that MA$_{adaptive\ p_m}$ achieved better performance than the variants with p_m = 0.05 and 0.2; for example, in Figure 3.22, it obtained the optimal solution for problem J60.2-4 which the others could not.

Table 3.4 shows the $AvgDev(\%)$ and CPU times for all variants which reveals that MA$_{adaptive\ p_m}$ achieved both the best average deviation and best CPU time.

	MA$_{p_{m=0.05}}$	**MA$_{p_{m=0.2}}$**	**MA$_{adaptive\ p_m}$**
$\boldsymbol{AvgDev}(\%)$	4.25	3.94	**3.43**
CPU time	2.94	3.13	**2.25**

Table 3.4: Average deviations from best solutions and average CPU times of proposed algorithm with different mutation rates

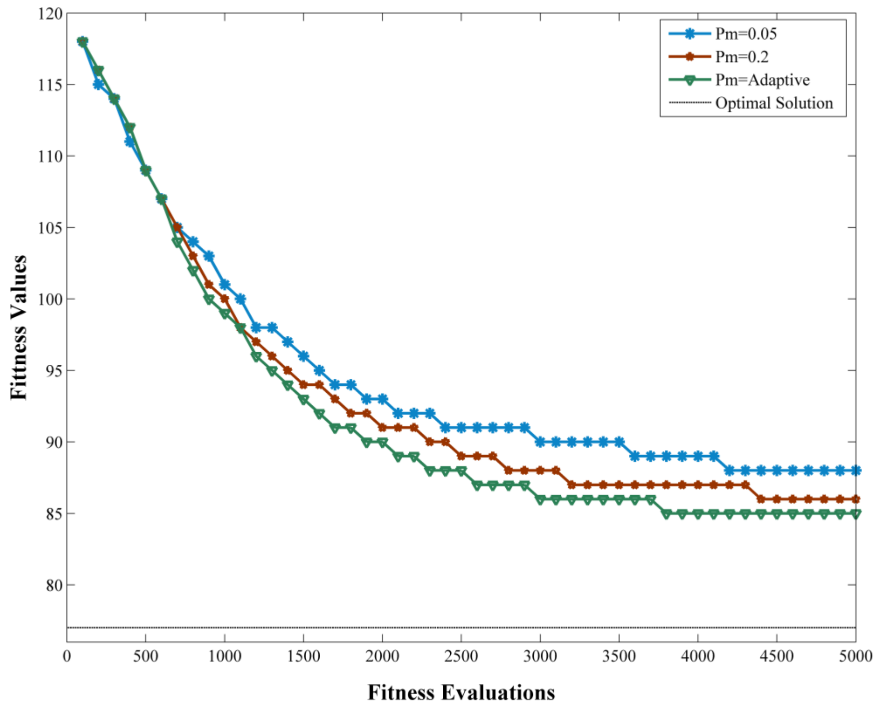


Figure 3.21: Convergence plots of J60.1-1 with different p_m values

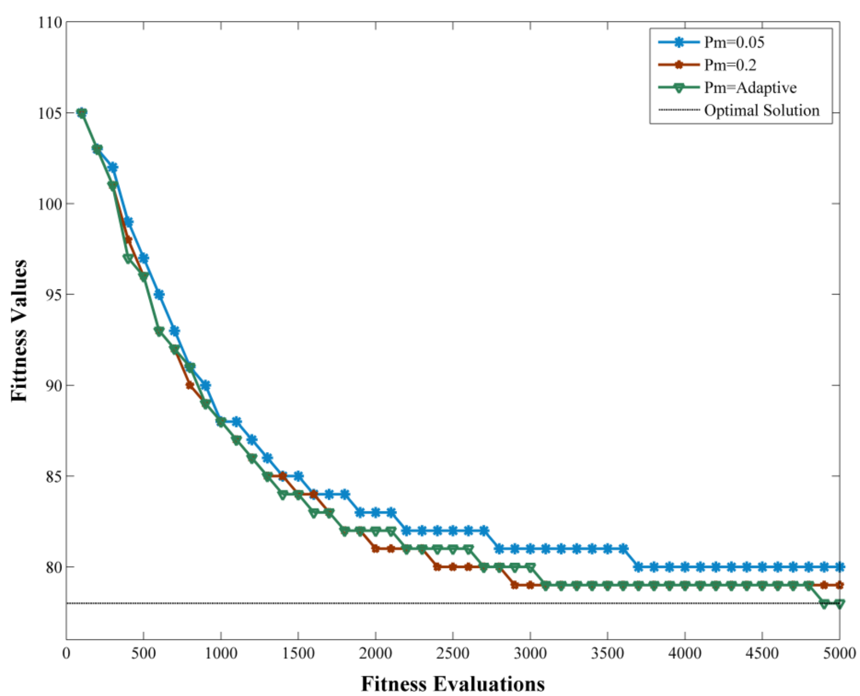


Figure 3.22: Convergence plots of J60.2-4 with different p_m values

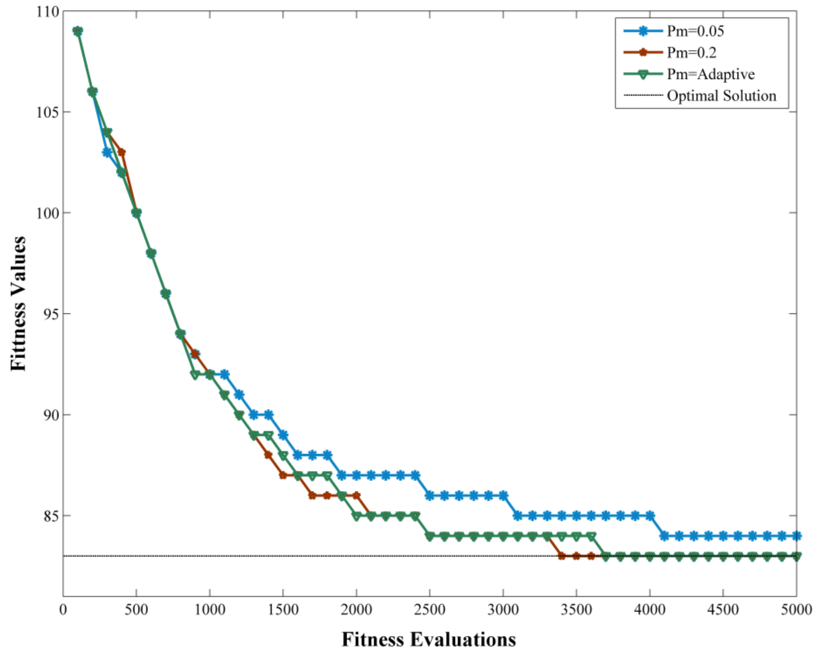


Figure 3.23: Convergence plots of J60.3-5 with different p_m values

3.3.4.4 Effect of PS

The MA was run using different values of PS of 50, 100 and 200, with Figures 3.24, 3.25 and 3.26 showing its convergence values for the three test problems. While it can be seen that the PS value of 100 outperformed the others in terms of the convergence rate, Table 3.5 shows that it was more expensive in terms of computational time.

PS	**50**	**100**	**200**
$AvgDev$(%)	4.73	**3.43**	5.71
CPU time	2.48	3	**1.95**

Table 3.5: Average deviations and CPU times of variants of PS

As a consequence, setting PS to a value of 100 was considered a good choice for obtaining better solutions quality.

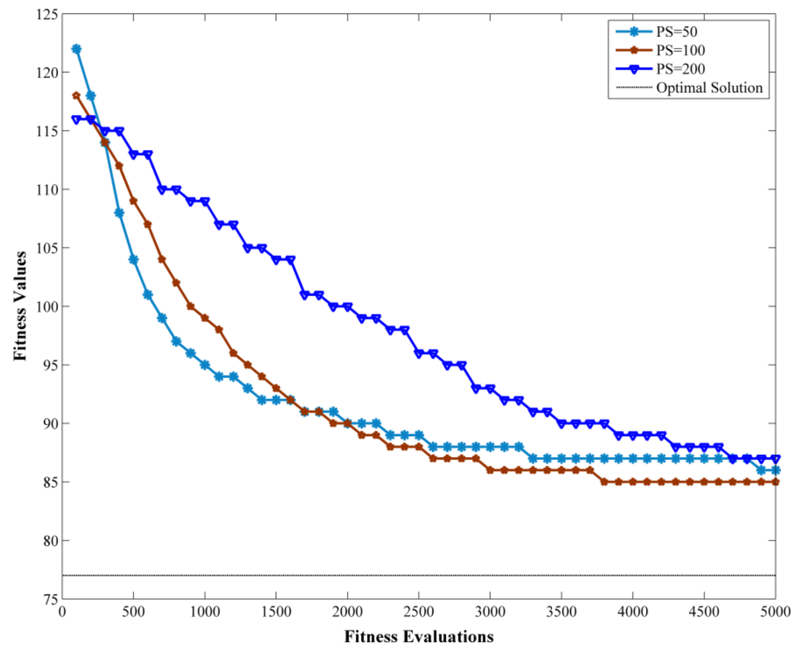


Figure 3.24: Convergence plots of J60.1-1 with different *PS* values

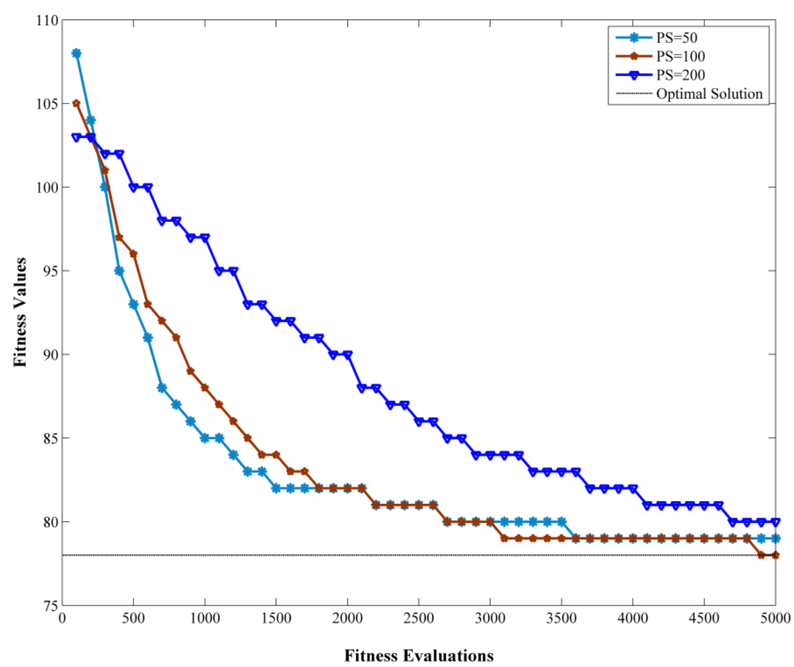


Figure 3.25: Convergence plots of J60.2-4 with different *PS* values

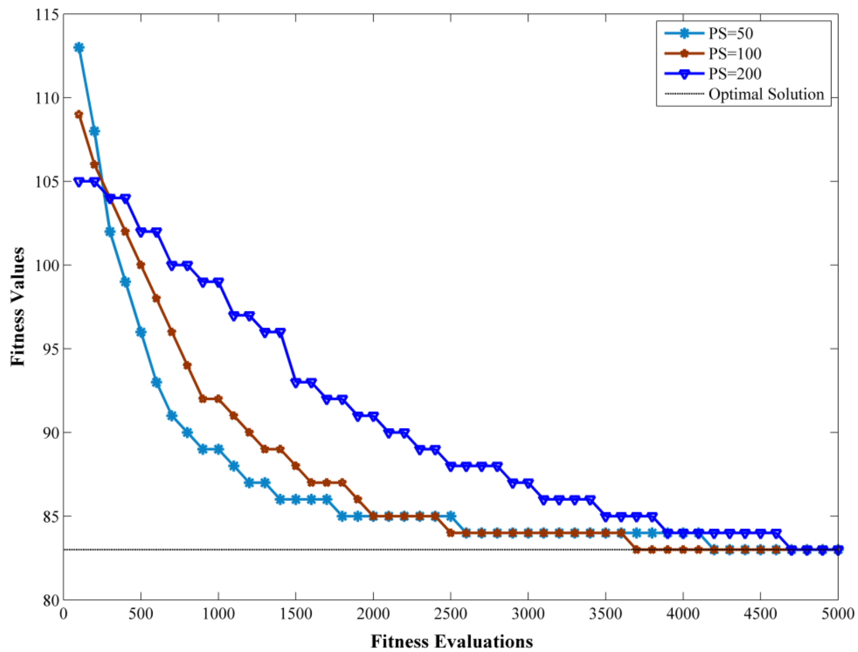


Figure 3.26: Convergence plots of J60.3-5 with different PS values

3.3.4.5 Single vs multiple local search

In this experiment, the algorithm was run with (1) the first LS, (2) second LS and (3) both LS procedures in a single framework; and (4) no LS. Figures 3.27, 3.28 and 3.29 indicate the effect of LS on the algorithm's performance.

It can be seen that the MA performed similarly for all cases but, as expected, using multiple local searches produced superior results to those of the other variants, as shown in Table 3.6.

LS types	Without LS	1st LS	2nd LS	Multi-LS
$AvgDev(\%)$	3.89	3.48	3.62	**3.43**
CPU time	2.82	3.08	2.97	**2.25**

Table 3.6: Average deviations and CPU times of variants with and without LS

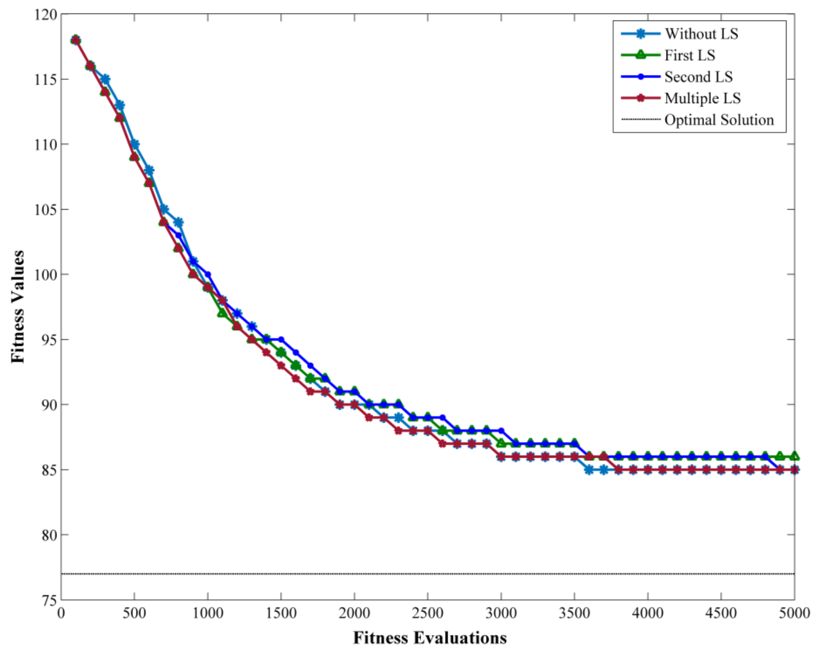


Figure 3.27: Convergence plots of J60.1-1 with single, multiple and no local searches

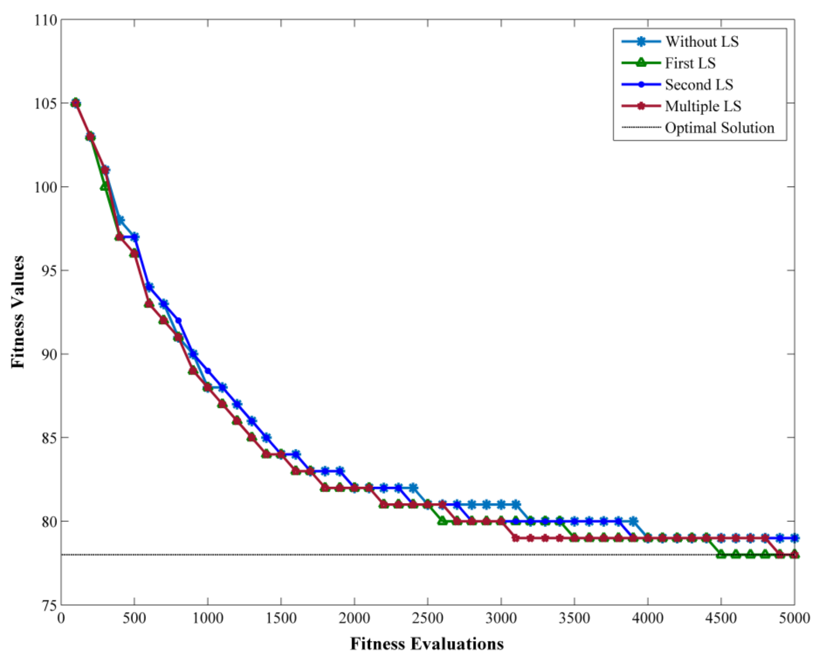


Figure 3.28: Convergence plots of J60.2-4 with single, multiple and no local searches

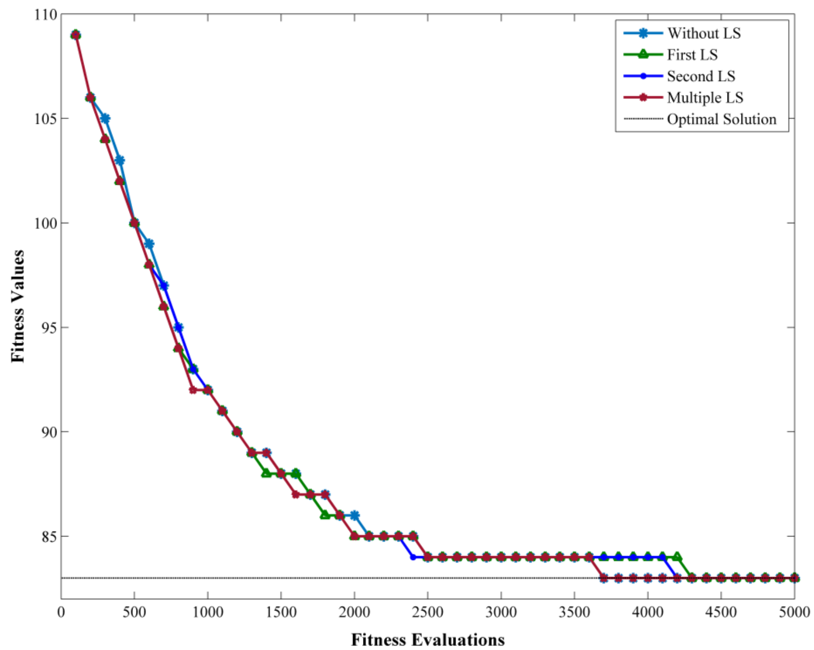


Figure 3.29: Convergence plots of J60.3-5 with single, multiple and no local searches

3.3.5 Comparison with other algorithms

In this section, a comparative study of variants of the traditional GA which were executed on the same computer configuration for the same test problems with (1) 0.2, (2) 0.05 and (3) adaptive mutation rates is discussed. Note that these variants did not use the proposed local search.

In Table 3.7, the average deviations from the critical path lower bounds are presented and it is clear that the proposed MA achieved better results for J30, J60, J90 and J120 instances than other variants of classical GAs.

Prob.	$GA_{p_{m=0.2}}$	$GA_{p_{m=0.05}}$	$GA_{adaptive\ p_m}$	MA
J30	0.0088	0.0119	0.0106	0.00
J60	4.53	3.84	3.61	3.02
J90	-	-	6.35	5.78
J120	34.97	33.86	33.21	32.94

Table 3.7: Average deviation from best solutions for benchmark instances with 30, 60 and 120 activities

As the complexity of the problems are based on NC, RF and RS values, the three groups, defined in Section 3.3.1, of the test problems were used to compare the performance of MA in each group with four other algorithms (B&B, GA, Lagrange relaxation based GA (GA_LR) and branch and cut (B&C)) proposed by Chakrabortty et al. (2015) using the same set of J30 problems. In B&B, authors applied the default exact B&B technique and solved all problems using a commercial optimization software LINGO v10.0 and then solved same instances by employing the built-in GA toolbox of Matlab (R2012b). For GA_LR, they relaxed all the equality constraints and used them as a penalty function in the objective values. Finally, authors proposed a specialized B&C approach using coin-branch & cut (CBC) solver adopted from OPTI toolbox.

Group	RS	**MA**		**B&B**		**GA**		**GA_LR**		**B&C**	
		$AD\%$	t	$AD\%$	t	$AD\%$	t	$AD\%$	t	$AD\%$	t
1	**0.20**	**0**	**3.09**	**0**	14.8	1.53	830	1.11	145.9	**0**	68.43
2	**0.50**	**0**	1.48	**0**	**1.0**	**0**	913	**0**	74.8	**0**	5.27
3	**0.70**	**0**	**0.13**	**0**	20.8	0.76	1000	0.20	460.3	**0**	2.77
Average		**0.00**	**1.57**	**0.00**	12.20	0.76	914.33	0.44	227.00	**0.00**	25.49

Table 3.8: Average deviation and CPU times of MA and other algorithms with different values of RS

From Table 3.8, it can be noticed that the proposed MA outperformed all other algorithms in terms of the average values of both average deviation from the optimal solution and the computational time. In addition, comparing the MA with both GA and GA_LR proves the effectiveness of the proposed repairing method and multiple local search techniques for improving the performance of GA in terms of quality of solution and CPU time as MA is faster than both GA and GA_LR (on average, 582 and 145 times, respectively).

3.4 Chapter Summary

During the last few decades, many exact, heuristic and meta-heuristic algorithms for solving RCPSPs have been introduced. Exact ones were found to be applicable for solving only small project instances. Whereas, heuristic methods can find near-optimal solutions at an acceptable computational cost but cannot guarantee optimal ones. Although meta-heuristic techniques are the most popular for handling combinatorial optimization problems and have consistently shown good performances, they are much more expensive than other approaches in terms of computational time.

Motivated by the mentioned gaps, this chapter presented a MA for solving RCPSP which combined GA with a heuristic repairing method and two local search techniques seeking to obtain good-quality solutions within low computational time.

The proposed MA was used to solve J30, J60, J90 and J120 instances from the PSPLIB and its results compared with those from variants of the traditional GA and some state-of-art algorithms. It was demonstrated that the MA had superior performance to all other variants of GAs when solving 61 problems with 30, 60, 90 and 120 activities, and could achieve optimal solutions for all the projects in the J30 instances and most of those in the J60, J90 and achieve low average deviation values for J120. Also, it was proven that the proposed heuristic repairing method had a great impact on the algorithm’s performance.

Based on the results obtained for the J30 test problems, MA demonstrated its competitive performance against four state-of-the-art algorithms in terms of the computational time and quality of solutions, which provided motivation for testing the adaptation of the heuristic repairing method for other evolutionary algorithms, i.e., DE, which is introduced in the next chapter.

Chapter 4

Differential Evolution for solving RCPSP

In chapter 3, the hybridization of a genetic algorithm (GA) with the proposed heuristic repairing technique has been discussed and experimented. In this chapter, the same concept is considered with differential evolution (DE) algorithm. Firstly, the algorithm framework is discussed and then its components are introduced and experimental details are presented. The influence of each component on its performance is also investigated. Finally, the results are compared with those obtained from some state-of-the-art algorithms.

4.1 Introduction

In the previous chapter, a memetic algorithm based on GA and multiple local searches for solving a resource constrained project scheduling problems (RCPSPs) was introduced. Although GA was able to obtain good results for many problems in comparison with those of other algorithms, it had a tendency to converge towards local optima or even arbitrary points rather than the global optimum of the problem. DE is a well-known population-based stochastic search technique that proved to be effective approach for solving global optimization problems (Vesterstrøm and Thomsen, 2004). Motivated by these facts, in this chapter, a hybrid technique which combines the search capabilities of DE and heuristic repairing method (as discussed in Chapter 3) for solving a RCPSP is introduced.

The proposed hybrid DE was tested by solving 60 standard benchmark problems, 15 with 30 activities and 15 each with 60, 90 and 120 activities. As in the previous chapter, these problems were chosen from the well-known test set library, the project scheduling problems library (PSPLIB) initiated by (Kolisch et al., 1999). The effects of the proposed DE algorithm's components, such as (1) the repairing rate, which is the number of individuals/solutions repaired by the proposed heuristic method to be feasible ones (R_m), (2) crossover rate, (3) mutation rate and (4) population size, on its performance are discussed. Then, the results obtained from its final variant are compared with those from different state-of-the-art algorithms.

The rest of this chapter is organized as follows. In section 4.2, the methodology of the proposed algorithm and its components are described. In section 4.3, the experimental study and analyses of the different control parameters of the proposed DE are discussed. Finally, a summary of the chapter is provided in section 4.4.

4.2 Methodology

In this chapter, a DE-based approach for solving RCPSPs, in which an initial population is generated randomly and then the repairing method applied to selected individuals to convert them from infeasible to feasible solutions, is proposed. All individuals are sorted according to their fitness and violation values and then a tournament selection is used to choose the best ones to act as parents for the next generation. Thereafter, improved mutation and crossover operators are applied with the aim of maintaining feasibility for the generated individuals. These processes are continued until pre-defined stopping conditions are met.

The framework of the algorithm is presented in Figure 4.1 and its components briefly discussed in this section.

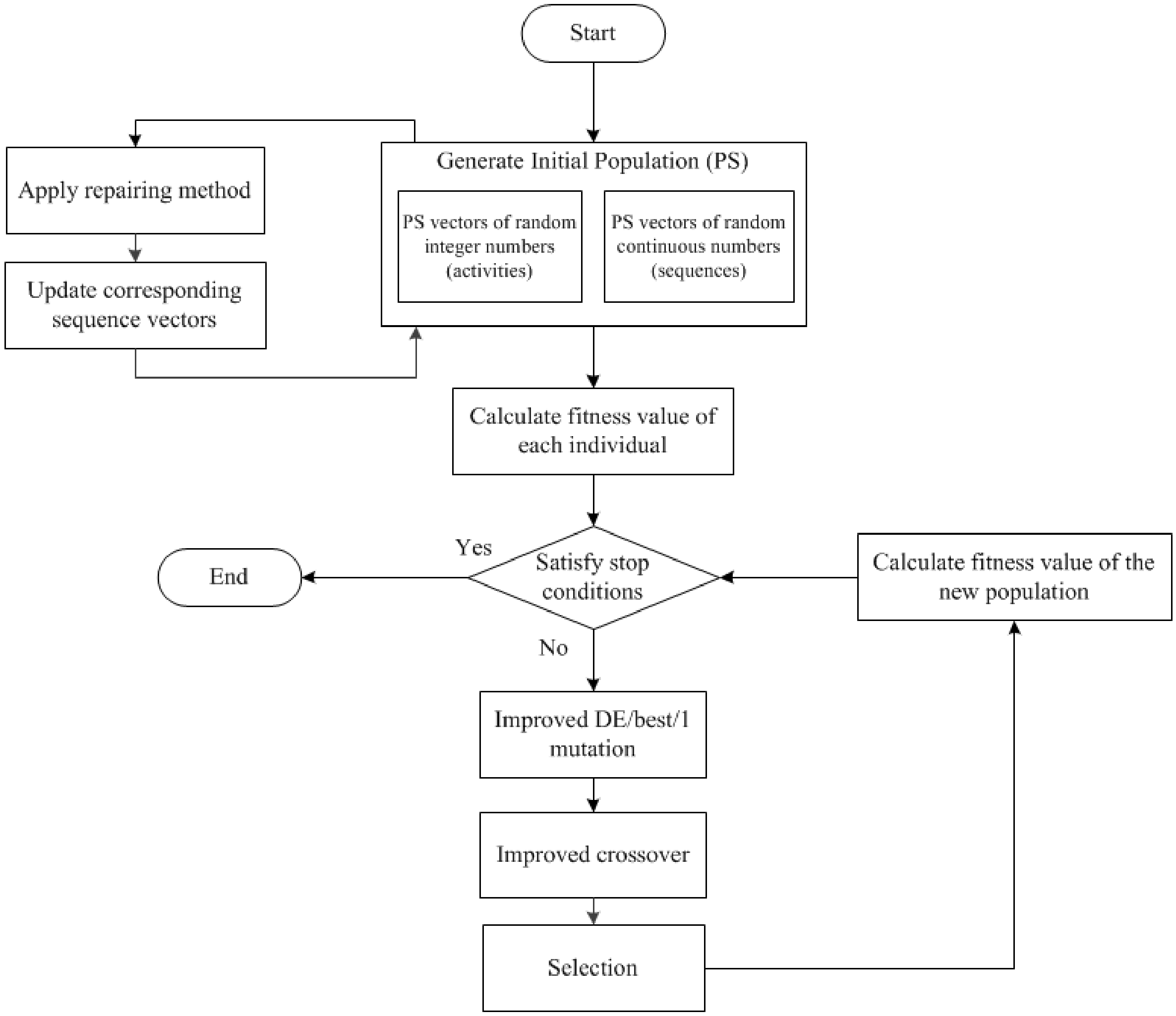


Figure 4.1: General framework of proposed DE

4.2.1 Chromosome representation

As described in Chapter 3, in the proposed DE algorithm, each individual is represented by a vector of integer values, the length of which equals the number of activities in the project (n). An example of the representation of an individual (chromosome) and the representations of the precedence constraints can be found in Section 3.2.1.

As DE was originally proposed to deal with the continuous space, the following representation is proposed to make it suitable for the discrete nature of RCPSPs. In it, n random vectors of continuous numbers are generated in the range of [0, 1], so that each integer value has a corresponding continuous value that determines its appearance in the schedule. Figure 4.2 presents an example of this representation.

Chromosome	0	1	2	4	3	…	n	$n+1$
Sequence	0	0.02	0.1	0.14	0.25	…	0.95	1

Figure 4.2: Randomly generated sequence for one individual

In Figure 4.2, two vectors had been generated with integer and continuous values where the Chromosome/integer vector represents the numbers of activities and the Sequence/continuous one represents the execution order of each activity in the project.

4.2.2 Fitness evaluations

As in Chapter 3, the fitness value and/or constraint violations are used to evaluate a solution. The fitness value is calculated based on time required to complete all activates not violating the resource availability constraint. In this mechanism, the activities of candidate solutions are scheduled by their order (sequences) in the generated schedule. Each activity can be processed if, and only if, its required number of resources does not exceed the available amount of resources at a specific time so that the schedule produced is guaranteed to satisfy the resource availability constraint. On the other hand, the violation value of each solution is determined by calculating the number of violations of the dependency constraint by each activity in the schedule.

4.2.3 Repairing method

The heuristic repairing method already described in Chapter 3 is used to reduce the violation values of some selected individuals to zero; in other words, it is applied to enhance the feasibility of these individuals to make them feasible solutions. In its mechanism, each activity in an individual satisfies its precedence relationships. From our observations, it is noted that providing feasible solutions in the initial population may significantly increase the convergence rate of the proposed algorithm.

4.2.4 Improved DE operators

After calculating the fitness and constraint violations of each individual in the initial population, the DE begins to iteratively apply its different operators (mutation, crossover and selection) to the generated individuals to evolve them. The proposed mutation and crossover operators guarantee the feasibility of any newly generated individual as follows.

4.2.4.1 Mutation

In the improved DE/best/1 mutation, the mutant vectors ($\vec{v}_i$) are produced using the individuals according to:

$$\vec{v}_{i,g+1} = \vec{x}_{\lambda,g} + F \times (\vec{x}_{r1,g} - \vec{x}_{r2,g}) \tag{4.1}$$

which is the same as that in equation (2.6) in Chapter 2 except that $\vec{x}_\lambda$ is selected from the top 10% of solutions in the current population (g), $r_1, r_2 \in [1, PS]$ are randomly selected integer numbers which are not equal to either one another or the target individual (i) and F is the mutation scale factor. These mutant vectors are guaranteed to produce a feasible solution by applying a proposed approach aiming at changing the sequence of each activity in an individual to satisfy the conditions of its predecessors' and successors' activities. The proposed approach is described using the pseudo-code in Figure 4.3

```
For j = 1 to n do
   Find Pre_j; the predecessors of current gene (j)
   Calc. new_seq(j); the new sequence value of j
   If all(Pre_j) are already scheduled, then
      seq_j (the current sequence value of j) ← new_seq(j)
   Else
      For i = 1 to end of Pre_j do
         If seq(Prej) >= new_seq(j), then
            Swap seq(Prej) with new_seq(j)
         End if
      End for
   End if
End for
```

Figure 4.3: Proposed approach for obtaining feasible solutions from mutation and crossover

4.2.4.2 Crossover

Then, the binomial crossover is used to produce trail vectors ($u_{i,g+1}^{j}$) according to:

$$u_{i,g+1}^{j} = \begin{cases} v_{i,g+1}^{j}; & rand(j) \leq CR \text{ } or \text{ } j = a_j \\ x_{i,g}^{j}; & otherwise \end{cases} \tag{4.2}$$

where $i = 1,2, \dots, PS$; $j = 1,2, \dots, n$; n the number of activities in the project; CR is the crossover possibility in the range of [0,1], $rand(j)$ the jth evaluation of a uniform random number generator within [0,1] and a_j a randomly selected dimension to ensure that at least one element of $u_{i,g+1}^{j}$ is chosen from the mutant vectors.

Also, the sequence of each activity is re-arranged to satisfy the constraints of the predecessors' and successors' activities according to Figure 4.3.

4.2.4.3 Selection

Finally, for the selection process, the greedy selection strategy is adapted to the individuals to decide which from the trial vectors can survive to the next generation based on both their fitness and violation values according to:

$$\vec{x}_{i,g+1} = \begin{cases} \vec{u}_{i,g+1}, & f(\vec{u}_{i,g+1}) < f(\vec{x}_{i,g}) \\ \vec{x}_{i,g}, & otherwise \end{cases} \tag{4.3}$$

where $i = 1,2, \dots, PS$ and $\vec{u}_{i,g+1}$ the i mutant vector in the new generation (g+1).

4.3 Experimental study

The proposed DE algorithm was coded using Matlab R2013b and implemented on a PC with a 3.4 GHz CPU and Windows 7.

In this section, the computational results for 16 problems of J30 instance and 15 from each of J60, J90 and J120 ones, that is, a total of 61 problems, randomly selected from three different instances from the well-known standard benchmark test set library PSPLIB for testing DE in different complexity levels, with four types of resources used in each are discussed. Also, in order to judge the performance of the proposed algorithm, comparisons with state-of-art algorithms are also conducted. The benchmark problems and their complexity factors are explained in Chapter 3 (Section 3.3.1).

Usually, the average percentage deviations ($AvgDev(\%)$) from optimal solutions for J30 instances or from the critical path lower bounds for J60, J90 and J120 as reported by Stinson et al. (1978) are considered as a performance metric for comparison. Generally, the lower value of $AvgDev(\%)$, which can be calculated by equation (4.4), means obtaining higher quality solution.

$$AvgDev(\%) = \frac{1}{S} \times \sum_{s=1}^{S} \frac{BS_s - LB_s}{LB_s} \times 100 \tag{4.4}$$

where S is the total number of instances used, BS_S the best solution achieved by an algorithm for S instances and LB_S the pre-known lower bound of a s instance.

4.3.1 Parameter settings

The parameters of the proposed algorithm were set as follows: Fit_{Max}, the maximum number of calls of the fitness function evaluation (cfe), to values of 5000 and 50,000; R_m, the number of individuals to be repaired to be feasible using the proposed repairing method (Chapter 3, Section 3.2.3), to a value of 25% of PS; PS to 100; and F to 0.1.

CR was calculated adaptively using equation (4.5), where CR_{LB} =0.1 and CR_{UB} =0.9, to obtain a balance between a good initial value and the speed of convergence.

$$CR = CR_{LB} + CR_{UB} \times \frac{cfe}{Fit_{Max}} \tag{4.5}$$

Justifications of a selection of these parameters are discussed in the Section 4.3.3 by providing analysis of each parameter with different values.

4.3.2 Computational results

For each test problem, 30 independent runs were executed, with two stopping criteria, that is, (1) Fit_{Max} was reached or (2) no improvement in the fitness value was achieved for 150 consecutive generations and same parameter settings mentioned in Section 4.3.1.

The $AvgDev(\%)$ from the optimal solutions for the J30 instances and lower bounds for the J60, J90 and J120 ones ($AD\%$), as well as its standard deviation (STD) and the average CPU times in seconds (t), are given for 5,000, 50,000 and n ×10,000 maximum numbers of generations in Table 4.1.

No. of generations	J30			J60			J90			J120		
	$AD\%$	STD	t	$AD\%$	STD	t	$AD\%$	STD	t	$AD\%$	STD	t
5,000	0	0.16	16.13	6.19	1.08	41.86	7.17	1.81	72.12	50.88	3.00	106.71
50,000	0	0.09	24.18	3.34	0.70	192.26	6.88	0.83	306.89	35.46	0	485.66
n ×10,000	0	0	37.81	2.99	0	269.45	5.72	0	890.47	32.61	0	2636.9

Table 4.1: Results of proposed DE for J30, J60, J90 and J120 instances with 5,000, 50,000 and n ×10,000 max generations

From Table 4.1, it is clear that the performance of the proposed DE, in terms of solutions-quality and the standard deviation values for all instances, improved and the computational time significantly increased with increasing the number of generations.

4.3.3 Parametric analysis

Four sets of experiments were designed to analyze the effect of A) R_m, number of solutions repaired to be feasible, B) CR, C) F and D) PS on the performance of the proposed DE using the same data set with 15 test problems from the J60 instances used in the analysis in Chapter 3. Also, one problem was selected from three different instances with different complexity factors (Chapter 3) to study the effect of each parameter in various complexity environments. The results for the three chosen problems, 'J60.1-1, J60.2-4 and J60.3-5' are shown in this section with the convergence plots of each parameter given. For all the experiments, the proposed DE was executed for up to 25 runs with 5000 generations.

The selection of the parameters is done in a sequential manner in which the best parameter found in an experiment is fixed in the subsequent ones.

4.3.3.1 Effect of R_m

In the first experiment, the effect of the repairing method introduced in Chapter 3 on the performance of the proposed DE was investigated. In it, DE was run with different values of R_m of (1) R_m= 0% of PS, (2) R_m= 25% of PS, (3) R_m= 50% of PS, and (4) R_m= 100% of PS, with all other parameters fixed, as previously described in Section 4.3.1.

To provide an indication of the effect of R_m, the convergence plots of the J60.1-1, J60.2-4 and J60.3-5 problems are shown in Figures 4.4, 4.5 and 4.6, respectively. It is clear that using a repairing method in the initial population improved the performance of the algorithm in terms of convergence towards optimal or near-optimal solutions and thereby reduced the computational time, as shown in Table 4.2. However, it can be noted that, while applying the repairing method to a small percentage of the whole population, such as 25%, could significantly improve the solution quality, by increasing the value of R_m to 50% or 100%, the performance of the proposed DE might be degraded because repairing a large number of individuals may cause the algorithm to be confined in a local optimum.

The $AvgDev$(%) and CPU times of the test problems are presented in Table 4.2 which shows that R_m=25% was not the best in terms of CPU time but produced better-quality solutions than the others. Therefore, R_m was set to 25%.

$\boldsymbol{R_m}$	**0%**	**25%**	**50%**	**100%**
$\boldsymbol{AvgDev}$(%)	6.25	**4.44**	4.91	5.82
CPU time	72.69	71.90	67.06	**64.97**

Table 4.2: Average deviations and CPU times of variants of R_m

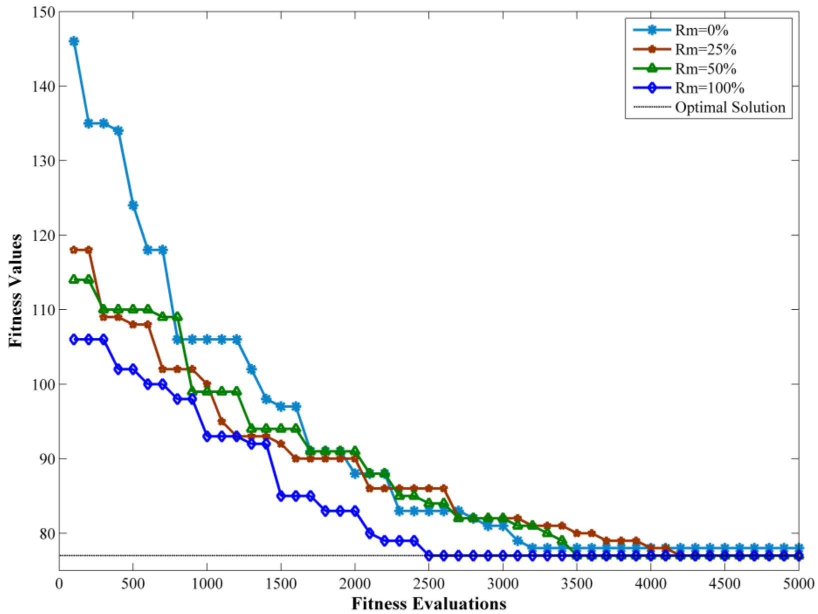


Figure 4.4: Convergence plots of J60.1-1 with different R_m values

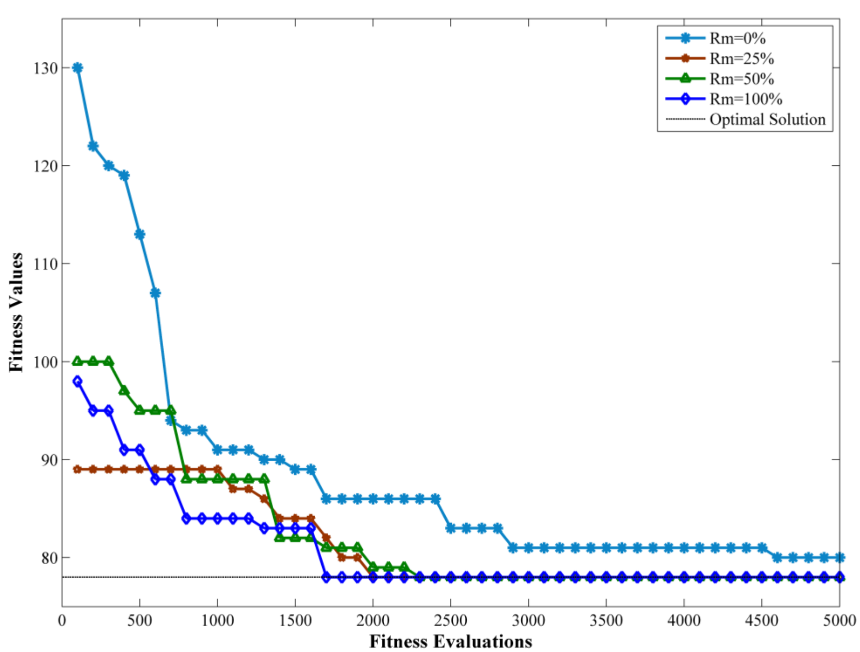


Figure 4.5: Convergence plots of J60.2-4 with different R_m values

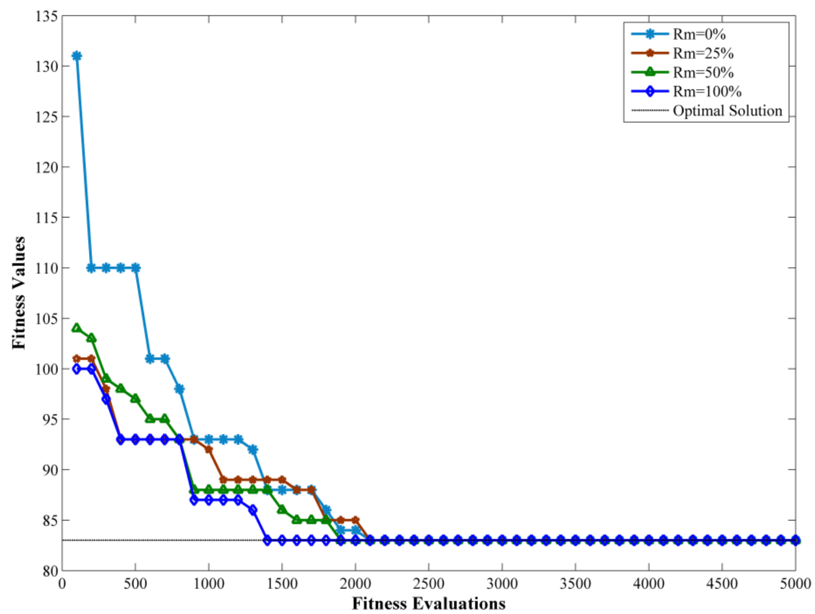


Figure 4.6: Convergence plots of J60.3-5 with different R_m values

4.3.3.2 Effect of CR

In Storn and Price (1997), it was mentioned that *CR*=0.1 was a good initial choice for the crossover rate while CR=0.9 or 1.0 could be used to try to increase the convergence speed. Therefore, the second experiment studied the effect of the crossover operator by running the proposed DE using different values of CR of (1) CR=0.1, (2) CR= 0.9 and (3) CR adaptively reduced from 0.9 to 0.1, with the best variant found in the previous subsection.

Figures 4.7, 4.8 and 4.9 present the convergence plots of the J60.1-1, J60.2-4 and J60.3-5 problems, respectively, in order to illustrate the effect of CR on DE's performance. It can be noted that the algorithm with CR=0.9 could easily become stuck in a local optimum solution even if the problem was not complex, as in Figure 4.9. On the other hand, when CR was calculated adaptively between 0.9 and 0.1, it outperformed both CR=0.1 and 0.9 in terms of solution quality.

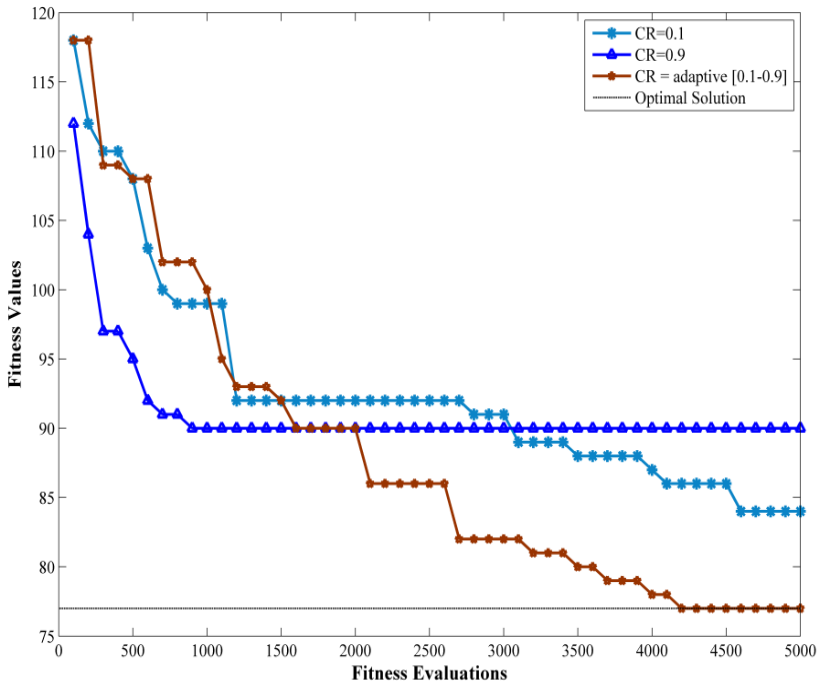


Figure 4.7: Convergence plots of J60.1-1 with different CR values

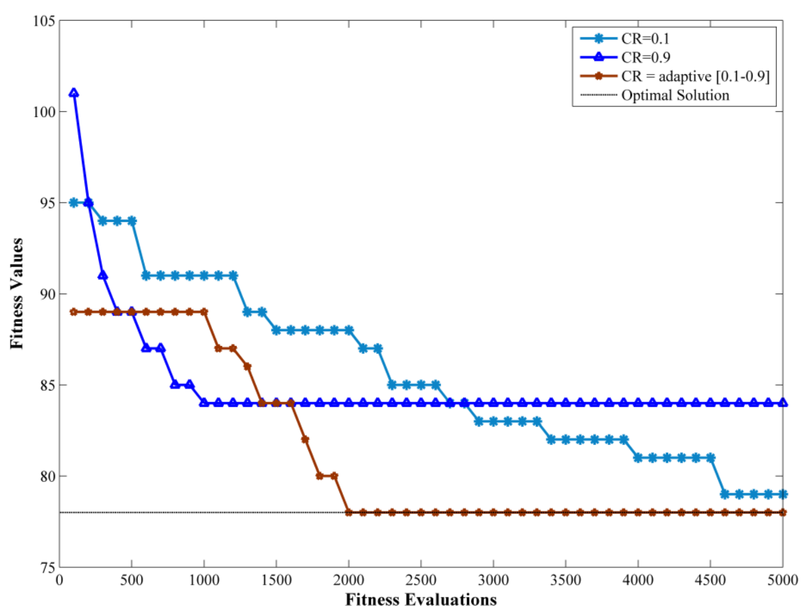


Figure 4.8: Convergence plots of J60.2-4 with different CR values

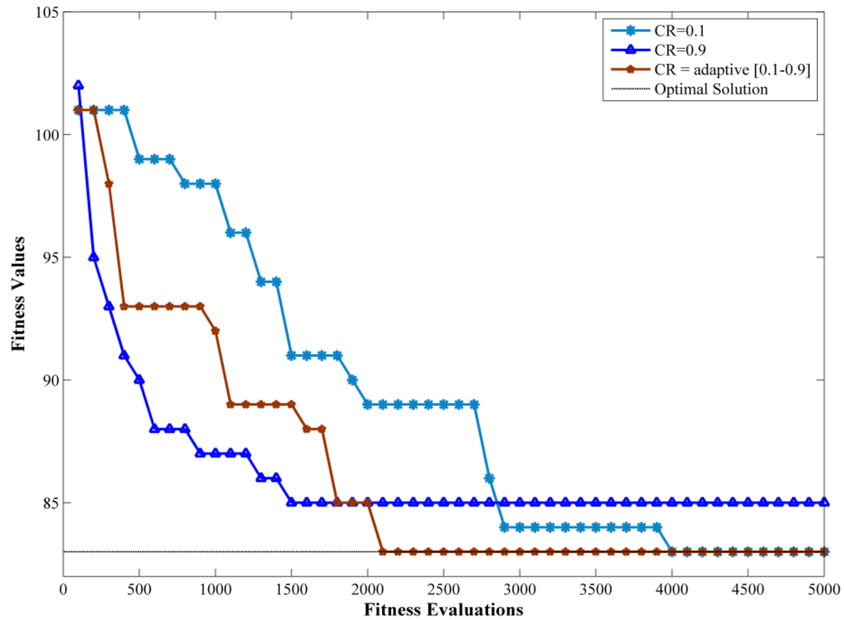


Figure 4.9: Convergence plots of J60.3-5 with different CR values

Table 4.3 shows the $AvgDev(\%)$ and computational times using different values of CR which clearly demonstrates that the adaptive CR outperformed all other variants in terms of both parameters.

$\boldsymbol{CR}$	**0.1**	**0.9**	**Adaptive [0.1-0.9]**
$\boldsymbol{AvgDev}(\%)$	11.01	15.59	**4.44**
CPU time	116.13	122.47	**70.18**

Table 4.3: Average deviations and CPU times of variants of $\boldsymbol{CR}$

4.3.3.3 Effect of F

To further study the effect of the F on the performance of the proposed DE algorithm, the data set was solved using different values of F of (1) F=0.1, (2) F=0.5, (3) F=0.9 and (3) F adaptively reduced from 0.9 to 0.1 and best variant of parameters found in the previous subsection.

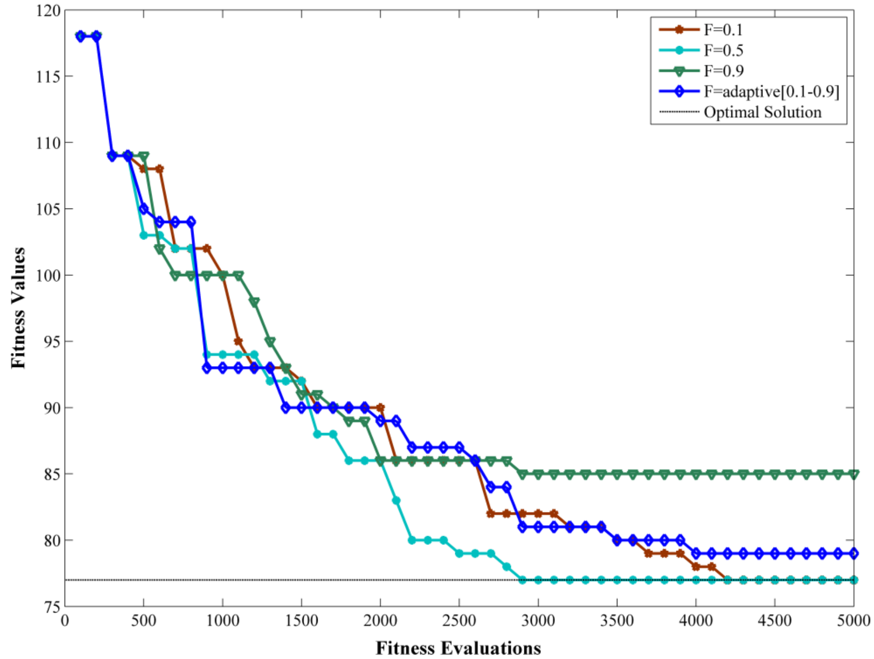


Figure 4.10: Convergence plots of J60.1-1 with different *F* values

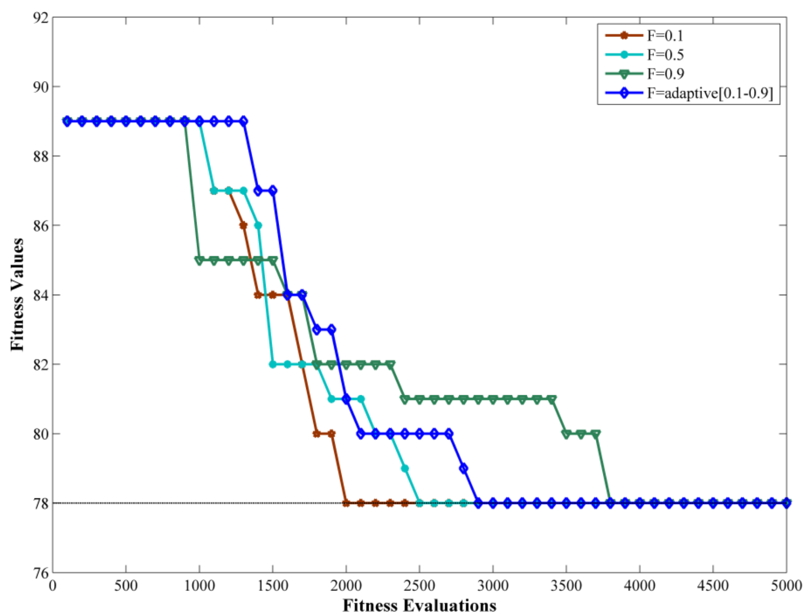


Figure 4.11: Convergence plots of J60.2-4 with different *F* values

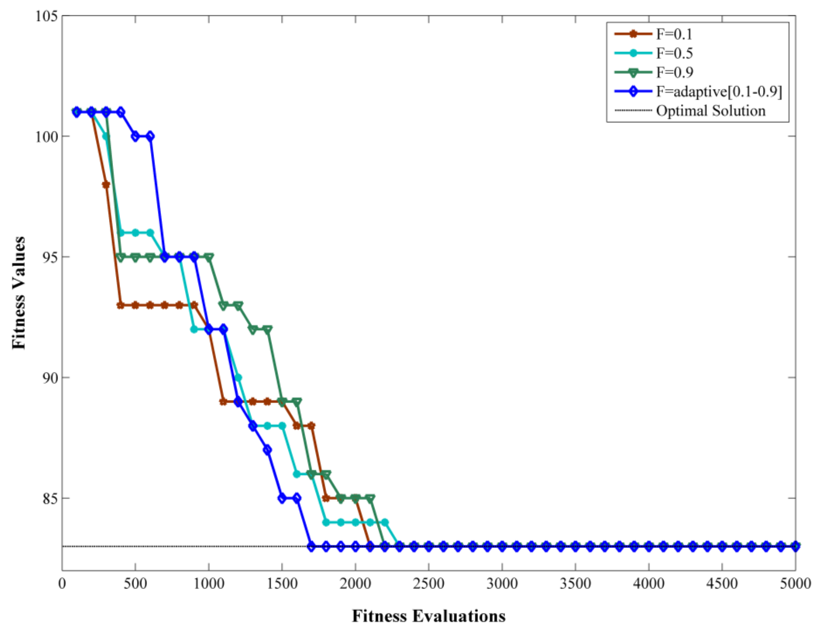


Figure 4.12: Convergence plots of J60.3-5 with different *F* values

Figures 4.10, 4.11 and 4.12 show that the performance of the proposed DE was improved by reducing the value of F; for instance, for the most complex problem (Figure 4.10), only F =0.1 and 0.5 achieved optimal solutions.

Table 4.4 demonstrates that F=0.1 had the best $AvgDev(\%)$ and CPU times of all the F values.

F	**0.1**	**0.5**	**0.9**	**Adaptive [0.1-0.9]**
$\boldsymbol{AvgDev}(\%)$	**4.44**	5.45	7.71	4.94
CPU time	**70.18**	70.61	81.56	88.09

Table 4.4: Average deviations from best solutions and average CPU times of proposed DE with different mutation scale factor values

4.3.3.4 Effect of PS

In the final experiment, the population size (PS) parameter was analyzed to study its effect on the performance of the proposed DE.

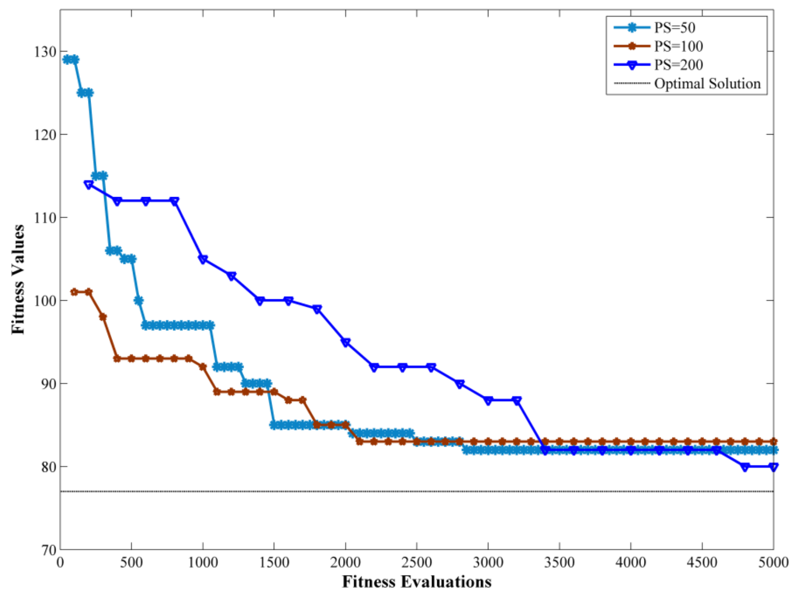


Figure 4.13: Convergence plots of J60.1-1 with different *PS* values

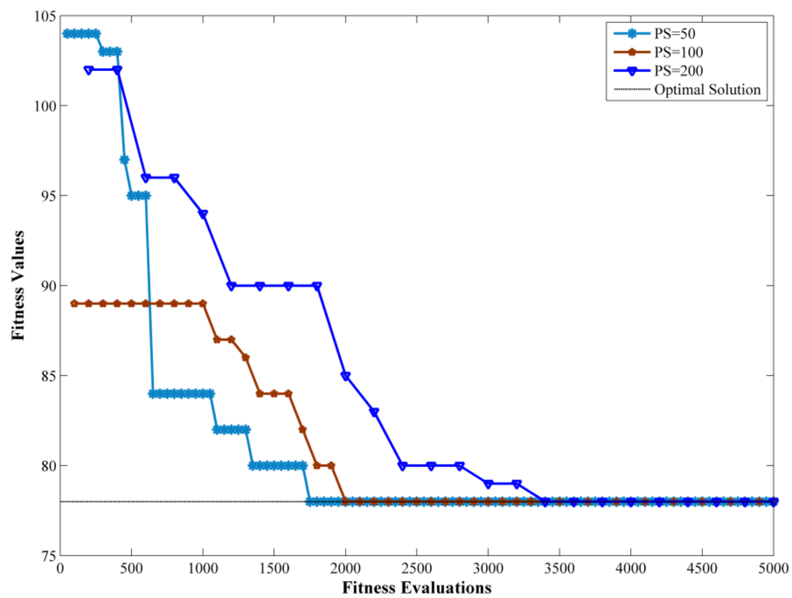


Figure 4.14: Convergence plots of J60.2-4 with different *PS* values

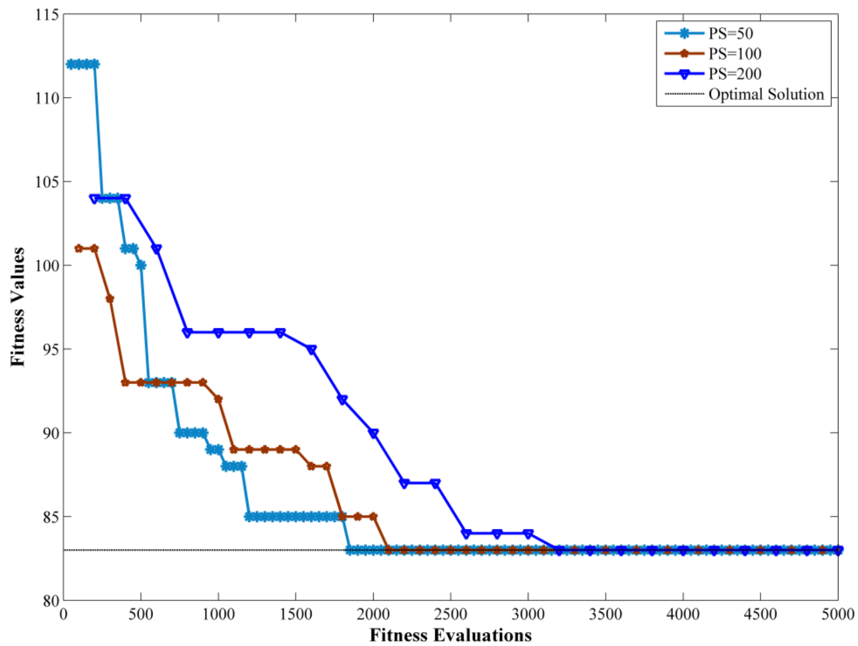


Figure 4.15: Convergence plots of J60.3-5 with different PS values

The algorithm was run using different values of PS= 50, PS=100 and PS=200. Figures 4.13, 4.14 and 4.15 show the convergences of DE for the selected data set from which it is clear that PS with a value of 100 outperformed the other PS values in terms of the convergence rate. Also, Table 4.5 demonstrates that PS=100 was much faster in terms of computational time.

PS	50	100	200
$AvgDev$(%)	6.36	**4.44**	5.54
CPU time	71.55	**70.18**	77.11

Table 4.5: Average deviations and CPU times of variants of PS

Therefore, PS was set to 100 as it produced good results in terms of both computational time and solution quality.

4.3.4 Comparison with other algorithms

In this section, a comparative study of, firstly, four algorithms introduced by Chakrabortty et al. (2015) , and secondly, three variants of the traditional GA with (1) 0.2, (2) 0.05 and (3) adaptive mutation rates, traditional DE and the proposed MA (discussed in Chapter 3) which were executed on the same computer configuration for the same dataset of problems is conducted in Tables 4.6 and 4.7, respectively.

Grouped by the values of the resource strength (RS) complexity factor (discussed in Chapter 3), Table 4.6 shows the computational times (t) and the average deviation from optimal solutions for test problems of three different instances randomly selected from J30 with three different values of RS complexity factor: 0.20, 0.50 and 0.70 and same values of both network complexity (NC) and resource factor (RF) with values 1.50 and 0.25.

Group	RS	**Improved DE**		**MA**		**B&B**		**GA**		**GA_LR**		**B&C**	
		$AD\%$	t	$AD\%$	t	$AD\%$	t	$AD\%$	t	$AD\%$	t	$AD\%$	t
1	**0.20**	**0**	49.90	**0**	**3.09**	**0**	14.8	1.53	830	1.11	145.9	**0**	68.43
2	**0.50**	**0**	55.02	**0**	1.48	**0**	**1.0**	**0**	913	**0**	74.8	**0**	5.27
3	**0.70**	**0**	15.36	**0**	**0.13**	**0**	20.8	0.76	1000	0.20	460.3	**0**	2.77
Average		**0**	37.81	**0**	**1.57**	**0**	12.20	0.76	914.33	0.44	227.00	**0**	25.49

Table 4.6: Average deviations and CPU times of proposed DE and other algorithms for J30 with different values of RS

From this table, it is clear that the improved DE achieved the optimal solutions for all solved instances. Although, it was slower than some algorithms such as MA and B&B, it steadily converged towards the optimal solutions with standard deviations equal zero, which means that the optimal solutions were obtained in each run of the algorithm for all problems, while the standard deviations of MA was 0.6.

For more judging on the performance of the proposed DE, its results compared with those obtained from different variants of classical GA and DE. In Table 4.7, the $AvgDev(\%)$ values from optimal solutions for J30 instance and the lower bound calculated by critical path method (CPM) for J60, J90 and J120 ones of all the comparative algorithms are listed. From this table, it is clear that, the proposed algorithm was able to obtain the best quality of solutions among all classical ones. The results demonstrate the effectiveness of the improved DE as it enhanced the performance of the traditional DE, which can be calculated using equation 4.6, by 20.77%, on average, and obtained better quality solutions than both classical variants of GA and MA by decreasing the $AvgDev(\%)$ values by 7.23% and 0.76%, respectively.

$$Rate \ = \sum_{y=1}^{4} \left(\frac{AD\%_w - AD\%_q}{AD\%_w} \right) \times 100 \tag{4.6}$$

where $y =$ {J30, J60, J90, J120}, $AD\%_q$ is the $AvgDev(\%)$ of the proposed algorithm, improved DE, and $AD\%_w$ is the $AvgDev(\%)$ obtained by other algorithms or variants.

Prob.	**GA**$_{p_{m=0.2}}$	**GA**$_{p_{m=0.05}}$	**GA**$_{adaptive\ p_m}$	**DE**	**MA**	**Improved DE**
J30	0.0088	0.0119	0.0106	0.43	**0.00**	**0.00**
J60	4.53	3.84	3.61	6.25	3.02	**2.99**
J90	-	-	6.35	7.09	5.78	**5.72**
J120	34.97	33.86	33.21	36.71	32.94	**32.61**

Table 4.7: Average deviations of proposed DE and state-of-the-art algorithms

In order to study the difference between any two stochastic algorithms, a statistical significant testing is performed by applying the Wilcoxon Signed Rank Test (Corder and Foreman, 2009) which can be used to judge the difference between paired scores as an alternative to the paired-samples t-test, when the population cannot be assumed to be normally distributed.

It is assumed that there is no significant difference between the best and/or mean values of two samples as a null hypothesis, the number of test problems N=16 for J30 and 15 for each J60, J90 and J120, and 90% confidence level. Based on the test results, three signs ($+$, $-$, and $\approx$) are assigned for the comparison of any two algorithms where "$+$" sign means the first algorithm is significantly better than the second, "$-$"sign means that the first algorithm significantly worse, and "$\approx$"sign means that there is no significant difference between the two algorithms (Elsayed et al., 2011b). The results based on the average deviation from the critical path values (LB_{CP}) and from the known optimal solutions (LB_{OP}) are presented in Table 4.8.

Algorithms	**Instance**	**Criteria**	***p*- Value**	**Decision**
Improved_DE - to - MA	J30	LB_{CP}	1.000	≈
		LB_{OP}	1.000	≈
	J60	LB_{CP}	1.000	≈
		LB_{OP}	1.000	≈
	J90	LB_{CP}	0.463	≈
		LB_{OP}	0.012	−
	J120	LB_{CP}	0.975	≈
		LB_{OP}	0.950	≈

Table 4.8: Wilcoxon Signed Rank Test for MA and Improved DE

From Table 4.8 and according to the p-values, it is clear that there is no significant difference between the MA and the improved DE, except for J90 as the p-value indicates that DE performed worse than MA in regarding to LB_{OP}.

4.4 Chapter Summary

In this chapter, a new strategy for improving the performance of DE was presented. It incorporated the proposed validation procedure (Chapter 3), which provided feasible solutions in the initial population, and improved DE operators which forced the direction of DE's search towards feasibility.

The main contributions in this chapter can be summarized as: (1) proposing a chromosome representation for dealing with the discrete nature of a problem through a continuous space; and (2) improving DE's mutation and crossover operators to produce feasible mutant and trail vectors which guarantee the feasibility of any individual they generate.

The numerical experiments on a well-known benchmark data set with 30, 60, 90 and 120 activities showed that the proposed algorithm was able to achieve optimal solutions for all the J30 instances and very low average deviation values for the J60, J90 and J120 ones.

Despite the well-known advantages of DE, its several drawbacks, including not guaranteeing convergence to the global optimum (Jia et al., 2011). Moreover, because of the existence of its inherent operations, such as its encoding scheme used to represent permutations as vectors and its process of redefining those vectors as solutions, the experimental study demonstrated that the degree of computational complexity of combinatorial problems for a DE algorithm was greater than that for a GA. Therefore, the computational time required by DE to converge towards the optimal solution significantly increased when the problem size increased.

Motivated by these conclusions, a bi-evolutionary algorithm which combines the proposed heuristic repairing method and good features of both MA and DE based on the experimental results shown in Chapters 3 and 4 is introduced and discussed in the next chapter.

Chapter 5

Bi-evolutionary Algorithm for solving RCPSPs

In this chapter, for solving resource-constrained project scheduling problems (RCPSPs), a bi-evolutionary algorithm (bi-EA) that combines two population-based algorithms GA and DE in one algorithmic framework is introduced. The effects of control parameters on the performance of the algorithm are examined through a parametric analysis. Finally, the experimental results are compared with those from the proposed MA and DE techniques, discussed in previous chapters, and state-of-the-art algorithms are provided.

5.1 Introduction

In the previous two chapters, two population-based algorithms for solving RCPSPs, MA and DE, integrated with a new heuristic repairing method were proposed. Despite their good results, experimental studies showed that they had the following drawbacks.

- In many problems, although MA was able to obtain good results, in comparison with other algorithms, it had a tendency to converge towards local optima or even arbitrary points rather than the global optimum of the problem.
- DE's performance deteriorated as the dimensionality of the search space increased (Das et al., 2009).
- Although DE was good at exploring the search space, it was slow at exploiting the solution (Noman and Iba, 2008).
- The computational time of DE significantly increased with larger problem sizes due to its inherent operations, such as encoding permutations into vectors and redefining those vectors into solutions, as described in Chapter 4.

In order to overcome these shortcomings, an algorithm for solving RCPSPs by combining the search capabilities of MA and DE in one framework is proposed. This new bi-evolutionary algorithm begins by dividing the initial population into two sub-populations, one of which deals with the integer search space with MA and another the continuous search space with DE. The repairing method discussed in Chapter 3 is also applied to the individuals in both sub-populations in order to enhance their feasibility.

The algorithm is tested on a set of instances with 30, 60, 90 and 120 activities taken from the well-known project scheduling problems library (PSPLIB) (Kolisch et al., 1999).

The rest of this chapter is organized as follows. In section 5.2, the methodology of the bi-EA and its components are described. Section 5.3 provides the experimental study of bi-EA, analyzes its different control parameters and compares its results with those from the proposed memetic algorithm (MA) and DE (Chapter 3 and 4, respectively) and state-of-the-art algorithms. Finally, a summary of this chapter is presented in section 5.4.

5.2 Methodology

The general framework of the proposed bi-EA is illustrated in Figure 5.1.

Firstly, the initial population (PS) is divided into two sub-populations, each of which is processed by either MA or DE in parallel. In order to increase the exploration in the early stages of the search process and the exploitation capability later, each sub-population size is adaptively reduced according to:

$$new_{PS_l} = \left(\frac{lower_{PS_l} - upper_{PS_l}}{Fit_{Max}} \times cfe\right) + upper_{PS_l} \tag{5.1}$$

where $l = \{MA, DE\}$, cfe is the number of fitness evaluations, Fit_{Max} the maximum number of cfe and $lower_{PS_l}$ and $upper_{PS_l}$ the minimum and maximum population sizes of the two sub-populations, respectively.

At the end of each generation, the success rates ($Succ_{rate}$) of MA and DE are calculated separately according to:

$$Succ_{rate_{l,g}} = \frac{si_{l,g}}{PS_{l,g}}, \qquad \forall l = \text{MA } or \text{ DE} \tag{5.2}$$

where si_l is the total number of individuals successfully improved by each l, l either MA or DE, and PS_l the sub-population size assigned to each l in the current generation (g).

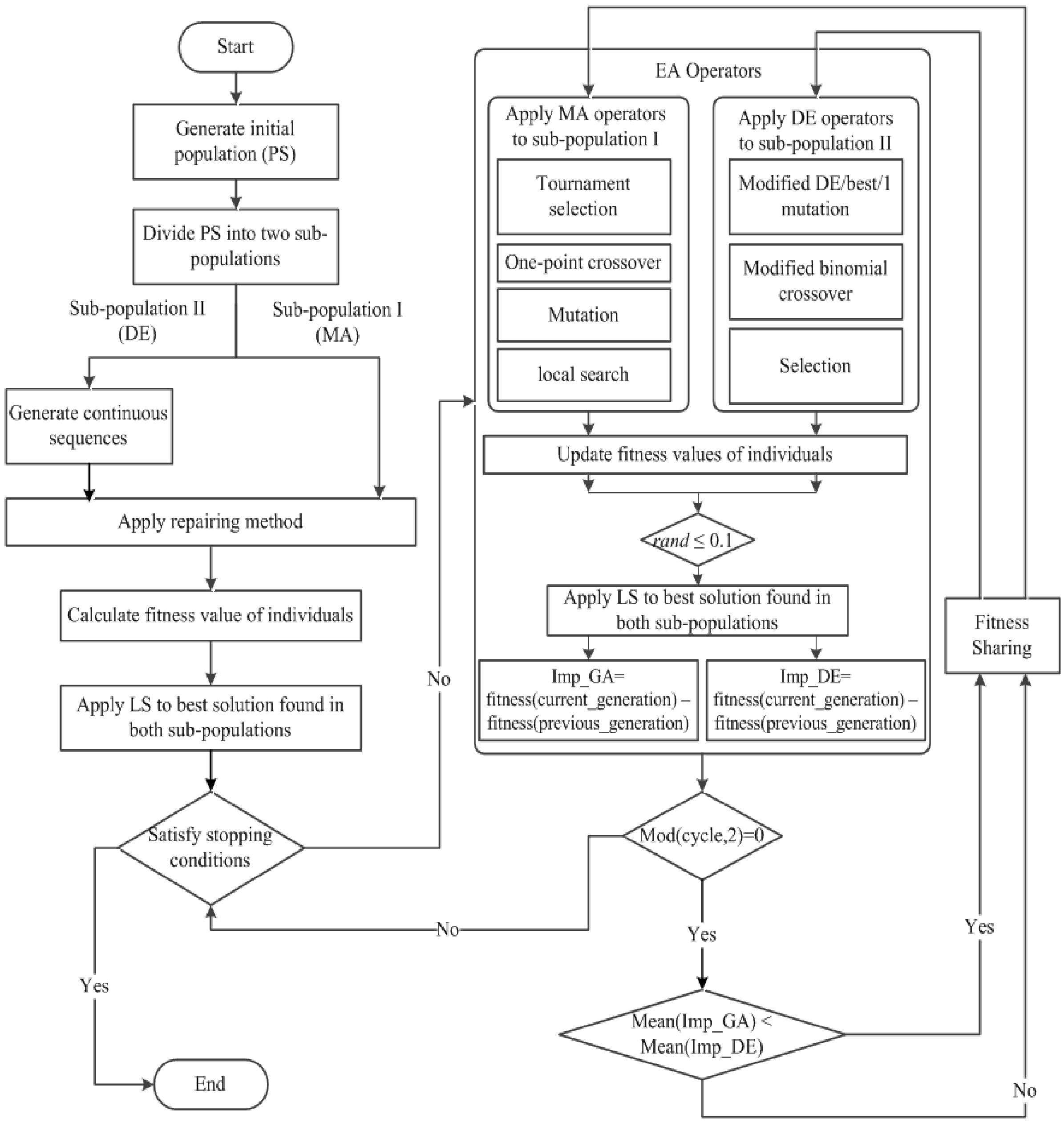


Figure 5.1: General framework of proposed bi-EA

Bi-EA utilizes the good search features of MA and DE by automatically switching between them according to their performances or $Succ_{rate}$ which means that more emphasis is placed on the best performing algorithm during the evolutionary process.

A brief description of each component of bi-EA is provided in the following sub-sections.

5.2.1 Chromosome representation and initial population

As described in Chapter 3, every chromosome from the proposed MA is represented by a vector, the length of which equals the number of activities in the project represented as integer values.

For DE, in the initial population, an additional random continuous vector is generated for each individual as a random sequence (i.e., the order of scheduling/execution each gene in the individual), as stated in Chapter 4.

In bi-EA, the predecessor-successor relationships among activities in a project are represented by an incidence matrix which makes it easy to check the precedence constraints which are represented in the same way as in Chapters 3 and 4.

For the initial population, PS individuals are randomly generated. Then, this population is divided into two sub-populations of size $PS/2$, where one is a discrete space and evolved by MA and the other a real-value space processed by DE.

5.2.2 Fitness calculation and proposed repairing method

For fitness calculations, the activities of the candidate solutions are scheduled according to their appearances in the schedule generated by MA and their corresponding values (order) in the sequences vector in DE. In bi-EA, each activity can be scheduled if, and only if, its required amount of resources does not exceed the pre-defined resource limitation (or resource availability) at a specific time. Consequently, the order of activities within the generated schedule is modified, if needed, to satisfy the resource availability constraint. To compute the total makespan (project duration) of a candidate solution, as the solution in bi-

EA is scheduled activity by activity, each time a new activity is added, the makespan of the project is updated to equal the finish time of the new activity.

As described in previous chapters, RCPSPs are complex optimization problems for which the evolutionary process takes too long to converge if there are no feasible solutions in the initial population. As a solution is considered feasible if no constraints are violated, in Chapter 3, a repairing method for converting an infeasible solution to a feasible one to ensure feasibility in the initial population and speed up the convergence rate of the algorithm is proposed. In this technique, the activities in an individual are re-ordered to satisfy the predecessor-successor relationships among them and, to ensure diversity in the population, this is applied to a certain percentage of the population size (R_m).

5.2.3 MA and DE

Bi-EA incorporates the same MA and DE processes as in Chapters 3 and 4, respectively. From these chapters, it is found that GA was a standout amongst the most well-known heuristic algorithms to deal with optimization problems and they addressed an intense and powerful methodology for large scale RCPSPs. One more point of interest of utilizing GA is that awful individuals in the initial population don't altogether influence the final solution negatively, as in every generation the fitter individuals only will be survived for the next generation. Moreover, DE had mainly demonstrated great convergence properties and is principally straightforward and more reliable.

5.2.4 Local search

Although local search may bring less attractive solutions or get stuck at local optima, it leads to better solutions at times. In bi-EA, a local search is applied to the best solution found in both sup-populations for finding out the possibility of reducing the fitness value by rescheduling some of activities so as not to violate the feasibility constraints.

In it, the last predecessor and the first successor activities of each activity (j) in the individual are determined. If the finish time +1 of j's last predecessor equal to the start time of j and the j's finish time +1 equal to start time of its first successor, then no gap existed among them. Otherwise, the resource availability is checked at this specific time zone (the gab) and the activity is relocated by changing its start and finish time without violating resource availability or precedence constraints. Rescheduling activities by reducing their start and finish times to fit with their predecessor and successor activities in an individual leads to reduce the total duration of the project and hence obtaining solution better than the original one. The pseudo-code of the proposed local search is given in Figure 5.2.

For j = 1 to n **do**
Find l_Pre_j; the last predecessor of current gene (j)
Find f_succ_j; the first successor of current gene (j)
    If Finish_time(l_Pre_j)+1≠Start_time(j) **or** Finish_time(j)≠Start_time(f_succ_j)+1

//check resource availability after adding resources required by j to those occupied by the ongoing activities at t ($A(t)$).

        If resource($A(t)$) + resource(j)≤ max_resource_availability,
            Start_time (j) ←Finish_time (l_Pre_j)+1
            Finish_time (j) ←Start_time (f_succ_j)+1
        End if
    End if
End for

Figure 5.2: Pseudo-code of the local search

In order to show the influence of applying the proposed local search on an individual, an example is provided with problem J30.13_5 from PSPLIB with makespan of 87.

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

After relocating its activities using the proposed local search, the following feasible individual was produced with makespan 80.

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

The start and finish time of each activity before and after applying the proposed local search to the individual are shown in Figures 5.3 and 5.4, respectively.

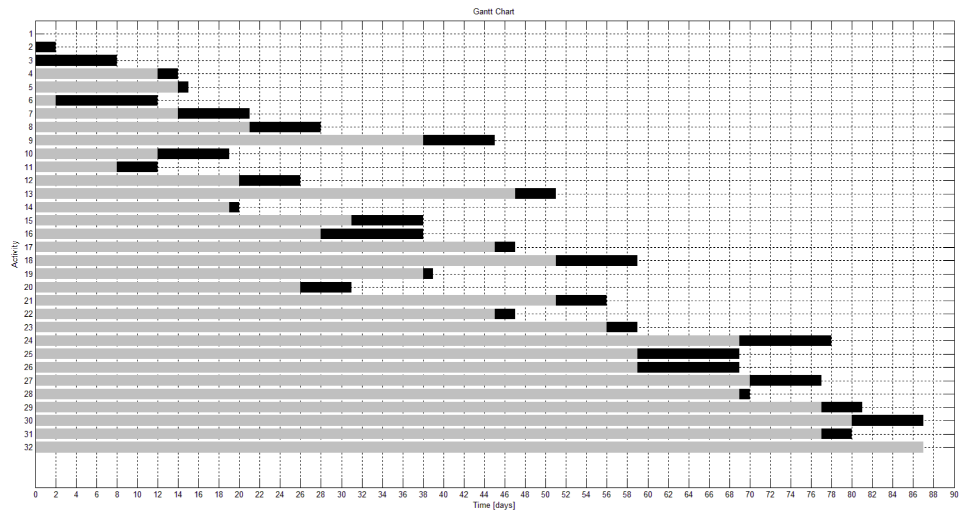


Figure 5.3: Start and finish times of each activity in individual before applying proposed local search

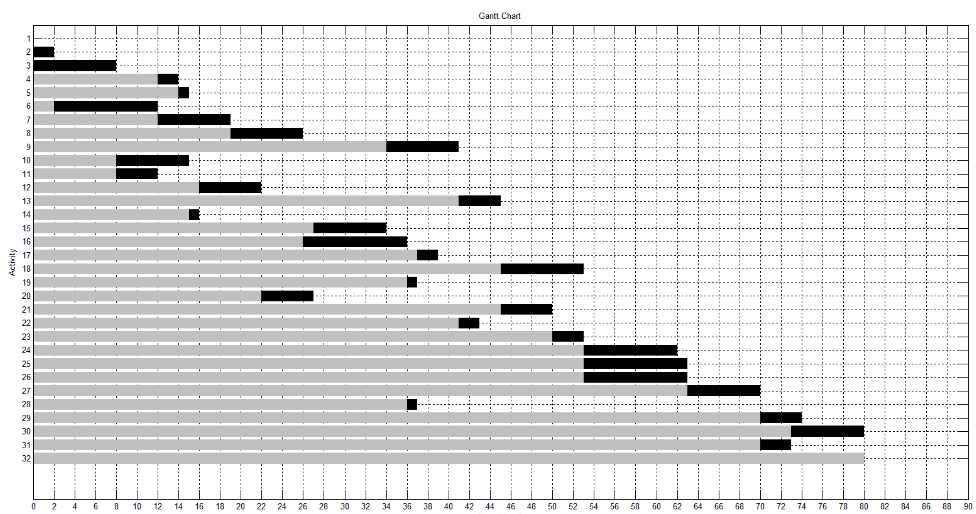


Figure 5.4: Start and finish times of each activity in individual after applying proposed local search

5.2.5 Switching between MA and DE

Initially, the two sub-populations are evolved in parallel for a *cycle* which is defined as a given number of generations, at the end of which the performances of both algorithms are determined by calculating their average success rates ($Succ_{rate}$) over this *cycle* (CS). For the next *cycle*, one sub-population is chosen to be evolved with either MA or DE according to the best average $Succ_{rate}$. However, the best few individuals from the non-selected sub-population are shared in the evolutionary process of the selected one. Then, in the following *cycle*, both algorithms are re-run in parallel, each with its own sub-population, after sharing the current best individuals each one obtained during its search.

This information-sharing process is very important as both algorithms are continually updated with the latest improvements in the population so that either one or both start to explore new regions of the search space instead of rediscovering the same ones. The process of running a 'single sub-population' and 'both sub-populations' or a 'single-EA' and 'bi-EA' continues in alternate cycles until one of the stopping criteria is met, a process shown in Figure 5.5.

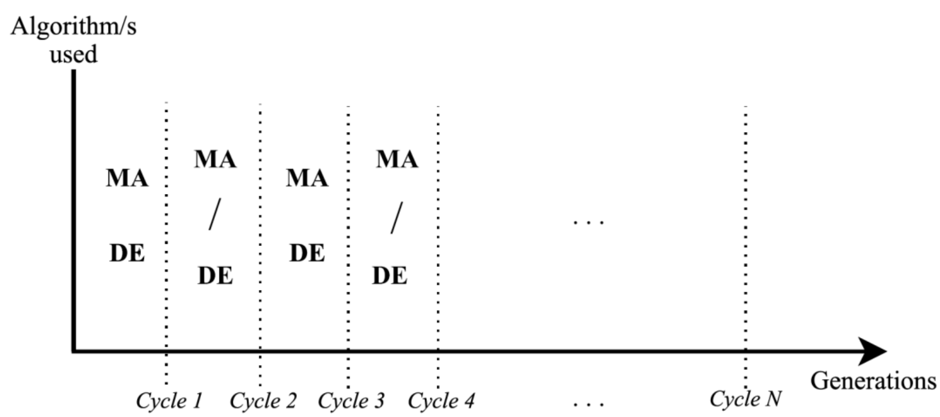


Figure 5.5: Trade-off between MA and DE for every *cycle*

5.3 Experimental Results

Bi-EA was coded using Matlab R2013b and implemented on a PC with a 3.4 GHz CPU and Windows 7, with a standard benchmark set of problems from the PSPLIB used to analyze its performance.

In bi-EA, the same data set as in the previous chapters was used with an extended number of test problems, that is, 2040 with different dimensions, 480 of each of the J30, J60 and J90 instances and 600 of the J120 ones from the PSPLIB.

In this section, a brief parametric analysis of bi-EA is discussed and then the computational results from its final version presented. Finally, comparisons of the proposed MA, DE and bi-EA are conducted and the bi-EA compared with some state-of-the-art algorithms.

To do this, results such as the average percentage deviation from the optimal for the J30 instances and deviations from the lower bound generated by the critical path method (CPM), as reported in Stinson et al. (1978), for the J60, J90 and J120 were used. The average deviation can be calculated by:

$$AvgDev(\%) = \left(\frac{1}{S}\sum_{s=1}^{S}\frac{BS_s - LB_s}{LB_s}\right) \times 100 \tag{5.3}$$

Where S is the total number of instances used, BS_S the best solution achieved by the algorithm for S instances and LB_S the pre-known lower bound of a s instance.

5.3.1 Parameter settings

For each test problem, 30 independent runs were executed, with the two stopping criteria: (i) run for up to Fit_{Max} fitness evaluations; or (ii) no improvement in the fitness value during 150 consecutive generations, where Fit_{Max} was set to values of 1000, 5000 and 50,000.

For MA, based on the parametric analyses conducted in Chapter 3, the tournament pool size (TS) was randomly selected as 2 or 3, the crossover operator (p_c) set to 1 and the mutation rate (p_m) adaptively calculated as:

$$p_m = Max\left(\partial_{min}, \partial_{max} - (\partial_{max} - \partial_{min}) \times \left(\frac{cfe}{Fit_{Max}}\right)\right) \tag{5.4}$$

where ∂ is the lower limit of the mutation rate, ∂_{max} the initial value of the mutation rate and cfe the number of fitness evaluations, with ∂_{min}=0.05 and ∂_{max}=0.2.

For DE**,** based on the parametric analyses in Chapter 4, the mutation factor (F) was set to 0.1 with mutation probability (p_m) equal 1. Crossover rate (CR) was calculated adaptively using equation (4.4) in Chapter 4, where CR_{LB}= 0.1 and CR_{UB}= 0.9 for creating a balance between a good initial value and the convergence speed.

In general, R_m, which is the number of individuals repaired to be feasible using the proposed repairing method explained in Chapters 3 and 4, was set to 25% of each sub-population size which calculated adaptively from 100 to 30. Finally, CS set to 50 generations. Table 5.1 summaries the parameter settings of bi-EA.

General	**MA**	**DE**
PS is adaptively reduced from 100 to 30 for each sub-population.	Tournament selection size (TS) = random between 2 and 3	CR is calculated adaptively from 0.1 to 0.8
R_m = 25%	CR = 1	Mutation rate = 1, F=0.9
CS (Cycle size) = 50	p_m adaptive from 0.2 to 0.05	The greedy selection strategy to choose the best solution is adopted.
$MaxFit$= 1,000, 5,000 and 50,000		

Table 5.1: Summary of parameter settings of bi-EA

Also, Section 5.3.3 describes justifications of a selection of the general parameters by studying those using different values.

5.3.2 Computational results

In this sub-section, the results from the final variant of the proposed bi-EA are presented.

Tables 5.2 and 5.3 show a summary of the results for the J30, J60, J90 and J120 instances and the success rates (%), respectively of bi-EA up to three different numbers of schedules of 1000, 5000 and 50,000.

In table 5.2, the average deviation ($AD\%$) from the optimal solutions for J30 and the critical path lower bound ones for J60, J90 and J120 from PSPLIB, the standard deviation of $AD\%$ and the computational time (t) of all instances are shown.

No. of generations	J30			J60			J90			J120		
	$AD\%$	STD	t	$AD\%$	STD	t	$AD\%$	STD	t	$AD\%$	STD	t
1,000	0.37	0.75	3.37	13.36	1.11	9.46	14.09	1.01	15.56	42.87	2.78	31.27
5,000	0.22	0.63	14.35	12.51	1.02	40.74	13.15	0.96	48.63	40.46	2.25	165.91
50,000	0.1	0.47	32.37	12.1	0.80	97.11	13.09	0.86	145.78	38.89	2.17	684.50

Table 5.2: Summary results of all J30, J60, J90 and J120 instances with 1,000, 5,000 and 50,000 generations

In Table 5.3, the percentage of the success rate of bi-EA which can be calculated according to (5.5) is shown.

$$Success = \frac{Number\ of\ problems\ optimally\ solved}{Number\ of\ problems\ in\ each\ instance} \times 100 \tag{5.5}$$

No. of generations	J30	J60	J90	J120
1,000	83.33	59.38	63.96	16.33
5,000	90	59.79	64.85	17.67
50,000	94	60.21	64.79	19.5

Table 5.3: Success rates (%) of bi-EA for J30, J60, J90 and J120 with 1,000, 5,000 and 50,000 generations

From these tables, it can be noticed that the performance of the proposed bi-EA increased with increasing the maximum number of generations as, for example, the improvement in solution-quality ($AD\%$) for J30, which can be calculated using equation 4.6, increased with increasing number of generations from 1,000 to 50,000 by 11.35% and from 5,000 to 50,000 by 4.26%, and for J120, by 16.26% and 9.39%, respectively.

In order to show the influence of the complexity factors (discussed in Chapter 3) on the performance of bi-EA, Tables 5.4 provides the $AvgDev(\%)$ from the critical path lower bound solution ($AD\%$) and the average computational time (t) for 48 instances of each J30, J60 and J90, and 60 instances of J120, where each instance contains 10 different problems, running with 5,000 generations for 30 independent runs grouped into 12 groups based on the values of the three complexity factors: resource factor (RF), network complexity (NC) and resource strength (RS). Based on this classification, each group contains four instances (40 problems) and has same values of NC and RF and RS ranges from 0.2 to 1 for J30, J60 and J90. For J120, each group contains five instances (50 problems) with same NS and RF values and RS ranges from 0.1 to 0.5.

Group	Complexity factors			J30		J60		J90			J120	
	NC	RF	RS	$AD\%$	t	$AD\%$	t	$AD\%$	t	RS	$AD\%$	t
1	1.50	0.25	0.20-1	0.00	5.52	3.80	21.85	3.89	37.62	0.1-0.5	11.94	181.58
2	1.50	0.50	0.20-1	0.15	17.57	9.58	44.84	10.75	58.11	0.1-0.5	35.30	252.62
3	1.50	0.75	0.20-1	0.36	16.32	13.10	41.68	16.91	54.42	0.1-0.5	49.45	197.47
4	1.50	1.00	0.20-1	0.59	14.50	20.14	47.09	19.33	55.72	0.1-0.5	60.77	155.59
5	1.80	0.25	0.20-1	0.00	2.89	3.22	24.26	3.12	34.28	0.1-0.5	10.17	105.71
6	1.80	0.50	0.20-1	0.13	13.52	10.68	41.92	11.01	38.22	0.1-0.5	33.46	150.25
7	1.80	0.75	0.20-1	0.45	17.03	15.23	42.54	17.34	51.14	0.1-0.5	49.96	163.30
8	1.80	1.00	0.20-1	0.55	24.88	21.16	51.67	19.32	53.70	0.1-0.5	59.69	168.66
9	2.10	0.25	0.20-1	0.00	3.60	3.75	22.62	2.67	31.58	0.1-0.5	11.26	122.71
10	2.10	0.50	0.20-1	0.03	12.06	10.55	43.36	11.70	53.03	0.1-0.5	36.95	155.59
11	2.10	0.75	0.20-1	0.24	21.88	16.79	51.05	18.56	56.37	0.1-0.5	57.49	166.33
12	2.10	1.00	0.20-1	0.18	22.43	22.16	73.16	23.13	63.58	0.1-0.5	69.10	171.14
Average				**0.22**	**14.35**	**12.51**	**42.17**	**13.14**	**48.98**		**40.46**	**165.91**

Table 5.4: Average deviations and computational times of all instances of J30, J60, J90 and J120 grouped by values of complexity factors

According to Kolisch et al. (1995), the complexity of a RCPSP increases with an increasing RF value and decreasing both NC and RS values. From this table, it can be noticed that the quality of the solutions obtained by bi-EA for J30 is decreased by increasing RF value as from group 1 to 4, and the t is increased as from group 5 to 8. However, for the hardest group, i.e. group 4 (as it has the lowest NC and the highest RF values), bi-EA obtained best solution in less computational time as its design prevents it to be trapped in a local solution while dealing with complex problems. For some other groups, such as 2, 5 and 9 of J90 and J120, bi-EA achieved less $AD\%$ values because of the exploitation capability added to bi-EA by applying the proposed LS, as it performs a depth search to the best solutions to obtain better ones. Contrariwise, for some cases, bi-EA took some time to obtain the best solution because of the lack of diversity that may be happened due to producing individuals with fitness values similar to their parents, which is usually seen in problems with small number of activities.

5.3.3 Parametric analysis

Two parameters, PS and CS, were analyzed to investigate their effects on the performance of bi-EA, with the $AvgDev(\%)$ of the solutions calculated using different PS values of 30 (PS_{30}), 100 (PS_{100}), 200 (PS_{200}), 300 (PS_{300}) and adaptive from 100 to 30 for each sub-population ($sub_{adaptPS_{100\sim30}}$) while using different values of CS of 10, 30, 50, 70 and 100, with all other parameters fixed as described in Section 5.3.1.

In the following sub-sections, the best parameter found in an experiment is fixed in the subsequent ones, as the selection of the parameters is done in a sequential manner.

5.3.3.1 Effect of PS

As the number of populations is a very significant parameter in meta-heuristic algorithms, it is very important to define an appropriate population size to help the algorithm achieve the optimal solution. While a low one may cause an incomplete convergence and prevent global optimum solutions being obtained, on the other hand, a large one may require a great deal of computational time before the best solution is obtained as in Figure 5.6 where PS with 300 began to achieve better $AvgDev(\%)$ than both 100 and 200 for J120; however, it took 5.6% time more than the adaptive one. Therefore, to achieve a good balance between the computational time and solution quality, each sub-population's size was adaptively reduced according to equation (5.1). According to Figure 5.6, the algorithm showed almost similar performance for small instances such as J30 and J60; however, for larger ones, J90 and J120, the figure shows that $sub_{adaptPS_{100\sim30}}$ obtained better results than other PS values in terms of the quality of solutions by achieving the lowest $AvgDev(\%)$.

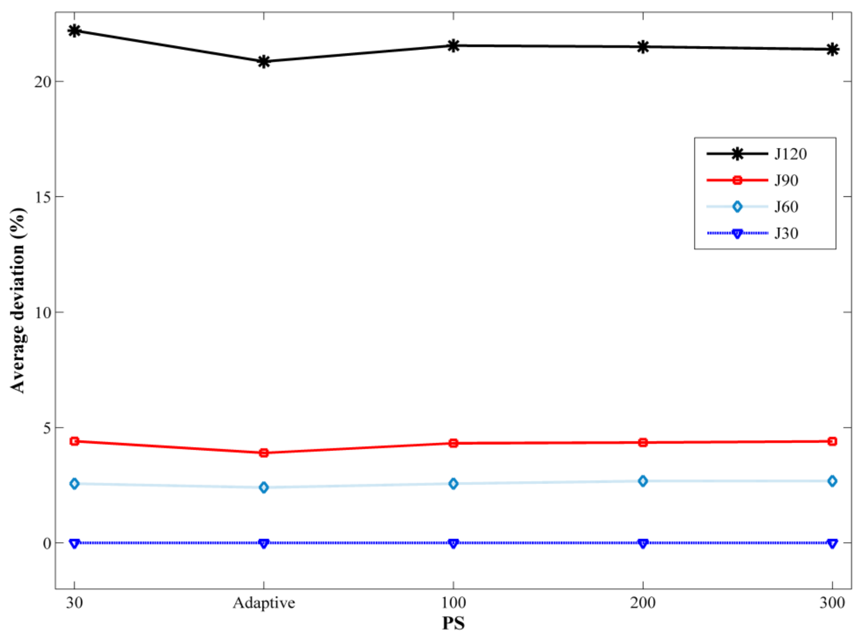


Figure 5.6: Trends of $AvgDev(\%)$ using different PS values

5.3.3.2 Effect of CS

The CS is also a very important parameter in bi-EA as it controls the duration of either MA or DE by determining the number of iterations the selected EA should accomplish before the next performance measurement which computes both algorithms' average success rates. The algorithm with the highest success rate was selected to evolve the population up to the next CS. In this analysis, performance measurements of both MA and DE were taken at the end of every CS, where CS = 10, 30, 50, 70 and 100 iterations.

Figure 5.7 shows that a CS of 50 outperformed other values in terms of $AvgDev(\%)$ as for J120, CS with 50 improved the quality of the obtained solutions by 7.3% than 10 and by 2.97% than 100. This may have been because it provided a good balance by allowing sufficient time for the algorithm to obtain good solutions, unlike CS=10 and 30, and prevented the algorithm from becoming trapped in a local optimum solution for a long time, as may have happened for CS=70 and 100.

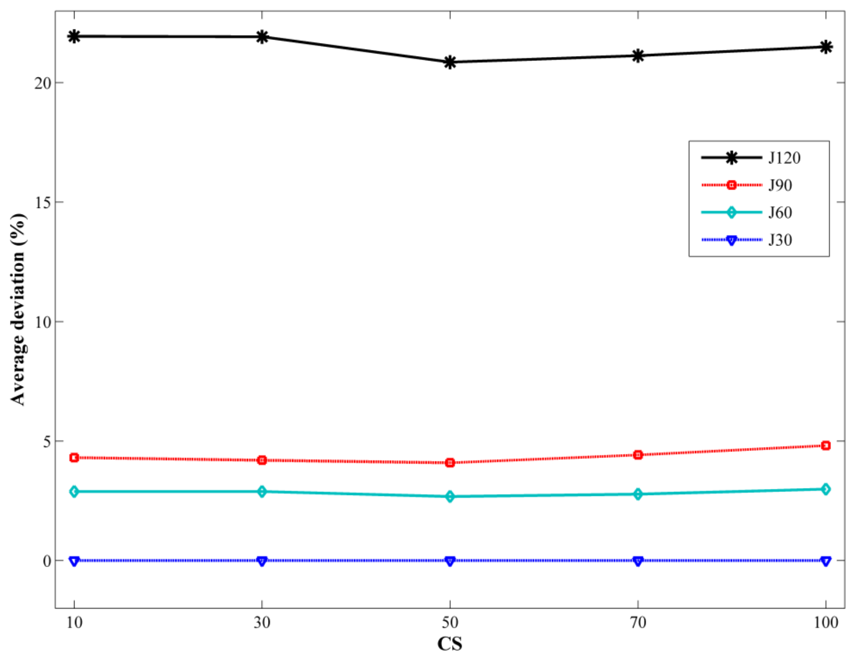


Figure 5.7: Trends of $AvgDev(\%)$ using different CS values

5.3.4 Comparisons with other algorithms

To judge the performance of the proposed bi-EA, comparisons of its results with those obtained from previously proposed MA and DE (Chapters 3 and 4, respectively) and some well-known state-of-the-art-algorithms are discussed in the following sub-sections.

5.3.4.1 Comparison of Bi-EA and proposed MA and DE

In order to demonstrate the benefit of bi-EA, the same data set used in previous chapters, which contained 16 problems from different instances of J30 and 15 from each J60, J90 and J120, was used, with each algorithm run for up to n ×10,000 generations for every 30 independent runs, where n is the number of activities in each instance. The average deviations from critical path lower bound ($AvgDev$(%)), average CPU times per run and numbers of optimal solutions achieved (No. of opt.) of MA, DE and bi-EA are shown in Table 5.5.

From this table, it is clear that MA was the fastest but not had the best deviation values or solution quality. Although, DE was computationally expensive, it could achieve better average deviation value than MA for J60, J90 and J120 because of its capability to explore a wide range of the search space. Bi-EA was able to produce high-quality solutions with minimum deviation values than MA in large-sized instances and achieve optimal solutions for more test problems than both MA and DE in lower computational time than DE.

Alg.		**J30**	**J60**	**J90**	**J120**
MA	AvgDev(%)	0.00	3.02	5.78	32.94
	Avg. CPU time	**1.48**	**14.77**	**12.68**	**28.82**
	No. of opt.	16/16	8/15	10/15	0/15
DE	AvgDev(%)	0.00	2.99	5.72	32.61
	Avg. CPU time	37.81	269.45	890.47	2636.90
	No. of opt.	16/16	8/15	7/15	0/15
Bi-EA	AvgDev(%)	**0.00**	**2.92**	**5.59**	**30.69**
	Avg. CPU time	22.51	173.545	364.46	1013.067
	No. of opt.	**16/16**	**8/15**	**10/15**	**1/15**

Table 5.5: Comparisons of MA, DE and bi-EA

To indicate the numbers of Fit_{Max} each algorithm used before obtaining the best solution, Figure 5.8 shows those of MA, DE and bi-EA used for solving 16 randomly selected problems of the J30 instance. It can be seen that bi-EA could reach the optimal solution with a much lower Fit_{Max} value than both MA and DE and, therefore, obtained good-quality solutions in less computational time. Also, DE used a high value of Fit_{Max} for solving J30.1-2, while MA and bi-EA solved same problem in much lower value and this indicates that DE can effectively explores the search space. However, its exploitation capability of a certain solution is very slow.

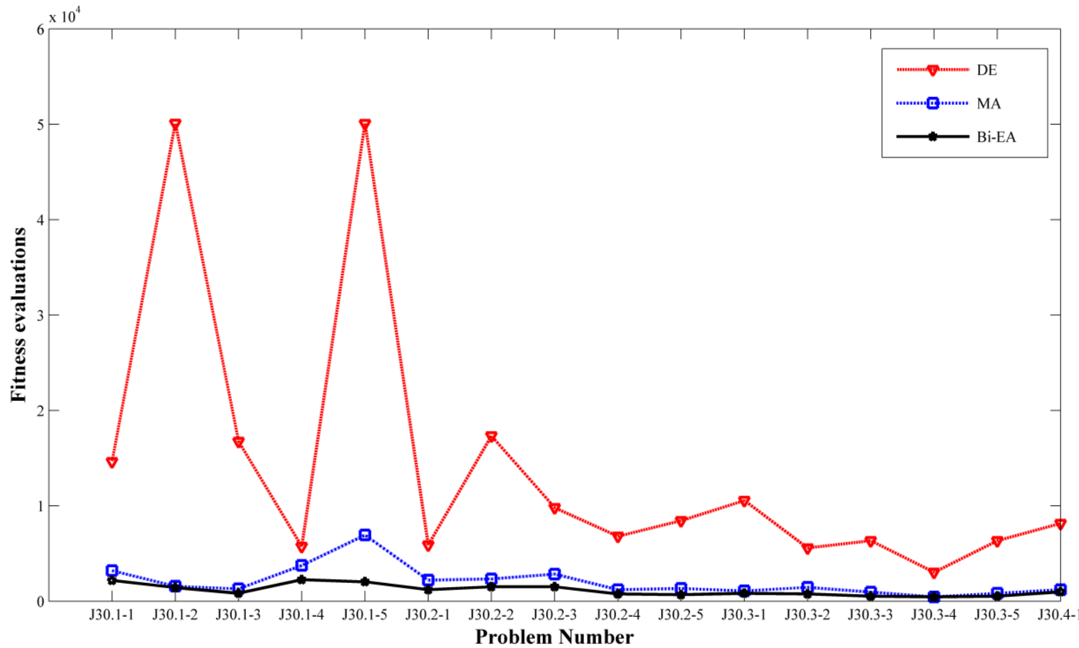


Figure 5.8: Fit_{Max} for each problem using GA/MA and bi-EA

For studying the differences among the performance of MA, DE and Bi-EA in more meaningful way, a statistical significant testing using Wilcoxon Signed Rank Test (Corder and Foreman, 2009) with 10% significance level (shown in Chapter 4) and based on the average deviation from the critical path values (LB_{CP}) and from the known optimal solutions (LB_{OP}) is performed.

From Table 5.6, it can be seen that there is no significant difference between the performances of MA, DE and Bi-EA for small instances such as J30 and J60. However, for large-sized instances, the table shows that Bi-EA is significantly better than both MA for J120 and DE for J90 in regarding to the LB_{OP}. Also, it is clear that there is a significant difference between Bi-EA and DE for J120 in regarding to LB_{CP}.

Algorithms	**Instance**	**Criteria**	***P*- Value**	**Decision**
Bi-EA - to - MA	J30	LB_{CP}	1.000	≈
		LB_{OP}	1.000	≈
	J60	LB_{CP}	0.317	≈
		LB_{OP}	0.317	≈
	J90	LB_{CP}	0.593	≈
		LB_{OP}	0.593	≈
	J120	LB_{CP}	0.100	≈
		LB_{OP}	0.088	+
Bi-EA - to - DE	J30	LB_{CP}	1.000	≈
		LB_{OP}	1.000	≈
	J60	LB_{CP}	1.000	≈
		LB_{OP}	1.000	≈
	J90	LB_{CP}	0.753	≈
		LB_{OP}	0.069	+
	J120	LB_{CP}	0.087	+
		LB_{OP}	0.133	≈

Table 5.6: Wilcoxon Signed Rank Test for MA, DE and Bi-EA

In order to clarify the performance of bi-EA in terms of the complexity factors, the average deviation (AD%) from the critical path lower bound, the standard deviation (STD) of AD% and the computational time (t) of the four algorithms: branch and bound (B&B), GA, Lagrange relaxation based GA (GA_LR) and branch and cut (B&C) introduced by Chakrabortty et al. (2015), and described in Chapter 3, the proposed MA (discussed in Chapter 3) and the improved DE (discussed in Chapter 4) on the same data set (15 problems from three different instances from J30) are shown in Table 5.6.

In this table, groups 1, 2 and 3 have same values of both NC and RF with values 1.50 and 0.25, respectively and three different values of RS=0.20, 0.50 and 0.70, respectively.

From this table, it can be observed that the three proposed algorithms (MA, DE and bi-EA) had the best quality of solutions and MA was faster than others. However, DE had the lowest standard deviation value (STD) which demonstrates its reliability for achieving the optimal solutions. Consequently, bi-EA had lower computational times than DE and higher reliability than MA.

Group	**1**			**2**			**3**			**Average**		
$\boldsymbol{RS}$	**0.20**			**0.50**			**0.70**					
Algorithms	$AD\%$	t	STD	$AD\%$	t	STD	$AD\%$	t	STD	$AD\%$	t	STD
Bi-EA	**0**	40.71	0.42	**0**	20.94	0.19	**0**	8.29	**0**	**0**	23.31	0.20
Improved DE	**0**	49.90	**0.34**	**0**	55.02	**0.12**	**0**	15.36	**0**	**0**	39.45	**0.16**
MA	**0**	**3.09**	1.61	**0**	1.48	0.20	**0**	**0.13**	**0**	**0**	**1.57**	0.60
B&B	**0**	14.8	**-**	**0**	1.0	**-**	**0**	20.8	**-**	**0**	12.20	-
GA	1.53	830	**-**	**0**	913	**-**	0.67	1000	**-**	0.76	914.33	-
GA_LR	1.11	145.9	**-**	**0**	74.8	**-**	0.20	460	**-**	0.44	227.00	-
B&C	**0**	68.43	**-**	**0**	5.27	**-**	**0**	2.77	**-**	**0**	25.49	-

Table 5.7: Average deviations and CPU times of proposed bi-EA and other algorithms for J30 with different values of $\boldsymbol{RS}$

The above comparisons demonstrate the effectiveness of the use of multiple algorithm strategy for solving RCPSPs as the proposed bi-EA uses the good search features of both MA and DE. Therefore, it was able to achieve good quality solutions for some problems than GA (feature from DE) in lower computational times than DE (feature from GA). As a consequence, bi-EA achieved very low average deviations in lower Fit_{Max} than both MA and DE with different complexity levels.

5.3.4.2 Comparisons of Bi-EA and state-of-the-art algorithms

To compare bi-EA and other algorithms from the literature, the performance of bi-EA was tested by setting the maximum number of generated schedules to 1000, 5000 and 50,000 and, for each, running the algorithm independently using 2040 test problems for 30 runs.

In this sub-section, comparisons of bi-EA and 18 algorithms selected from the literature based on their published average deviations from the optimal solution for J30 and the critical path lower bound for J60 and J120 are discussed. The $AvgDev(\%)$ is calculated using equation (5.3). In Tables 5.7 to 5.9, the $AvgDev(\%)$ values of all the comparative algorithms for the J30, J60 and J120 instances, respectively, with different maximum number of generations are listed.

Algorithm	Reference	Maximum number of schedules		
		1,000	5,000	50,000
Bi-EA	This study	0.37	0.22	0.1
Multi-agent optimization algorithm	(Zheng and Wang, 2015)	0.17	0.06	0.01
PSO	(Fahmy et al., 2014)	0.22	0.05	0.02
Magnet-based GA	(Zamani, 2013)	0.14	0.04	0.00
Shuffled frog-leaping	(Fang and Wang, 2012)	-	0.21	0.18
Neurogenetic approach	(Agarwal et al., 2011)	0.13	0.10	-
ABC	(Ziarati et al., 2011)	0.98	0.57	0.20
BSO	(Ziarati et al., 2011)	0.65	0.36	0.17
BA	(Ziarati et al., 2011)	0.63	0.33	0.16
ABC-FBI	(Ziarati et al., 2011)	0.47	0.28	0.09
BSO-FBI	(Ziarati et al., 2011)	0.45	0.22	0.07
BA-FBI	(Ziarati et al., 2011)	0.42	-	-
ACOSS	(Chen et al., 2010)	0.14	0.06	0.01
Random key-based GA	(Mendes et al., 2009)	0.06	0.02	0.01
GA- random key	(Hartmann, 1998)	1.03	0.56	0.23
Sampling-LFT	(Kolisch, 1996b)	1.40	1.29	1.13
Sampling-Adaptive	(Kolisch and Drexl, 1996)	0.74	0.52	-
Sampling-random using parallel SGS	(Kolisch, 1995)	1.77	1.48	1.22
GA- problem space	(Leon and Balakrishnan, 1995)	2.08	1.59	-

Table 5.8: $AvgDev$(%) for J30 instances

Algorithm	Reference	Maximum number of schedules		
		1,000	5,000	50,000
Bi-EA	This study	13.36	12.51	12.1
Multi-agent optimization algorithm	(Zheng and Wang, 2015)	11.67	10.84	10.64
PSO	(Fahmy et al., 2014)	11.86	11.19	10.85
Magnet-based GA	(Zamani, 2013)	11.33	10.94	10.65
Shuffled frog-leaping	(Fang and Wang, 2012)	-	10.87	10.66
Neurogenetic approach	(Agarwal et al., 2011)	11.51	11.29	–
ABC	(Ziarati et al., 2011)	14.57	13.12	12.53
BSO	(Ziarati et al., 2011)	13.67	12.70	12.45
BA	(Ziarati et al., 2011)	13.35	12.83	12.41
ABC-FBI	(Ziarati et al., 2011)	12.61	12.24	11.23
BSO-FBI	(Ziarati et al., 2011)	12.58	12.29	11.21
BA-FBI	(Ziarati et al., 2011)	12.55	12.04	11.16
ACOSS	(Chen et al., 2010)	11.35	10.98	10.67
Random key-based GA	(Mendes et al., 2009)	11.72	11.04	10.67
GA- random key	(Hartmann, 1998)	14.68	13.32	12.25
Sampling-LFT	(Kolisch, 1996b)	13.59	13.23	12.85
Sampling-Adaptive	(Kolisch and Drexl, 1996)	-	11.17	10.74
Sampling-random using parallel SGS	(Kolisch, 1995)	15.94	15.17	14.22
GA- problem space	(Leon and Balakrishnan, 1995)	14.33	13.49	-

Table 5.9: $AvgDev$(%) for J60 instances

Algorithm	Reference	Maximum number of schedules		
		1,000	5,000	50,000
Bi-EA	This study	42.87	40.46	38.89
Multi-agent optimization algorithm	(Zheng and Wang, 2015)	33.87	32.64	31.02
PSO	(Fahmy et al., 2014)	35.60	33.78	32.4
Magnet-based GA	(Zamani, 2013)	34.02	32.89	31.30
Shuffled frog-leaping	(Fang and Wang, 2012)	-	33.2	31.11
Neurogenetic approach	(Agarwal et al., 2011)	34.65	34.15	-
ABC	(Ziarati et al., 2011)	43.24	39.87	37.36
BSO	(Ziarati et al., 2011)	41.18	37.86	35.70
BA	(Ziarati et al., 2011)	40.38	38.12	36.12
ABC-FBI	(Ziarati et al., 2011)	37.85	36.82	35.02
BSO-FBI	(Ziarati et al., 2011)	37.84	36.51	34.86
BA-FBI	(Ziarati et al., 2011)	37.72	36.76	34.55
ACOSS	(Chen et al., 2010)	35.19	32.48	30.56
Random key based GA	(Mendes et al., 2009)	35.87	33.03	31.44
GA- Random key	(Hartmann, 1998)	45.82	42.25	38.83
Sampling-LFT	(Kolisch, 1996b)	42.84	41.84	40.63
Sampling-random using parallel SGS	(Kolisch, 1995)	44.46	43.05	41.44
Sampling-random using serial SGS	(Kolisch, 1995)	49.25	47.61	45.60
GA- problem space	(Leon and Balakrishnan, 1995)	42.91	40.69	-

Table 5.10: $AvgDev$(%) for J120 instances

Table 5.7 summarizes the average percentage deviation from the optimal solution for J30 instances. From it, bi-EA obtains the optimal solutions for 452 out of 480 problems, i.e. for 94% of the instances using a very low number of fitness evaluations (=1,189 cfe). Comparing with most of comparative algorithms, the proposed bi-EA shows more consistency for solving J30 instances with competitive average deviation values.

Tables 5.8 and 5.9 summarize the average percentage deviation from the critical path-based lower bound for J60 and J120 instances, respectively reported by Stinson et al. (1978). Bi-EA showed a competitive performance against some state-of-the-art-algorithms with less consumption of fitness evaluations. For J60, bi-EA used number of fitness evaluations = 3,790 cfe and for J120, = 9,925 cfe which demonstrate that bi-EA used a very low fitness evaluations which in turn saves more CPU times.

Although bi-EA shows low average deviation values for solving RCPSPs with different numbers of activity such as J30, J60, J90 and J120, it doesn't achieve the best performance among all and the reasons for this drawback might be that GA and DE used are not powerful enough and/or not complementary to each another. Also, they might not maintain sufficient diversity that could help them escape from local solutions.

5.4 Chapter Summary

As a matter of fact, no single evolutionary algorithm (EA) is consistently able to solve all types of problems. In this chapter, a multiple algorithms strategy which combines the good search features of two well-known EAs: MA based on GA and DE is proposed. Performances of each algorithm were improved through the previous chapters of this thesis where MA and DE were discussed in Chapters 3 and 4, respectively.

In bi-EA, to overcome the shortcomings of each MA and DE, the search capabilities of GA, which solves complex problems with multiple solutions and DE, which has demonstrated great convergence properties and its execution is relatively straightforward are integrated with the proposed heuristic repairing procedure (discussed in Chapter 3), which provides feasible solutions in the initial population.

The algorithm is tested on a set of 2040 well-known project scheduling problems taken from PSPLIB, with instances of 30, 60, 90 and 120 activities and its results compared with those obtained by MA (Chapter 3), DE (Chapter 4) and 18 other state-of-the-art algorithms. The results show the capability of the proposed bi-EA to attain good quality solutions, and therefore validate its effectiveness for solving RCPSPs.

Chapter 6

Conclusions and Future Research Directions

This chapter concludes research of this thesis by summarizing the significant technical contributions in the domain of resource-constrained project scheduling problems (RCPSPs) provided by this study and the major conclusions that can be drawn from the experiments conducted. Also, some conceivable directions for further research are suggested.

6.1 Summary of Research Conducted

In Chapter 2, several recent methods and ideas for RCPSPs in the literature were discussed. Of them, although hybrid methods have performed best in solving such problems, their actual capabilities have not yet been fully explored. As focusing on their individual components was necessary to provide a better understanding of their performances, before discussing the design aspects of the proposed bi-evolutionary algorithm (bi-EA), its components were investigated and improved through Chapters 3 and 4.

All the algorithms proposed in this thesis were tested and analyzed using two sets of benchmark problems taken from the well-known project scheduling library (PSPLIB) initiated by Kolisch et al. (1999), as demonstrated in Chapter 3. The first set, which contains small numbers of problems from different instances and complexity levels, was used as to initially test the algorithms' performances and the second set, which contains all the benchmark problems for all instances in the PSPLIB, to prove the effectiveness of the final variant of bi-EA.

In Chapter 3, a memetic algorithm (MA) based on the genetic algorithm (GA) was introduced and investigated as one component of the bi-EA. Its general framework, which used a proposed heuristic repairing method for providing feasible solutions in the initial population and multiple local search (MLS) strategies for increasing the exploitation capability of the algorithm, was discussed. Also, in order to determine the best selection of the MA's parameters, it was tested on the first set of test problems with different experiments run to analyze the effects of these parameters. The experimental results obtained from the final MA variant were presented and compared with those from three variants of the traditional GA executed on the same computer configuration and four other state-of-the-art algorithms for the same test problems. This comparative study validated the effectiveness of the MA.

The second component of the bi-EA was the differential evolution (DE) algorithm, the performance of which was improved in Chapter 4 by integrating the proposed heuristic repairing method (discussed in Chapter 3) and amending the DE operators to be able to deal with the discrete nature of RCPSPs and produce new feasible solutions, even from infeasible ones. The improved DE was also tested on the first set of test problems in order to choose the best parameters. Different experiments were conducted and the performance of the improved DE compared with those of the MA and other state-of-the-art algorithms. The results demonstrated that DE obtained optimal solutions for more problems than the MA and performed reliably but was more expensive in terms of computational time.

Motivated by the above results, the bi-EA, which combined the search capabilities of the proposed MA and DE in one algorithmic framework, was introduced in Chapter 5. The general framework design included: (1) sharing information between the two algorithms; (2) automatically switching between the algorithms according to their performances to place more emphasis on the best-performing EA during the evolutionary process; and (3) applying a proposed local search to the best solution in a bid to obtain the optimal solution. The bi-EA was tested using both the first and second sets of test problems in order to compare its performance with those of the proposed MA and DE, and best state-of-the-art algorithms, respectively. The results proved its effectiveness for solving complex RCPSPs and showed that it performed better than both the MA and DE as bi-EA was able to obtain optimal solutions for more test problems than the MA and required less computational time than DE.

6.2 Conclusions

The proposed framework of the bi-EA involved multiple methodologies which were improved through the chapters of this thesis, examined using PSPLIB benchmark problems and showed to have the benefits of not only improving the quality of solutions but also reducing computational times.

In the following sub-sections, the conclusions drawn concerning each algorithm are discussed.

6.2.1 Genetic Algorithm (GA)

In Chapter 3, an experimental study of a GA for RCPSPs was conducted using three variants of a traditional GA with mutation rates of 0.2, 0.05 and adaptively reduced from 0.2 to 0.05, and four different algorithms selected from the literature. From the results, it could be concluded that a traditional GA with simple crossover and mutation operators and without any local search technique behaved in quite different ways for different problem sizes of RCPSP; for example, it was capable of producing good solutions for small problem instances such as J30 but, for medium (J60) and large ones (J90 and J120), performed less satisfactorily. Moreover, it was noted that its evolutionary process took too long to converge because of the lack of feasible solutions in the initial populations.

A novel heuristic repairing method for converting some infeasible solutions in the initial populations to feasible ones was proposed. A new MLS strategy was included in the GA for solving RCPSPs, with the purpose of avoiding the algorithm being trapped in local optima through moving its search to more promising areas instead of rediscovering the same area of the search space. Comparing the proposed MA with other classical GA variants proved the effectiveness of the proposed repairing method as it improved the performance of the GA, which can be calculated using equation 4.6, by 80.66% in terms of the quality of solutions obtained while the MLS technique had 11.83% and used 20.21% less CPU time than the classical GA.

6.2.2 Differential Evolution (DE) Algorithm

In Chapter 4, DE with improved crossover and mutation operators was integrated with the proposed heuristic repairing method and tested on the first dataset discussed in Section 6.1. The results obtained from DE with and without the proposed repairing method and those from four other algorithms from the literature, branch and bound (B&B), GA, Lagrange relaxation-based GA (GA_LR), branch and cut (B&C) and classical variants of GA and DE, were compared.

From this comparative study, it could be concluded that adopting the repairing method increased the percentage of quality solutions obtained by 28.96% and saved 10.62% of CPU time. Moreover, the improved DE saved 95% and 83% more CPU time than GA and GA_LR, respectively.

Comparing the improved DE with classical variants of GA and DE showed that the improved DE achieved 1.06%, 17.18%, 9.92% and 1.81% better quality of solutions for J30, J60, J90 and J120, respectively than GA and achieved 43%, 52.16%, 19.32% and 11.17% for J30, J60, J90 and J120, respectively better than DE. All improvement rates were calculated using equation 4.6 in Chapter 4.

6.2.3 Bi-evolutionary Algorithm (Bi-EA)

In Chapter 5, based on the results from Chapters 3 and 4, a new algorithm that utilized the power of multiple EAs (GA and DE) was developed. In it, an adaptive mechanism paid more attention to the best-performing EAs while heuristic methods were used to maintain the feasibility of the solutions in both initial population and new generated ones in each iteration. Bi-EA was tested using both the datasets described in Section 6.1 and compared with the previously proposed MA and DE, the four other algorithms mentioned in Section 6.2.2 and 18 other state-of-the-art algorithms.

This comparative study showed that bi-EA, MA and DE had same performance while dealing with small instances, such as J30. For larger ones, bi-EA outperformed both MA and DE in terms of solution quality, as the results of J60, J90 and J120 instances showed that bi-EA achieved better solutions-quality by 3.31%, 3.29% and 6.83% than MA, and

2.34%, 2.27% and 5.88% than DE, respectively. Moreover, the results demonstrated the effectiveness of the use of multiple algorithm strategy in solving RCPSPs as the proposed bi-EA was able to gather the good search features of both MA and DE. So, it was able to achieve better quality of solutions, 4.48 % on average, than GA (feature from DE) in lower computational times, 49.18 % on average, than DE (feature from GA). Consequently, bi-EA achieved very low average deviations in lower Fit_{Max} than both MA and DE for problems in different complexity levels. All improvement rates were calculated using equation 4.6 in Chapter 4.

Also, bi-EA showed competitive results against those from other state-of-the-art algorithms and achieved a very low average deviation from the best known solutions. However, it does not achieve the best performance among all. The reasons for this drawback might be that GA and DE used are not powerful enough. In addition, they are not complementary to each another. They cannot maintain sufficient diversity that could help them escape from local solutions.

6.3 Future Research Directions

The algorithms performed in this thesis could be extended in some of the following ways.

- The proposed bi-EA could be extended to solve multi-mode RCPSPs (MRCPSPs) which are more realistic and more complex than single-mode ones.
- The performance of the bi-EA for solving RCPSPs with uncertainty, where the durations of activities and their resource requirements are rarely definitively known, could be studied.
- The bi-EA could be further improved using multi-operator techniques such as multi-parent crossover (Elsayed et al., 2011a).
- More effective MLS strategies based on a single or multiple EAs could be developed.
- Some diversity techniques for maintaining the diversity of all the proposed algorithms, such as fitness sharing and clustering methods, could be adopted.

- The proposed algorithms could be extended to solve increased-demand multi-objective optimization problems.

References

Abdolshah, M 2014. A Review of Resource-Constrained Project Scheduling Problems (RCPSP) Approaches and Solutions.

Aerts, JC & Heuvelink, GB 2002. Using simulated annealing for resource allocation. *International Journal of Geographical Information Science,* 16**, 6,** 571-587.

Afshar-Nadjafi, B, Karimi, H, Rahimi, A & Khalili, S 2015. Project scheduling with limited resources using an efficient differential evolution algorithm. *Journal of King Saud University - Engineering Sciences,* 27**, 2,** 176-184.

Agarwal, A, Colak, S & Erenguc, S 2011. A neurogenetic approach for the resource-constrained project scheduling problem. *Computers & Operations Research,* 38**, 1,** 44-50.

Agarwal, A, Colak, S & Eryarsoy, E 2006. Improvement heuristic for the flow-shop scheduling problem: An adaptive-learning approach. *European Journal of Operational Research,* 169**, 3,** 801-815.

Agrawal, RB, Deb, K, Deb, K & Agrawal, RB 1995. Simulated Binary Crossover for Continuous Search Space. *Complex Systems,* 9**,** 115–148.

Ahn, CW & Ramakrishna, RS 2003. Elitism-based compact genetic algorithms. *Evolutionary Computation, IEEE Transactions on,* 7**, 4,** 367-385.

Ahuja, RK, Ergun, Ö, Orlin, JB & Punnen, AP 2002. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics,* 123**, 1,** 75-102.

Alcaraz, J & Maroto, C 2001. A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research,* 102**, 1-4,** 83-109.

Alvarez-Valdes, R & Tamarit, JM 1989. Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis. *Advances in project scheduling,* 134.

Artigues, C, Demassey, S & Néron, E 2013. *Resource-constrained project scheduling: models, algorithms, extensions and applications*, John Wiley & Sons.

Ashlock, D 2006. *Evolutionary computation for modeling and optimization*, Springer Science & Business Media.

Baar, T, Brucker, P & Knust, S 1999. *Tabu search algorithms and lower bounds for the resource-constrained project scheduling problem*, Springer.

Baccarini, D 1996. The concept of project complexity—a review. *International Journal of Project Management,* 14**, 4,** 201-204.

Bäck, T, Hammel, U & Schwefel, H-P 1997. Evolutionary computation: Comments on the history and current state. *Evolutionary computation, IEEE Transactions on,* 1**, 1,** 3-17.

Bai, Q 2010. Analysis of particle swarm optimization algorithm. *Computer and information science,* 3**, 1,** p180.

Baker, JE. Reducing bias and inefficiency in the selection algorithm. *Proceedings of the second international conference on genetic algorithms*, 1987. 14-21.

Bean, JC 1994. Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing,* 6**, 2,** 154-160.

Beheshti, Z & Shamsuddin, SMH 2013. A review of population-based meta-heuristic algorithms. *Int. J. Adv. Soft Comput. Appl,* 5**, 1,** 1-35.

Beni, G & Wang, J 1993. Swarm Intelligence in Cellular Robotic Systems. *In:* Dario, P, Sandini, G & Aebischer, P (eds.) *Robots and Biological Systems: Towards a New Bionics?* : Springer Berlin Heidelberg.

Berthold, T, Heinz, S, Lübbecke, ME, Möhring, RH & Schulz, J 2010. A constraint integer programming approach for resource-constrained project scheduling. *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems.* Springer.

Bettemir, ÖH & Sonmez, R 2015. Hybrid Genetic Algorithm with Simulated Annealing for Resource-Constrained Project Scheduling. *Journal of Management in Engineering,* 31**, 5,** 04014082.

Beyer, H-G & Schwefel, H-P 2002. Evolution strategies–A comprehensive introduction. *Natural computing,* 1**, 1,** 3-52.

Bierwirth, C & Mattfeld, DC 1999. Production scheduling and rescheduling with genetic algorithms. *Evolutionary computation,* 7**, 1,** 1-17.

Blackstone, JH, Phillips, DT & Hogg, GL 1982. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *The International Journal of Production Research,* 20**, 1,** 27-45.

Błażewicz, J, Ecker, KH, Pesch, E, Schmidt, G & Weglarz, J 2013. *Scheduling computer and manufacturing processes*, Springer Science & Business Media.

Blazewicz, J, Lenstra, JK & Kan, AR 1983. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics,* 5**, 1,** 11-24.

Blickle, T & Thiele, L 1995a. A comparison of selection schemes used in genetic algorithms. TIK-Report.

Blickle, T & Thiele, L 1995b. A Mathematical Analysis of Tournament Selection. *Proceedings of the 6th International Conference on Genetic Algorithms.* Morgan Kaufmann Publishers Inc.

Blum, C & Roli, A 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR),* 35**, 3,** 268-308.

Boctor, FF 1990. Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research,* 49**, 1,** 3-13.

Boctor, FF 1996. Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research,* 34**, 8,** 2335-2351.

Bouleimen, K & Lecocq, H 2003. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research,* 149**, 2,** 268-281.

Brucker, P, Drexl, A, Möhring, R, Neumann, K & Pesch, E 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research,* 112**, 1,** 3-41.

Brucker, P & Knust, S 1999. 2 Solving Large-Sized Resource-Constrained Project Scheduling Problems. *Project Scheduling.* Springer.

Brucker, P, Qu, R & Burke, E 2011. Personnel scheduling: Models and complexity. *European Journal of Operational Research,* 210**, 3,** 467-473.

Burke, R 2013. *Project management: planning and control techniques,* New Jersey, USA.

Chakrabortty, RK, Sarker, RA & Essam, DL. Resource constrained project scheduling: a branch and cut approach. *International Conference on Computers and Industrial Engineering,* 28-30 October 2015 *France*. Elsevier, In press.

Chandra, A & Chattopadhyay, S 2012. Role of mutation strategies of differential evolution algorithm in designing hardware efficient multiplier-less low-pass FIR filter. *Journal of Multimedia,* 7**, 5,** 353-363.

Chen, P-H & Shahandashti, SM 2009. Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints. *Automation in Construction,* 18**, 4,** 434-443.

Chen, R-M 2011. Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem. *Expert Systems with Applications,* 38**, 6,** 7102-7111.

Chen, R-M, Wu, C-L, Wang, C-M & Lo, S-T 2010. Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB. *Expert Systems with Applications,* 37**, 3,** 1899-1910.

Cheng, X & Wu, C 2006. Hybrid algorithm for complex project scheduling. *Computer Integrated Manufacturing Systems-Beijing,* 12**, 4,** 585.

Cho, J-H & Kim, Y-D 1997. A simulated annealing algorithm for resource constrained project scheduling problems. *Journal of the Operational Research Society***,** 736-744.

Christofides, N, Alvarez-Valdés, R & Tamarit, JM 1987. Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research,* 29**, 3,** 262-273.

Chudasama, C, Shah, S & Panchal, M. Comparison of parents selection methods of genetic algorithm for TSP. *International Conference on Computer Communication and Networks CSI-COMNET-2011, Proceedings*, 2011. 85-87.

Corder, GW & Foreman, DI 2009. Comparing two related samples: The Wilcoxon signed ranks test. *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach***,** 38-56.

Damak, N, Jarboui, B, Siarry, P & Loukil, T 2009. Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Computers & Operations Research,* 36**, 9,** 2653-2659.

Damay, J, Quilliot, A & Sanlaville, E 2007. Linear programming based algorithms for preemptive and non-preemptive RCPSP. *European Journal of Operational Research,* 182**, 3,** 1012-1022.

Darwin, C 1859. On the origins of species by means of natural selection. *London: Murray***,** 247.

Das, PP & Acharyya, S. Meta-heuristic approaches for solving Resource Constrained Project Scheduling Problem: A Comparative study. *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on*, 10-12 June 2011 2011a. 474-478.

Das, PP & Acharyya, S. Simulated annealing variants for solving resource constrained project scheduling problem: a comparative study. *Computer and Information Technology (ICCIT), 2011 14th International Conference on*, 2011b. IEEE, 469-474.

Das, S, Abraham, A, Chakraborty, UK & Konar, A 2009. Differential evolution using a neighborhood-based mutation operator. *Evolutionary Computation, IEEE Transactions on,* 13**, 3,** 526-553.

Das, S, Abraham, A & Konar, A 2008. Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. *Advances of Computational Intelligence in Industrial Systems.* Springer.

Das, S & Suganthan, PN 2011. Differential evolution: a survey of the state-of-the-art. *Evolutionary Computation, IEEE Transactions on,* 15**, 1,** 4-31.

Davis, EW & Patterson, JH 1975. A comparison of heuristic and optimum solutions in resource-constrained project scheduling. *Management science,* 21**, 8,** 944-955.

Davis, L 1991. Handbook of genetic algorithms.

Dawande, M, Geismar, HN, Sethi, SP & Sriskandarajah, C 2005. Sequencing and scheduling in robotic cells: Recent developments. *Journal of Scheduling,* 8**, 5,** 387-426.

De Carvalho, MFH & Haddad, RBB 2012. *Production scheduling on practical problems*, INTECH Open Access Publisher.

De Nijs, F. 2013. *Project Scheduling: The Impact of Instance Structure on Heuristic Performance.* TU Delft, Delft University of Technology.

Deb, K, Anand, A & Joshi, D 2002. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation,* 10**, 4,** 371-395.

Debels, D, De Reyck, B, Leus, R & Vanhoucke, M 2006. A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research,* 169**, 2,** 638-653.

Debels, D & Vanhoucke, M 2007. A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. *Operations Research,* 55**, 3,** 457-469.

Demeulemeester, E & Herroelen, W 1992. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management science,* 38**, 12,** 1803-1818.

Demeulemeester, EL & Herroelen, W 2002. *Project scheduling: a research handbook*, Springer Science & Business Media.

Deneubourg, J-L, Aron, S, Goss, S & Pasteels, JM 1990. The self-organizing exploratory pattern of the argentine ant. *Journal of insect behavior,* 3**, 2,** 159-168.

Dianati, M, Song, I & Treiber, M 2002. An introduction to genetic algorithms and evolution strategies. Citeseer.

Dorigo, M 1992. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy*.

Dorigo, M & Blum, C 2005. Ant colony optimization theory: A survey. *Theoretical Computer Science,* 344**, 2–3,** 243-278.

Dorigo, M, Caro, GD & Gambardella, LM 1999. Ant algorithms for discrete optimization. *Artificial life,* 5**, 2,** 137-172.

Dorigo, M, Maniezzo, V & Colorni, A 1996. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* 26**, 1,** 29-41.

Dorndorf, U & Pesch, E 1995. Evolution based learning in a job shop scheduling environment. *Computers & Operations Research,* 22**, 1,** 25-40.

Elfeky, E, Sarker, R & Essam, D 2008. Analyzing the Simple Ranking and Selection Process for Constrained Evolutionary Optimization. *Computer Science and Technology,* 23**, 1,** 19-34.

Elsayed, SM. 2012. *Evolutionary Approach for Constrained Optimization.* Australian Defence Force Academy.

Elsayed, SM, Sarker, RA & Essam, DL. GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems. *Evolutionary Computation (CEC), 2011 IEEE Congress on*, 5-8 June 2011 2011a. 1034-1040.

Elsayed, SM, Sarker, RA & Essam, DL 2011b. Multi-operator based evolutionary algorithms for solving constrained optimization problems. *Computers & Operations Research,* 38**, 12,** 1877-1896.

Espinosa-Aranda, JL, García-Ródenas, R, Ramírez-Flores, MDC, López-García, ML & Angulo, E 2015. High-speed railway scheduling based on user preferences. *European Journal of Operational Research,* 246**, 3,** 772-786.

Fahmy, A, Hassan, TM & Bassioni, H 2014. Improving RCPSP solutions quality with Stacking Justification–Application with particle swarm optimization. *Expert Systems with Applications,* 41**, 13,** 5870-5881.

Fang, C & Wang, L 2012. An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem. *Computers & Operations Research,* 39**, 5,** 890-901.

Fang, H-L. 1994. *Genetic algorithms in timetabling and scheduling.* Citeseer.

Fleszar, K & Hindi, KS 2004. Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research,* 155**, 2,** 402-413.

Fogel, DB & Fogel, LJ. An introduction to evolutionary programming. *Artificial Evolution*, 1996. Springer, 21-33.

Fogel, LJ, Owens, AJ & Walsh, MJ 1966. Artificial intelligence through simulated evolution.

Garey, MR & Johnson, DS 1975. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing,* 4**, 4,** 397-411.

Garey, MR & Johnson, DS 1979. Computers and intractability: a guide to the theory of NP-completeness. 1979. *San Francisco, LA: Freeman*.

Garfinkel, RS & Nemhauser, GL 1972. *Integer programming*, Wiley New York.

Garg, R & Mittal, S 2014. Effect of Local Search on the Performance of Genetic Algorithm.

Gavrilas, M. Heuristic and metaheuristic optimization techniques with application to power systems. *Proceedings of the 12th WSEAS international conference on Mathematical methods and computational techniques in electrical engineering*, 2010. World Scientific and Engineering Academy and Society (WSEAS), 95-103.

Giffler, B & Thompson, GL 1960. Algorithms for Solving Production-Scheduling Problems. *Operations Research,* 8**, 4,** 487-503.

Gigerenzer, G & Gaissmaier, W 2011. Heuristic Decision Making. *Annual Review of Psychology,* 62**, 1,** 451-482.

Glover, F 1989. Tabu search-part I. *ORSA Journal on computing,* 1**, 3,** 190-206.

Glover, F 1990a. Tabu search—part II. *ORSA Journal on computing,* 2**, 1,** 4-32.

Glover, F 1990b. Tabu search: A tutorial. *Interfaces,* 20**, 4,** 74-94.

Goldberg, DE, Korb, B & Deb, K 1989. Messy genetic algorithms: Motivation, analysis, and first results. *Complex systems,* 3**, 5,** 493-530.

Golmakani, HR & Namazi, A 2012. An artificial immune algorithm for multiple-route job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology,* 63**, 1-4,** 77-86.

Gonçalves, JF, Mendes, JDM & Resende, MG 2008. A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research,* 189**, 3,** 1171-1190.

Hall, NG, Kamoun, H & Sriskandarajah, C 1998. Scheduling in robotic cells: Complexity and steady state analysis. *European Journal of Operational Research,* 109**, 1,** 43-65.

Hartmann, S 1998. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL),* 45**, 7,** 733-750.

Hartmann, S 2002. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics (NRL),* 49**, 5,** 433-448.

Hartmann, S & Briskorn, D 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research,* 207**, 1,** 1-14.

Hartmann, S & Kolisch, R 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research,* 127**, 2,** 394-407.

Haupt, R 1989. A survey of priority rule-based scheduling. *Operations-Research-Spektrum,* 11**, 1,** 3-16.

Haupt, RL & Haupt, SE 2004. The binary genetic algorithm. *Practical Genetic Algorithms, Second Edition***,** 27-50.

Haupt, SE & Haupt, RL. Optimizing complex systems. *Aerospace Conference, IEEE*, 21-28 Mar. 1998. 241-247 vol.244.

Held, M & Karp, RM 1962. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics***,** 196-210.

Herroelen, W & Demeulemeester, E 1994. Resource-Constrained Project Scheduling A View on Recent Developments. *Tijdschrift voor Economie en Management, XXXIX, 371*, 395.

Holland, JH 1975. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence,* U Michigan Press.

Icmeli, O & Erenguc, SS 1994. A tabu search procedure for the resource constrained project scheduling problem with discounted cash flows. *Computers & Operations Research,* 21**, 8,** 841-853.

Jalilvand, A, Khanmohammadi, S & Shabaninia, F. Scheduling of sequence-dependant jobs on parallel multiprocessor systems using a branch and bound-based Petri net.

Emerging Technologies, 2005. Proceedings of the IEEE Symposium on, 2005. IEEE, 334-339.

Ji, D & Wang, XJ. An Efficient Evolutionary Programming. *Information Science and Engineering. ISISE '08. International Symposium on*, 20-22 Dec. 2008. 401-404.

Jia, D, Zheng, G & Khan, MK 2011. An effective memetic differential evolution algorithm based on chaotic local search. *Information Sciences*, 181**, 15,** 3175-3187.

Jourdan, L, Dhaenens, C & Talbi, E-G 2001. A genetic algorithm for feature selection in data-mining for genetics. *Proceedings of the 4th Metaheuristics International ConferencePorto (MIC'2001)***,** 29-34.

Józefowska, J, Mika, M, Różycki, R, Waligóra, G & Węglarz, J 2001. Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, 102**, 1-4,** 137-155.

Kelley, JE 1963. The critical-path method: Resources planning and scheduling. *Industrial scheduling*, 13**,** 347-365.

Kelley Jr, JE & Walker, MR. Critical-path planning and scheduling. *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference*, 1959. ACM, 160-173.

Kennedy, J 2010. Particle swarm optimization. *Encyclopedia of Machine Learning.* Springer.

Kennedy, J, Kennedy, JF, Eberhart, RC & Shi, Y 2001. *Swarm intelligence*, Morgan Kaufmann.

Kochetov, Y & Stolyar, A. Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. *Proceedings of the 3rd international workshop of computer science and information technologies*, 2003.

Kolisch, R 1995. Project scheduling under resource constraints: efficient heuristics for several problem classes. *Physica-Verlag, Wurzburg*.

Kolisch, R 1996a. Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, 14**, 3,** 179-192.

Kolisch, R 1996b. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research,* 90**, 2,** 320-333.

Kolisch, R & Drexl, A 1996. Adaptive search for solving hard project scheduling problems. *Naval Research Logistics (NRL)*, 43**, 1,** 23-40.

Kolisch, R & Hartmann, S 1999. Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. *In:* Węglarz, J (ed.) *Project Scheduling: Recent Models, Algorithms and Applications.* Boston, MA: Springer US.

Kolisch, R, Schwindt, C & Sprecher, A 1999. Benchmark instances for project scheduling problems. *Project Scheduling.* Springer.

Kolisch, R & Sprecher, A 1997. PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. *European journal of operational research,* 96**, 1,** 205-216.

Kolisch, R, Sprecher, A & Drexl, A 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management science,* 41**, 10,** 1693-1703.

Koné, O, Artigues, C, Lopez, P & Mongeau, M 2013. Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources. *Flexible Services and Manufacturing Journal,* 25**, 1-2,** 25-47.

Korejo, I, Yang, S & Li, C 2010. A directed mutation operator for real coded genetic algorithms. *Applications of Evolutionary Computation.* Springer.

Koza, JR & Poli, R 2005. Genetic Programming. *In:* Burke, EK & Kendall, G (eds.) *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques.* Boston, MA: Springer US.

Krichen, S & Chaouachi, J 2015. The Resource Constrained Project Scheduling Problem. 69-82.

Kurtulus, IS & Narula, SC 1985. Multi-project scheduling: Analysis of project performance. *IIE transactions,* 17**, 1,** 58-66.

Lawrence, S 1985. Resource constrained project scheduling–A computational comparison of heuristic scheduling techniques. Technical Report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh.

Lee, J-K & Kim, Y-D 1996. Search heuristics for resource constrained project scheduling. *Journal of the Operational Research Society***,** 678-689.

Leon, VJ & Balakrishnan, R 1995. Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. *Operations-Research-Spektrum,* 17**, 2-3,** 173-182.

Li, C, Bettati, R & Zhao, W. Static priority scheduling for ATM networks. *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, 1997. IEEE, 264-273.

Li, K & Willis, R 1992. An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research,* 56**, 3,** 370-379.

Lupetti, S & Zagorodnov, D. Data popularity and shortest-job-first scheduling of network transfers. *Digital Telecommunications,, 2006. ICDT'06. International Conference on*, 2006. IEEE, 26-26.

Magalhães-Mendes, J 2013. A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. *WSEAS transactions on computers,* 12**, 4,** 164-173.

Malcolm, DG, Roseboom, JH, Clark, CE & Fazar, W 1959. Application of a technique for research and development program evaluation. *Operations research,* 7**, 5,** 646-669.

Mccarthy, J 1989. Artificial Intelligence, Logic and Formalizing Common Sense. *In:* Thomason, RH (ed.) *Philosophical Logic and Artificial Intelligence.* Dordrecht: Springer Netherlands.

Mendes, JJDM, Gonçalves, JF & Resende, MG 2009. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research,* 36**, 1,** 92-109.

Meredith, JR & Mantel Jr, SJ 2011. *Project management: a managerial approach*, John Wiley & Sons.

Merkle, D, Middendorf, M & Schmeck, H 2002. Ant colony optimization for resource-constrained project scheduling. *Evolutionary Computation, IEEE Transactions on,* 6**, 4,** 333-346.

Michalewicz, Z 1992. *Genetic Algorithms + Data Structures = Evolution Programs,* New York, Springer-Verlag.

Michalewicz, Z 1996. Heuristic methods for evolutionary computation techniques. *Journal of Heuristics,* 1**, 2,** 177-206.

Miller, BL, Goldberg, DE & Goldberg, DE 1995. Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Systems,* 9**,** 193-212.

Mohebifar, A 2006. New binary representation in genetic algorithms for solvingTSP by mapping permutations to a list of ordered numbers. *WSEAS Transactions on Computers Research,* 1**, 2,** 114-118.

Moscato, P & Cotta, C 2010. A Modern Introduction to Memetic Algorithms. *In:* Gendreau, M & Potvin, J-Y (eds.) *Handbook of Metaheuristics.* Boston, MA: Springer US.

Moumene, K & Ferland, JA 2009. Activity list representation for a generalization of the resource-constrained project scheduling problem. *European Journal of Operational Research,* 199**, 1,** 46-54.

Munns, A & Bjeirmi, BF 1996. The role of project management in achieving project success. *International journal of project management,* 14**, 2,** 81-87.

Neri, F & Cotta, C 2012. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation,* 2**,** 1-14.

Neumann, K 1990. *Stochastic project networks: Temporal analysis, scheduling and cost minimization*, Springer Science & Business Media.

Noman, N & Iba, H 2008. Accelerating differential evolution using an adaptive local search. *Evolutionary Computation, IEEE Transactions on,* 12**, 1,** 107-125.

Nonobe, K & Ibaraki, T 2002. Formulation and tabu search algorithm for the resource constrained project scheduling problem. *Essays and surveys in metaheuristics.* Springer.

Norman, BA. 1995. *The random keys genetic algorithm for complex scheduling problems.*

Norman, BA & Bean, JC 1995. Random keys genetic algorithm for scheduling: unabridged version. *Ann Arbor,* 1001**,** 48109.

OğUz, O & Bala, H 1994. A comparative study of computational procedures for the resource constrained project scheduling problem. *European Journal of Operational Research,* 72**, 2,** 406-416.

Ólafsson, S 2006. Metaheuristics. *In:* Henderson, NA (ed.) *Handbook on Simulation.* Handbooks in Operations Research and Management Science: Elsevier.

Ozdamar, L 1999. A genetic algorithm approach to a general category project scheduling problem. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* 29**, 1,** 44-59.

Özdamar, L & Ulusoy, G 1996. An iterative local constraints based analysis for solving the resource constrained project scheduling problem. *Journal of Operations Management,* 14**, 3,** 193-208.

Palpant, M, Artigues, C & Michelon, P 2004. LSSPER: solving the resource-constrained project scheduling problem with large neighbourhood search. *Annals of Operations Research,* 131**, 1-4,** 237-257.

Panwalkar, SS & Iskander, W 1977. A Survey of Scheduling Rules. *Operations Research,* 25**, 1,** 45-61.

Patterson, JH 1976. Project scheduling: The effects of problem structure on heuristic performance. *Naval Research Logistics Quarterly,* 23**, 1,** 95-123.

Patterson, JH 1984a. A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem. *Manage. Sci.,* 30**, 7,** 854-867.

Patterson, JH 1984b. A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. *Management science,* 30**, 7,** 854-867.

Pinedo, ML 2012. *Scheduling: theory, algorithms, and systems*, Springer Science & Business Media.

Poli, R, Kennedy, J & Blackwell, T 2007. Particle swarm optimization. *Swarm intelligence,* 1**, 1,** 33-57.

Price, K, Storn, RM & Lampinen, JA 2006. *Differential evolution: a practical approach to global optimization*, Springer Science & Business Media.

Price, KV, Storn, RM & Lampinen, JA 2005. *Differential evolution: a practical approach to global optimization,* Berlin, Springer.

Pritsker, AaB, Waiters, LJ & Wolfe, PM 1969. Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. *Management Science,* 16**, 1,** 93-108.

Pukkala, T 2009. Population-based methods in the optimization of stand management. *Silva Fennica,* 43**, 2,** 261-274.

Radcliffe, NJ 1991. Equivalence Class Analysis Of Genetic Algorithms. *Complex Systems,* 5**, 2,** 183–205.

Rechenberg, I 1984. The evolution strategy. a mathematical model of darwinian evolution. *Synergetics—from microscopic to macroscopic order.* Springer.

Robbins, J 2008. An explanation of computer forensics. *National Forensics Center,* 774**,** 10-143.

Ronald, S. Robust encodings in genetic algorithms: a survey of encoding issues. *IEEE Conference on Evolutionary Computation*, 1997. 43-48.

Sabar, M, Montreuil, B & Frayret, J-M 2012. An agent-based algorithm for personnel shift-scheduling and rescheduling in flexible assembly lines. *Journal of Intelligent Manufacturing***,** 1-12.

Salman, AA & Hamdan, S 2012. Differential evolution-based algorithm for solving the department's course-scheduling problem. *Kuwait Journal of Science & Engineering,* 39**, 1 B,** 175-209.

Sampson, SE & Weiss, EN 1993. Local search techniques for the generalized resource constrained project scheduling problem. *Naval Research Logistics (NRL),* 40**, 5,** 665-675.

Sarker, R, Kamruzzaman, J & Newton, C 2003. Evolutionary optimization (EvOpt): a brief review and analysis. *International Journal of Computational Intelligence and Applications,* 3**, 04,** 311-330.

Schwefel, H-PP 1993. *Evolution and optimum seeking: the sixth generation,* John Wiley & Sons, Inc.

Sebt, MH & Alipouri, Y 2012. Solving resource-constrained project scheduling problem with evolutionary programming. *journal of the Operational Research Society,* 64**, 9,** 1327-1335.

Shan, M-C & Murphy, MC 1994. Method of automatically controlling the allocation of resources of a parallel processor computer system by calculating a minimum execution time of a task and scheduling subtasks against resources to execute the task in the minimum time. Google Patents.

Sivanandam, S & Deepa, S 2007. *Introduction to genetic algorithms*, Springer Science & Business Media.

Stinson, JP, Davis, EW & Khumawala, BM 1978. Multiple resource–constrained scheduling using branch and bound. *AIIE Transactions,* 10**, 3,** 252-259.

Storer, RH, Wu, SD & Vaccari, R 1992. New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling. *Management Science*, 38**, 10,** 1495-1509.

Storn, R. On the usage of differential evolution for function optimization. *Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS)*, 1996. 519–523.

Storn, R & Price, K 1995. *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces,* ICSI Berkeley.

Storn, R & Price, K 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization,* 11**, 4,** 341-359.

Thesen, A 1976. Heuristic scheduling of activities under resource and precedence restrictions. *Management science,* 23**, 4,** 412-422.

Thomas, PR & Salhi, S 1998. A tabu search approach for the resource constrained project scheduling problem. *Journal of Heuristics,* 4**, 2,** 123-139.

Tian, W & Demeulemeester, E 2013. Railway scheduling reduces the expected project makespan over roadrunner scheduling in a multi-mode project scheduling environment. *Annals of Operations Research,* 213**, 1,** 271-291.

Toklu, YC 2002. Application of genetic algorithms to construction scheduling with or without resource constraints. *Canadian Journal of Civil Engineering,* 29**, 3,** 421-429.

Tsutsui, S, Yamamura, M & Higuchi, T. Multi-parent Recombination with Simplex Crossover in Real Coded Genetic Algorithms. *Genetic Evolutionary Computation Conference*, 1999. 657–664.

Uckun, S, Bagchi, S, Kawamura, K & Miyabe, Y 1993. Managing genetic search in job shop scheduling. *IEEE Expert,* 8**, 5,** 15-24.

Ulusoy, G & Özdamar, L 1989. Heuristic performance and network/resource characteristics in resource-constrained project scheduling. *Journal of the Operational Research Society***,** 1145-1152.

Ulusoy, G & Özdamar, L 1994. A constraint-based perspective in resource constrained project scheduling. *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH,* 32**, 3,** 693-705.

Valls, V, Ballestin, F & Quintanilla, S 2008. A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research,* 185**, 2,** 495-508.

Vesterstrøm, J & Thomsen, R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. *Evolutionary Computation, 2004. CEC2004. Congress on*, 2004. IEEE, 1980-1987.

Walker, M 2001. Introduction to genetic programming. *Tech. Np: University of Montana*.

Whitehouse, GE & Brown, JR 1979. GENRES: An extension of Brooks algorithm for project scheduling with resource constraints. *Computers & Industrial Engineering,* 3**, 3,** 261-268.

Whitley, LD, Starkweather, T & Fuquay, DA. Scheduling problems and traveling salesmen: The genetic edge recombination operator. *ICGA*, 1989. 133-140.

Widmer, M, Hertz, A & Costa, D 2010. Metaheuristics and Scheduling. *Production Scheduling.* ISTE.

Wright, AH 1991. *Genetic algorithms for real parameter optimization,* Morgan Kaufmann, San Mateo, Morgan Kaufmann.

Xin, Y, Yong, L & Guangming, L 1999. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation,* 3**, 2,** 82-102.

Yan, R, Li, W, Jiang, P, Zhou, Y & Wu, G 2014. A Modified Differential Evolution Algorithm for Resource Constrained Multi-project Scheduling Problem. *Journal of Computers,* 9**, 8,** 1922-1927.

Yao, X & Liu, Y 1997. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks,* 8**, 3,** 694-713.

Yao, X, Liu, Y & Lin, G 1999. Evolutionary programming made faster. *Evolutionary Computation, IEEE Transactions on,* 3**, 2,** 82-102.

Yaseen, SG & Al-Slamy, NM 2008. Ant colony optimization. *IJCSNS,* 8**, 6,** 351.

Zamani, R 2013. A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem. *European Journal of Operational Research,* 229**, 2,** 552-559.

Zhang, C, Li, P, Rao, Y & Li, S 2005a. A New Hybrid GA/SA Algorithm for the Job Shop Scheduling Problem. *In:* Raidl, G & Gottlieb, J (eds.) *Evolutionary Computation in Combinatorial Optimization.* Springer Berlin Heidelberg.

Zhang, H 2011. Ant colony optimization for multimode resource-constrained project scheduling. *Journal of Management in Engineering,* 28**, 2,** 150-159.

Zhang, H, Li, H & Tam, C 2006. Particle swarm optimization for resource-constrained project scheduling. *International Journal of Project Management,* 24**, 1,** 83-92.

Zhang, H, Li, X, Li, H & Huang, F 2005b. Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction,* 14**, 3,** 393-404.

Zheng, X-L & Wang, L 2015. A multi-agent optimization algorithm for resource constrained project scheduling problem. *Expert Systems with Applications,* 42**, 15,** 6039-6049.

Ziarati, K, Akbari, R & Zeighami, V 2011. On the performance of bee algorithms for resource-constrained project scheduling problem. *Applied Soft Computing,* 11**, 4,** 3720-3733.

Appendices

In this section, the detailed results from all proposed algorithms including the best, median, mean, worst, standard deviation (STD), average CPU time (t) per run in seconds, average deviation of the best solution obtained by the proposed algorithms from the optimal solutions (LB_{OP}) and from the critical path lower bound solution (LB_{CP}) are presented in four appendices.

1. Appendix A: computational results of 16 problems from J30 instance and 15 from each of J60, J90 and J120 ones obtained by the proposed MA discussed in Chapter 3.
2. Appendix B: computational results of 16 problems from J30 instance and 15 from each of J60, J90 and J120 ones obtained by the improved DE discussed in Chapter 4.
3. Appendix C: computational results of 16 problems from J30 instance and 15 from each of J60, J90 and J120 ones obtained by the proposed bi-evolutionary (bi-EA) algorithm discussed in Chapter 5.
4. Appendix D: computational results of 480 problems from each of J30, J60 and J90, and 600 problems of J120 with a total of 2040 problems obtained by the proposed bi-evolutionary (bi-EA) algorithm.

Appendix A

Table 1

The detailed results of 16 problems of J30 obtained by MA

The results are obtained from 30 runs with up to $n \times 10{,}000$ fitness evaluations for each, where n=30.

Prob. No	Best	Median	Mean	Worst	STD	t	LB_{OP}	LB_{CP}
J30, 1-1	43	49	47.57	53	4.27	5.28	0	0
J30, 1-2	47	48	47.93	54	1.28	3.11	0	0
J30, 1-3	47	47	48.20	51	1.86	3.81	0	0
J30, 1-4	62	62	62.10	64	0.40	0.21	0	0
J30, 1-5	39	40	39.93	40	0.25	3.04	0	0
J30, 2-1	38	38	38	38	0	6.02	0	0
J30, 2-2	51	51	51.93	53	1.01	0.73	0	0
J30, 2-3	43	43	43	43	0	0.22	0	0
J30, 2-4	43	43	43	43	0	0.11	0	0
J30, 2-5	51	51	51	51	0	0.31	0	0
J30, 3-1	72	72	72	72	0	0.20	0	0
J30, 3-2	40	40	40	40	0	0.14	0	0
J30, 3-3	57	57	57	57	0	0.10	0	0
J30, 3-4	98	98	98	98	0	0.07	0	0
J30, 3-5	53	53	53	53	0	0.16	0	0
J30, 4-1	49	49	49	49	0	0.18	0	0
Total Average					**0.57**	**1.48**	**0**	**0**

Table 2

The detailed results of 15 problems of J60 obtained by MA

The results are obtained from 30 runs with up to $n \times 10{,}000$ fitness evaluations for each, where n=60.

Prob. No	Best	Median	Mean	Worst	STD	t	LB_{OP}	LB_{CP}
J60, 1-1	77	82	80.57	90	3	22.89	0	0.00
J60, 1-2	70	75	75.5	88	3.43	27.55	2.94	7.69
J60, 1-3	70	76	74.88	78	2.07	7.99	2.94	4.48
J60, 1-4	91	93	93.68	100	1.95	8.06	0	15.19
J60, 1-5	76	81	80.67	83	1.67	15.81	4.11	11.76
J60, 2-1	65	65	65.77	69	1.43	11.99	0	0.00
J60, 2-2	82	82	82	82	0	16.1	0	0.00
J60, 2-3	78	78	78	78	0	5.67	0	1.30
J60, 2-4	78	78	78	78	0	3.92	0	0.00
J60, 2-5	54	54	54.57	59	1.14	32.55	0	1.89
J60, 3-1	60	60.5	61.4	68	1.87	16.57	0	0.00
J60, 3-2	69	69	69	69	0	3.33	0	0.00
J60, 3-3	105	105	105	105	0	39.55	0	2.94
J60, 3-4	81	81	81	81	0	6.13	0	0.00
J60, 3-5	83	83	83	83	0	3.51	0	0.00
Total Average					**1.10**	**14.77**	**0.67**	**3.02**

Table 3

The detailed results of 15 problems of J90 obtained by MA

The results are obtained from 30 runs with up to $n \times 10{,}000$ fitness evaluations for each, where n=90.

Prob. No	Best	Median	Mean	Worst	STD	t	LB_{OP}	LB_{CP}
J90, 1-1	85	91	91.1	98	3.59	20.68	16.44	26.87
J90, 1-2	99	105	104.7	113	3.75	21.15	7.61	12.5
J90, 1-3	71	79	77.93	84	3.23	12.99	7.58	20.34
J90, 1-4	92	98	97.6	104	2.9	26.31	6.98	21.05
J90, 1-5	89	97.5	96.67	107	5.18	14.19	2.3	5.95
J90, 2-1	96	96	96	96	0	3.19	0	0
J90, 2-2	114	114	114	114	0	2.08	0	0
J90, 2-3	75	75	75.57	77	0.68	13.71	0	0
J90, 2-4	70	70	70.13	73	0.57	9.61	0	0
J90, 2-5	100	101	101.2	105	1.52	18.61	0	0
J90, 3-1	81	81	81	81	0	3.29	0	0
J90, 3-2	84	84	84	84	0	1.21	0	0
J90, 3-3	71	72	72.03	75	1.13	22.62	0	0
J90, 3-4	104	104	104	104	0	1.03	0	0
J90, 3-5	75	77	77.7	83	1.7	19.49	0	0
Total Average					**1.62**	**12.68**	**2.73**	**5.78**

Table 4

The detailed results of 15 problems of J120 obtained by MA

The results are obtained from 30 runs with up to n × 10,000 fitness evaluations for each, where n=120.

Prob. No	Best	Median	Mean	Worst	STD	t	LB_{OP}	LB_{CP}
J120, 1-1	129	135.5	137.07	152	5.19	44.51	22.86	30.30
J120, 1-2	135	141	142.67	157	5.29	27.58	23.85	56.98
J120, 1-3	142	151	151.13	163	5.90	25.14	13.6	73.17
J120, 1-4	117	124.5	124.53	135	3.75	35.75	20.62	48.10
J120, 1-5	140	147	147.97	165	5.19	27.27	25	48.94
J120, 2-1	99	105.5	105.10	111	3.03	33.94	13.79	41.43
J120, 2-2	88	94	95.50	105	4.34	39.12	17.33	20.55
J120, 2-3	116	120	120.43	128	2.67	28.53	26.09	48.72
J120, 2-4	114	122	122.13	132	4.88	18.52	20	29.55
J120, 2-5	124	133	132.17	139	4.09	31.12	20.39	36.26
J120, 3-1	96	101.5	101.63	110	3.70	16.77	20	21.52
J120, 3-2	93	99.5	99.67	107	3.31	14.26	5.68	5.68
J120, 3-3	105	112	111.37	117	3.07	26.64	5	5.00
J120, 3-4	82	87	86.87	92	2.64	23.38	15.49	15.49
J120, 3-5	91	95	95.03	106	3.24	39.79	8.33	12.35
Total Average					**4.02**	**28.82**	**17.20**	**32.94**

Appendix B

Table 5

The detailed results of 16 problems of J30 obtained by improved DE

The results are obtained from 30 runs with up to $n \times 10{,}000$ fitness evaluations for each, where n=30.

Prob. No	**Best**	**Median**	**Mean**	**Worst**	**STD**	$\boldsymbol{t}$	$\boldsymbol{LB_{OP}}$	$\boldsymbol{LB_{CP}}$
J30, 1-1	43	43	43	43	0	90.5654	0	0
J30, 1-2	47	47	47	47	0	29.2564	0	0
J30, 1-3	47	47	47	47	0	26.8152	0	0
J30, 1-4	62	62	62	62	0	40.0727	0	0
J30, 1-5	39	39	39	39	0	62.7958	0	0
J30, 2-1	38	38	38	38	0	12.778	0	0
J30, 2-2	51	51	51	51	0	203.8402	0	0
J30, 2-3	43	43	43	43	0	23.7015	0	0
J30, 2-4	43	43	43	43	0	10.9883	0	0
J30, 2-5	51	51	51	51	0	23.7808	0	0
J30, 3-1	72	72	72	72	0	25.6995	0	0
J30, 3-2	40	40	40	40	0	18.9279	0	0
J30, 3-3	57	57	57	57	0	12.9811	0	0
J30, 3-4	98	98	98	98	0	9.115	0	0
J30, 3-5	53	53	53	53	0	10.0562	0	0
J30, 4-1	83	83	83	83	0	3.507305	0	0
Total Average					**0**	**37.81**	**0**	**0**

Table 6

The detailed results of 15 problems of J60 obtained by improved DE

The results are obtained from 30 runs with up to $n \times 10{,}000$ fitness evaluations for each, where n=60.

Prob. No	Best	Median	Mean	Worst	STD	t	LB_{OP}	LB_{CP}
J60, 1-1	77	77	77	77	0	179.6	0	0
J60, 1-2	70	70	70	70	0	891.05	2.94	7.69
J60, 1-3	71	71	71	71	0	785.1	4.41	5.97
J60, 1-4	93	93	93	93	0	375.94	2.2	17.72
J60, 1-5	73	73	73	73	0	389.18	0	7.35
J60, 2-1	65	65	65	65	0	136.79	0	0
J60, 2-2	82	82	82	82	0	164.07	0	0
J60, 2-3	78	78	78	78	0	120.22	0	1.3
J60, 2-4	78	78	78	78	0	130.37	0	0
J60, 2-5	54	54	54	54	0	219.72	0	1.89
J60, 3-1	60	60	60	60	0	316.16	0	0
J60, 3-2	69	69	69	69	0	104.78	0	0
J60, 3-3	105	105	105	105	0	36.7	0	2.94
J60, 3-4	81	81	81	81	0	92.21	0	0
J60, 3-5	83	83	83	83	0	99.85	0	0
Total Average					**0**	**269.45**	**0.64**	**2.99**

Table 7

The detailed results of 15 problems of J90 obtained by improved DE

The results are obtained from 30 runs with up to $n \times 10{,}000$ fitness evaluations for each, where n=90.

Prob. No	Best	Median	Mean	Worst	STD	t	LB_{OP}	LB_{CP}
J90, 1-1	80	80	80	80	0.00	1815.03	17.81	19.40299
J90, 1-2	92	92	92	92	0.00	1923.39	8.70	4.545455
J90, 1-3	70	70	70	70	0.00	1668.02	10.61	18.64407
J90, 1-4	105	105	105	105	0.00	1553.28	8.14	38.15789
J90, 1-5	87	87	87	87	0.00	1273.52	3.45	3.571429
J90, 2-1	96	96	96	96	0.00	184.00	0.00	0
J90, 2-2	114	114	114	114	0.00	409.28	0.00	0
J90, 2-3	75	75	75	75	0.00	290.51	0.00	0
J90, 2-4	70	70	70	70	0.00	402.14	0.00	0
J90, 2-5	100	100	100	100	0.00	1002.71	1.00	0
J90, 3-1	81	81	81	81	0.00	216.75	0.00	0
J90, 3-2	84	84	84	84	0.00	257.95	0.00	0
J90, 3-3	72	72	72	72	0.00	1159.55	5.63	1.408451
J90, 3-4	104	104	104	104	0.00	132.90	0.00	0
J90, 3-5	75	75	75	75	0.00	1067.98	1.33	0
Total Average					**0.00**	**890.47**	**3.78**	**5.72**

Table 8

The detailed results of 15 problems of J120 obtained by improved DE

The results are obtained from 30 runs with up to $n \times 10{,}000$ fitness evaluations for each, where n=60.

Prob. No	Best	Median	Mean	Worst	STD	t	LB_{OP}	LB_{CP}
J120, 1-1	129	129	129	129	0	4249.45	22.86	30.3
J120, 1-2	142	142	142	142	0	3697.89	30.28	65.12
J120, 1-3	137	137	137	137	0	2732.47	9.6	67.07
J120, 1-4	121	121	121	121	0	2313.81	24.74	53.16
J120, 1-5	127	127	127	127	0	4131.49	13.39	35.11
J120, 2-1	100	100	100	100	0	2233.82	14.94	42.86
J120, 2-2	91	91	91	91	0	2470.03	21.33	24.66
J120, 2-3	110	110	110	110	0	3205.18	19.57	41.03
J120, 2-4	109	109	109	109	0	2950.46	14.74	23.86
J120, 2-5	132	132	132	132	0	1646.67	28.16	45.05
J120, 3-1	95	95	95	95	0	1964.78	18.75	20.25
J120, 3-2	91	91	91	91	0	2175.33	3.41	3.41
J120, 3-3	106	106	106	106	0	2119.1	6	6
J120, 3-4	80	80	80	80	0	2287.51	12.68	12.68
J120, 3-5	96	96	96	96	0	1375.52	14.29	18.52
Total Average					**0**	**2636.90**	**16.98**	**32.61**

Appendix C

Table 9

The detailed results of 16 problems of J30 obtained by bi-EA

The results are obtained from 30 runs with up to $n \times 10{,}000$ fitness evaluations for each, where n=30.

Prob. No	**Best**	**Median**	**Mean**	**Worst**	**STD**	t	LB_{OP}	LB_{CP}
J30, 1-1	43	43	43	43	0	49.83	0	0
J30, 1-2	47	48	47.67	49	0.55	40.88	0	0
J30, 1-3	47	47	47	47	0	20.21	0	0
J30, 1-4	62	62	62.53	64	0.63	46.93	0	0
J30, 1-5	39	40	40.17	44	0.91	45.69	0	0
J30, 2-1	38	38	38	38	0	25.35	0	0
J30, 2-2	51	53	52.23	53	0.97	35.58	0	0
J30, 2-3	43	43	43	43	0	21.89	0	0
J30, 2-4	43	43	43	43	0	9.57	0	0
J30, 2-5	51	51	51	51	0	12.33	0	0
J30, 3-1	72	72	72	72	0	10.7	0	0
J30, 3-2	40	40	40	40	0	12.32	0	0
J30, 3-3	57	57	57	57	0	6.67	0	0
J30, 3-4	98	98	98	98	0	4.16	0	0
J30, 3-5	53	53	53	53	0	7.6	0	0
J30, 4-1	83	83	83	83	0	10.42	0	0
Total Average					**0.19**	**22.51**	**0**	**0**

Table 10

The detailed results of 15 problems of J60 obtained by bi-EA

The results are obtained from 30 runs with up to n × 10,000 fitness evaluations for each, where n=60.

Prob. No	Best	Median	Mean	Worst	STD	t	LB_{OP}	LB_{CP}
J60, 1-1	77	80	80.27	86	2.8	251.58	0	0.00
J60, 1-2	70	70	70	70	0	332.1	2.94	7.69
J60, 1-3	70	73	73.27	78	2.53	261.3	2.94	4.48
J60, 1-4	91	93	92.8	95	0.96	219.7	0	15.19
J60, 1-5	75	78	78.27	84	2.41	294.15	2.74	10.29
J60, 2-1	65	65	65	65	0	158.63	0	0.00
J60, 2-2	82	82	82	82	0	87.8	0	0.00
J60, 2-3	78	79	79.07	83	1.23	140.23	0	1.30
J60, 2-4	78	78	78	78	0	86.96	0	0.00
J60, 2-5	54	55	55.47	60	1.33	251.77	0	1.89
J60, 3-1	60	62	62.37	67	1.61	256.47	0	0.00
J60, 3-2	69	69	69	69	0	59.2	0	0.00
J60, 3-3	105	105	105	105	0	36.66	0	2.94
J60, 3-4	81	81	81	81	0	92.36	0	0.00
J60, 3-5	83	83	83	83	0	74.25	0	0.00
Total Average					**0.86**	**173.54**	**0.57**	**2.92**

Table 11

The detailed results of 15 problems of J90 obtained by bi-EA

The results are obtained from 30 runs with up to n × 10,000 fitness evaluations for each, where n=90.

Prob. No	Best	Median	Mean	Worst	STD	t	LB_{OP}	LB_{CP}
J90, 1-1	85	92.5	92.47	105	4.28	516.38	16.44	26.87
J90, 1-2	93	103	103.27	118	4.9	523.95	1.09	5.68
J90, 1-3	74	80	80.73	92	4.11	470.71	12.12	25.42
J90, 1-4	92	100.5	101.1	111	4.43	350.76	6.98	21.05
J90, 1-5	88	95	95.7	106	4.65	519.88	1.15	4.76
J90, 2-1	96	96	96	96	0	153.11	0	0
J90, 2-2	114	114	114.13	116	0.43	197.41	0	0
J90, 2-3	75	78.5	79	86	3.28	309.06	0	0
J90, 2-4	70	71.5	71.9	75	1.79	430.59	0	0
J90, 2-5	100	103	103.13	110	2.96	506.29	0	0
J90, 3-1	81	81	81	81	0	177.34	0	0
J90, 3-2	84	84	84	84	0	171.75	0	0
J90, 3-3	71	75	75.27	82	2.73	302.74	0	0
J90, 3-4	104	104	104	104	0	106.72	0	0
J90, 3-5	75	77	77.13	83	1.87	730.31	0	0
Total Average					**2.36**	**364.47**	**2.52**	**5.59**

Table 12

The detailed results of 15 problems of J120 obtained by bi-EA

The results are obtained from 30 runs with up to n × 10,000 fitness evaluations for each, where n=120.

Prob. No	Best	Median	Mean	Worst	STD	t	LB_{OP}	LB_{CP}
J120, 1-1	133	147.5	146.5	166	7.9	892.9	26.67	34.34
J120, 1-2	137	150	150.3	168	8	836.3	25.69	59.30
J120, 1-3	134	149	149	169	9.2	1255	7.20	63.41
J120, 1-4	121	133.5	134	148	6.6	1307.2	24.74	53.16
J120, 1-5	132	150	149.7	164	7.9	840.7	17.86	40.43
J120, 2-1	96	104	105	121	6.1	1013.3	10.34	37.14
J120, 2-2	87	93	95.1	115	6.2	681.7	16.00	19.18
J120, 2-3	111	124	124.1	136	7.4	1165	20.65	42.31
J120, 2-4	112	124	124.2	141	7.5	1249.8	17.89	27.27
J120, 2-5	125	138.5	139	153	7.6	1105.5	21.36	37.36
J120, 3-1	95	100	100.4	113	4.5	683.3	18.75	20.25
J120, 3-2	96	100	102.9	118	6	855.6	9.09	9.09
J120, 3-3	100	106	106.8	114	3.3	1058.9	0.00	0.00
J120, 3-4	77	85.5	85.5	92	4.2	1120.6	8.45	8.45
J120, 3-5	88	97	97.6	110	5.3	1130.2	4.76	8.64
Total Average					**6.50**	**1013.06**	**15.30**	**30.69**

Appendix D

In the following tables, the number of fitness evaluations (FE) used by bi-EA for solving each problem is also given.

Table 13

The detailed results of 480 problems of J30 obtained by bi-EA

The results are obtained from 30 runs with up to 50,000 fitness evaluations for each.

Prob. No	Best	Median	Mean	Worst	STD	t	FE	LB_{OP}	LB_{CP}
J30,1-1	43	43	43	43	0	2.81	200	0	0
J30,1-2	47	47	47	47	0	1.83	199	0	0
J30,1-3	47	47	47	47	0	1.52	199	0	0
J30,1-4	62	63	62.53	63	0.51	67.72	133	0	0
J30,1-5	39	39	39.23	40	0.43	35.55	137	0	0
J30,1-6	48	49	48.63	49	0.49	81.28	139	0	0
J30,1-7	60	60	60	60	0	1.11	133	0	0
J30,1-8	53	53	53.1	56	0.55	19.46	133	0	0
J30,1-9	49	49	49.47	52	0.82	43.42	209	0	0
J30,1-10	45	45	45.1	46	0.31	15.06	134	0	0
J30,2-1	38	38	38	38	0	1.24	133	0	0
J30,2-2	51	51	51	51	0	3.47	135	0	0
J30,2-3	43	43	43	43	0	2.55	134	0	0
J30,2-4	43	43	43	43	0	0.99	132	0	0
J30,2-5	51	51	51	51	0	1.02	133	0	0
J30,2-6	47	47	47	47	0	1	133	0	0
J30,2-7	47	47	47	47	0	1.03	133	0	0
J30,2-8	54	54	54	54	0	1.03	133	0	0
J30,2-9	54	54	54	54	0	1.14	133	0	0
J30,2-10	43	43	43	43	0	5.35	133	0	0
J30,3-1	72	72	72	72	0	0.98	133	0	0
J30,3-2	40	40	40	40	0	0.95	133	0	0
J30,3-3	57	57	57	57	0	1.05	133	0	0
J30,3-4	98	98	98	98	0	0.99	133	0	0
J30,3-5	53	53	53	53	0	1.04	132	0	0
J30,3-6	54	54	54	54	0	0.98	133	0	0
J30,3-7	48	48	48	48	0	0.96	132	0	0
J30,3-8	54	54	54	54	0	1.01	132	0	0
J30,3-9	65	65	65	65	0	1.07	133	0	0

J30,3-10	59	59	59	59	0	1.03	132	0	0
J30,4-1	49	49	49	49	0	0.96	132	0	0
J30,4-2	60	60	60	60	0	1.02	133	0	0
J30,4-3	47	47	47	47	0	1.02	133	0	0
J30,4-4	57	57	57	57	0	0.97	132	0	0
J30,4-5	59	59	59	59	0	1.02	132	0	0
J30,4-6	45	45	45	45	0	0.96	133	0	0
J30,4-7	56	56	56	56	0	0.96	132	0	0
J30,4-8	55	55	55	55	0	0.99	133	0	0
J30,4-9	38	38	38	38	0	1.03	133	0	0
J30,4-10	48	48	48	48	0	1.07	133	0	0
J30,5-1	53	53	53.77	58	1.55	58.35	783	0	0
J30,5-2	82	83	83.2	85	1.1	101.1	200	0	0
J30,5-3	76	82	80.5	83	2.47	108.49	1635	0	0
J30,5-4	63	65	64.97	69	1.75	72	542	0	0
J30,5-5	76	79	77.77	81	1.61	44.46	77	0	0
J30,5-6	64	66.5	66.7	72	2.2	84.9	1066	0	0
J30,5-7	76	81	80.3	85	2.28	99.77	11477	0	0
J30,5-8	67	67	69.33	75	2.83	49.81	349	0	0
J30,5-9	49	50	50.3	53	0.84	90.56	2895	0	0
J30,5-10	70	72	71.77	79	1.96	49.19	346	0	0
J30,6-1	59	60	59.7	61	0.65	49.64	420	0	0
J30,6-2	51	53	52.77	57	1.19	60.21	387	0	0
J30,6-3	48	49	49	50	0.95	45.09	499	0	0
J30,6-4	42	42	42.67	45	0.88	38.38	78	0	0
J30,6-5	67	67	67	67	0	1.71	80	0	0
J30,6-6	37	37	37.3	39	0.6	20.62	77	0	0
J30,6-7	46	46	46	46	0	3.93	77	0	0
J30,6-8	39	41	40.83	43	1.29	58.07	496	0	0
J30,6-9	51	51	51	51	0	4.28	76	0	0
J30,6-10	61	61	61.97	66	1.52	42.26	229	0	0
J30,7-1	55	55	55	55	0	0.56	76	0	0
J30,7-2	42	42	42	42	0	1.13	76	0	0
J30,7-3	42	42	42	42	0	0.72	76	0	0
J30,7-4	44	44	44.47	45	0.51	30.47	77	0	0
J30,7-5	44	44	44.33	45	0.48	27.1	191	0	0
J30,7-6	35	35	35	35	0	0.57	76	0	0
J30,7-7	50	50	50.7	53	1.29	16.95	77	0	0
J30,7-8	44	44	44	44	0	0.89	76	0	0
J30,7-9	60	60	60	60	0	0.68	77	0	0
J30,7-10	49	49	49.27	51	0.69	9.72	77	0	0
J30,8-1	44	44	44	44	0	0.57	76	0	0
J30,8-2	51	51	51	51	0	0.51	76	0	0
J30,8-3	53	53	53	53	0	0.52	76	0	0
J30,8-4	48	48	48	48	0	0.55	76	0	0
J30,8-5	58	58	58	58	0	0.57	76	0	0
J30,8-6	47	47	47	47	0	0.54	76	0	0
J30,8-7	41	41	41	41	0	0.53	76	0	0
J30,8-8	51	51	51	51	0	0.55	76	0	0
J30,8-9	39	39	39	39	0	0.52	76	0	0
J30,8-10	67	67	67	67	0	0.51	76	0	0
J30,9-1	83	83	83.27	88	0.94	15.39	233	0	0
J30,9-2	92	94	95.97	103	3.07	66.32	268	0	0

J30,9-3	70	73	73.07	78	2.16	94.65	11862	2.94	2.94
J30,9-4	72	74	73.83	77	1.34	88.18	11781	1.41	1.41
J30,9-5	70	72	72.07	78	2.33	41.62	124	0	0
J30,9-6	59	63	63	67	2.15	84	1179	0	0
J30,9-7	65	68	67.6	72	1.81	94.4	11935	3.17	3.17
J30,9-8	91	92	91.97	93	0.41	68.49	2172	0	0
J30,9-9	64	65	65.57	69	1.45	89.16	11976	1.59	1.59
J30,9-10	88	90	89.6	91	0.89	68.23	647	0	0
J30,10-1	42	42	42.03	43	0.18	8.67	83	0	0
J30,10-2	56	57	56.9	58	0.48	72.32	1447	0	0
J30,10-3	62	63.5	63.47	64	0.57	94.51	5207	0	0
J30,10-4	58	59	59.07	63	1.2	80.83	652	0	0
J30,10-5	41	42	41.63	43	0.61	73.31	274	0	0
J30,10-6	44	45	45.17	47	0.75	93.06	690	0	0
J30,10-7	49	49	49.13	50	0.35	25.31	159	0	0
J30,10-8	54	54	54.6	58	0.93	46.39	115	0	0
J30,10-9	49	49	49	49	0	0.8	76	0	0
J30,10-10	41	42	42.03	44	0.49	94.29	3962	0	0
J30,11-1	54	55	54.6	56	0.56	58.06	611	0	0
J30,11-2	56	56	56.43	58	0.82	31.92	77	0	0
J30,11-3	81	81	81	81	0	0.8	76	0	0
J30,11-4	63	63	63.03	64	0.18	5.34	77	0	0
J30,11-5	49	50	49.87	52	0.86	63.65	420	0	0
J30,11-6	44	44	44	44	0	7.02	77	0	0
J30,11-7	36	36	36.17	37	0.38	29.72	275	0	0
J30,11-8	62	62	62	62	0	1.98	77	0	0
J30,11-9	67	67	67	67	0	0.8	76	0	0
J30,11-10	38	38	38	38	0	2.54	81	0	0
J30,12-1	47	47	47	47	0	0.74	76	0	0
J30,12-2	46	46	46	46	0	0.76	76	0	0
J30,12-3	37	37	37	37	0	0.71	76	0	0
J30,12-4	63	63	63	63	0	0.73	76	0	0
J30,12-5	47	47	47	47	0	0.7	76	0	0
J30,12-6	53	53	53	53	0	0.76	76	0	0
J30,12-7	55	55	55	55	0	0.72	76	0	0
J30,12-8	35	35	35	35	0	0.68	76	0	0
J30,12-9	52	52	52	52	0	0.77	76	0	0
J30,12-10	57	57	57	57	0	0.7	76	0	0
J30,13-1	60	62	61.77	64	0.86	127.94	11788	3.45	3.45
J30,13-2	62	64	64.27	68	1.2	126.07	5778	0	0
J30,13-3	76	80	80.2	85	2.14	131.66	3422	0	0
J30,13-4	73	75	74.87	77	1.04	120.86	11783	1.39	1.39
J30,13-5	68	70	70.47	73	1.53	132.94	11706	1.49	1.49
J30,13-6	65	66	66.53	69	1.11	126.63	11784	1.56	1.56
J30,13-7	78	82	81.9	86	1.56	117.3	11676	1.3	1.3
J30,13-8	107	113	111.97	118	2.99	102.93	11667	0.94	0.94
J30,13-9	72	74.5	74.4	78	2.28	123.42	11934	1.41	1.41
J30,13-10	64	65	66.17	73	2.41	110.73	2585	0	0
J30,14-1	50	50.5	50.63	53	0.76	59.98	82	0	0
J30,14-2	53	54	54.07	56	0.78	98.76	1940	0	0
J30,14-3	58	60	59.8	61	0.96	115.39	7488	0	0
J30,14-4	50	51	51.17	54	0.91	88.01	1027	0	0
J30,14-5	52	54	53.8	56	1.03	105.86	7715	0	0

J30,14-6	35	35	35.6	38	0.77	70.52	652	0	0
J30,14-7	50	51	51.27	54	1.05	97.46	1530	0	0
J30,14-8	54	54	54	54	0	1.44	76	0	0
J30,14-9	47	48	47.7	50	0.65	102.39	11478	2.17	2.17
J30,14-10	61	61	61.33	63	0.76	28.88	153	0	0
J30,15-1	46	46	46	46	0	0.89	76	0	0
J30,15-2	47	47	47	47	0	0.8	76	0	0
J30,15-3	48	48	48	48	0	0.91	76	0	0
J30,15-4	48	48	48	48	0	0.77	76	0	0
J30,15-5	58	59	59.33	61	0.76	106.9	5588	0	0
J30,15-6	67	67	67	67	0	0.85	76	0	0
J30,15-7	47	47	47	47	0	1.42	76	0	0
J30,15-8	50	50	50	50	0	1.66	77	0	0
J30,15-9	54	54	54	54	0	0.88	76	0	0
J30,15-10	65	65	65	65	0	0.92	76	0	0
J30,16-1	51	51	51	51	0	0.83	76	0	0
J30,16-2	48	48	48	48	0	0.78	76	0	0
J30,16-3	36	36	36	36	0	0.79	76	0	0
J30,16-4	47	47	47	47	0	0.74	76	0	0
J30,16-5	51	51	51	51	0	0.76	76	0	0
J30,16-6	51	51	51	51	0	0.72	76	0	0
J30,16-7	34	34	34	34	0	0.73	76	0	0
J30,16-8	44	44	44	44	0	0.72	76	0	0
J30,16-9	44	44	44	44	0	0.77	76	0	0
J30,16-10	51	51	51	51	0	0.78	76	0	0
J30,17-1	64	64	64.87	66	1.01	37.17	153	0	0
J30,17-2	68	68	68	68	0	0.99	77	0	0
J30,17-3	60	60	60	60	0	0.66	76	0	0
J30,17-4	49	50	50.3	54	1.53	64.98	77	0	0
J30,17-5	47	48	47.93	50	0.98	55.79	79	0	0
J30,17-6	63	63	63	63	0	0.72	76	0	0
J30,17-7	57	57	57.2	59	0.55	9.35	77	0	0
J30,17-8	61	61	61	61	0	3.22	77	0	0
J30,17-9	48	48	48.13	50	0.43	9.52	76	0	0
J30,17-10	66	66	66.3	75	1.64	3.45	76	0	0
J30,18-1	53	53	53	53	0	0.76	76	0	0
J30,18-2	55	55	55	55	0	0.67	76	0	0
J30,18-3	56	56	56.7	59	1.29	18.35	76	0	0
J30,18-4	70	70	70	70	0	0.67	76	0	0
J30,18-5	52	52	52	52	0	0.89	77	0	0
J30,18-6	62	62	63.4	65	1.52	34.75	78	0	0
J30,18-7	48	48	48	48	0	0.7	76	0	0
J30,18-8	52	52	52	52	0	0.71	76	0	0
J30,18-9	47	47	47	47	0	1.69	76	0	0
J30,18-10	49	49	49.1	50	0.31	9.85	76	0	0
J30,19-1	40	40	40	40	0	0.78	76	0	0
J30,19-2	58	58	58	58	0	0.68	76	0	0
J30,19-3	83	83	83	83	0	0.68	76	0	0
J30,19-4	39	39	39	39	0	1.24	77	0	0
J30,19-5	48	48	48	48	0	0.97	77	0	0
J30,19-6	49	49	49	49	0	0.83	77	0	0
J30,19-7	57	57	57	57	0	0.78	76	0	0
J30,19-8	55	55	55	55	0	0.7	76	0	0

J30,19-9	38	38	38	38	0	1.37	77	0	0
J30,19-10	47	47	47	47	0	0.81	77	0	0
J30,20-1	57	57	57	57	0	0.67	76	0	0
J30,20-2	70	70	70	70	0	0.67	76	0	0
J30,20-3	49	49	49	49	0	0.64	76	0	0
J30,20-4	43	43	43	43	0	0.65	76	0	0
J30,20-5	61	61	61	61	0	0.68	76	0	0
J30,20-6	51	51	51	51	0	0.66	76	0	0
J30,20-7	42	42	42	42	0	0.67	76	0	0
J30,20-8	51	51	51	51	0	0.67	76	0	0
J30,20-9	41	41	41	41	0	0.6	76	0	0
J30,20-10	37	37	37	37	0	0.61	76	0	0
J30,21-1	84	89	87.13	93	2.49	66.11	458	0	0
J30,21-2	59	61	61.03	64	1.16	79.76	230	0	0
J30,21-3	76	76	77.23	83	1.65	43.09	308	0	0
J30,21-4	70	71.5	71.5	75	1.33	72.97	81	0	0
J30,21-5	55	57	56.97	62	1.9	72.94	880	0	0
J30,21-6	76	77	77.27	80	1.31	60.96	78	0	0
J30,21-7	65	68	67.83	74	2.39	68.17	880	0	0
J30,21-8	62	64	64.6	70	1.96	89.22	2666	0	0
J30,21-9	69	69	70.43	76	2.57	40.41	382	0	0
J30,21-10	69	73	72.3	77	2.38	76.84	656	0	0
J30,22-1	42	42	42.1	44	0.4	14.07	80	0	0
J30,22-2	45	45	45	45	0	5.29	77	0	0
J30,22-3	63	63	63	63	0	0.78	76	0	0
J30,22-4	42	42	42	42	0	8.24	192	0	0
J30,22-5	52	54	53.6	54	0.81	69.39	80	0	0
J30,22-6	52	53	52.9	54	0.76	64.84	191	0	0
J30,22-7	60	60	60.37	62	0.67	29.88	155	0	0
J30,22-8	55	56	56.33	59	1.15	73.64	346	0	0
J30,22-9	76	76	76	76	0	1	77	0	0
J30,22-10	55	55	55.6	58	1.04	29.93	80	0	0
J30,23-1	63	63	63	63	0	0.75	76	0	0
J30,23-2	53	53	53	53	0	0.75	76	0	0
J30,23-3	46	46	46	46	0	1.09	76	0	0
J30,23-4	65	65	65.07	66	0.25	13.2	81	0	0
J30,23-5	52	52	52	52	0	1.95	77	0	0
J30,23-6	48	48	48.13	50	0.51	8.65	77	0	0
J30,23-7	60	60	60	60	0	1.1	77	0	0
J30,23-8	48	48	48	48	0	2.48	78	0	0
J30,23-9	63	63	63	63	0	0.79	77	0	0
J30,23-10	61	61	61	61	0	0.72	77	0	0
J30,24-1	53	53	53	53	0	0.71	76	0	0
J30,24-2	58	58	58	58	0	0.7	76	0	0
J30,24-3	69	69	69	69	0	0.7	76	0	0
J30,24-4	53	53	53	53	0	0.7	76	0	0
J30,24-5	51	51	51	51	0	0.72	76	0	0
J30,24-6	56	56	56	56	0	0.71	76	0	0
J30,24-7	44	44	44	44	0	0.71	76	0	0
J30,24-8	38	38	38	38	0	0.68	76	0	0
J30,24-9	43	43	43	43	0	0.72	76	0	0
J30,24-10	53	53	53	53	0	0.7	76	0	0
J30,25-1	93	95	94.97	98	1.22	106.38	3164	0	0

J30,25-2	75	78	77.4	81	1.35	83.02	1034	0	0
J30,25-3	76	80	80.23	85	1.96	115.85	1682	0	0
J30,25-4	82	84	84.3	89	1.99	109.46	11748	1.23	1.23
J30,25-5	72	73	72.67	74	0.66	62.43	769	0	0
J30,25-6	59	61	61.53	65	1.43	114.95	11591	1.72	1.72
J30,25-7	96	98	98.33	103	1.37	90.59	12088	1.05	1.05
J30,25-8	71	72	72.77	75	1.25	90.47	11517	2.9	2.9
J30,25-9	84	86	86.4	88	1.54	82.34	1711	0	0
J30,25-10	58	60	60.23	62	1.1	85.85	11667	0	0
J30,26-1	59	59	59.3	61	0.65	24.7	78	0	0
J30,26-2	40	40	40	40	0	0.82	77	0	0
J30,26-3	58	58	58	58	0	1.09	76	0	0
J30,26-4	62	62	62	62	0	3.36	77	0	0
J30,26-5	74	74	74	74	0	0.93	76	0	0
J30,26-6	53	55	54.63	56	0.61	79.96	8476	0	0
J30,26-7	56	57	56.8	58	0.76	61.07	647	0	0
J30,26-8	66	66	66	66	0	0.92	77	0	0
J30,26-9	44	44	44.37	47	0.67	97.39	11668	2.33	2.33
J30,26-10	49	50	50.17	53	1.29	66.18	805	0	0
J30,27-1	43	43	43.4	45	0.72	23.33	80	0	0
J30,27-2	58	58	58	58	0	0.71	76	0	0
J30,27-3	60	60	60	60	0	0.72	76	0	0
J30,27-4	64	64	64.03	65	0.18	3.08	76	0	0
J30,27-5	49	49	49.07	51	0.37	5.89	77	0	0
J30,27-6	59	59	59.03	60	0.18	4.67	77	0	0
J30,27-7	49	49	49	49	0	5.76	120	0	0
J30,27-8	66	66	66	66	0	0.85	76	0	0
J30,27-9	55	55	55	55	0	0.84	76	0	0
J30,27-10	62	62	62	62	0	0.68	76	0	0
J30,28-1	69	69	69	69	0	0.7	76	0	0
J30,28-2	57	57	57	57	0	0.64	76	0	0
J30,28-3	40	40	40	40	0	0.66	76	0	0
J30,28-4	49	49	49	49	0	0.61	76	0	0
J30,28-5	73	73	73	73	0	0.63	76	0	0
J30,28-6	55	55	55	55	0	0.68	76	0	0
J30,28-7	48	48	48	48	0	0.63	76	0	0
J30,28-8	53	53	53	53	0	0.66	76	0	0
J30,28-9	62	62	62	62	0	0.66	76	0	0
J30,28-10	59	59	59	59	0	0.63	76	0	0
J30,29-1	87	90	89.87	94	1.8	108.55	11668	2.35	2.35
J30,29-2	91	93.5	94.07	101	2.13	114.41	11860	1.11	1.11
J30,29-3	78	80	79.87	83	1.11	99.3	1678	0	0
J30,29-4	104	104	104.83	112	1.66	92.25	11820	0.97	0.97
J30,29-5	98	104	103.7	110	2.78	101.21	5930	0	0
J30,29-6	92	94	93.7	96	1.12	92.79	2100	0	0
J30,29-7	73	74	74.73	78	1.53	106.07	1679	0	0
J30,29-8	81	84	83.93	90	1.86	133.12	12237	1.25	1.25
J30,29-9	99	104	103.8	111	2.41	103.85	11594	2.06	2.06
J30,29-10	77	78	78	80	1.08	85.89	11478	1.32	1.32
J30,30-1	47	48	48.7	51	1.39	84.88	500	0	0
J30,30-2	70	70	70.5	72	0.73	99.16	11556	2.94	2.94
J30,30-3	55	56	55.8	58	0.85	60.64	495	0	0
J30,30-4	53	54	54.4	55	0.56	89.77	9009	0	0

J30,30-5	54	55	55.07	56	0.58	100.13	2248	0	0
J30,30-6	62	62	62.53	64	0.82	53.19	462	0	0
J30,30-7	68	69	69.1	71	0.92	88.58	954	0	0
J30,30-8	46	46	46.07	47	0.25	13.17	229	0	0
J30,30-9	46	46	46.7	48	0.88	56.73	462	0	0
J30,30-10	53	54	54.03	55	0.89	57.67	457	0	0
J30,31-1	43	43	43	43	0	0.73	76	0	0
J30,31-2	63	63	63	63	0	0.69	76	0	0
J30,31-3	58	58	58	58	0	0.68	76	0	0
J30,31-4	50	50	50	50	0	0.72	76	0	0
J30,31-5	52	52	52.5	55	0.82	42.35	78	0	0
J30,31-6	53	53	53	53	0	0.68	76	0	0
J30,31-7	61	61	61.4	64	0.67	36.67	77	0	0
J30,31-8	58	58	58	58	0	1.36	77	0	0
J30,31-9	50	50	50.23	51	0.43	37.17	458	0	0
J30,31-10	55	56	55.83	56	0.38	76.82	193	0	0
J30,32-1	61	61	61	61	0	0.64	76	0	0
J30,32-2	60	60	60	60	0	0.68	76	0	0
J30,32-3	57	57	57	57	0	0.69	76	0	0
J30,32-4	68	68	68	68	0	0.66	76	0	0
J30,32-5	54	54	54	54	0	0.69	76	0	0
J30,32-6	44	44	44	44	0	0.74	76	0	0
J30,32-7	35	35	35	35	0	0.66	76	0	0
J30,32-8	54	54	54	54	0	0.68	76	0	0
J30,32-9	65	65	65	65	0	0.68	76	0	0
J30,32-10	51	51	51	51	0	0.67	76	0	0
J30,33-1	65	65	65	65	0	0.73	76	0	0
J30,33-2	60	62	61.13	62	1.01	35.4	77	0	0
J30,33-3	55	55	55.3	56	0.47	20.61	76	0	0
J30,33-4	77	77	77.2	78	0.41	15.33	77	0	0
J30,33-5	53	53	53.13	55	0.51	7.26	76	0	0
J30,33-6	59	59	59	59	0	0.6	76	0	0
J30,33-7	58	58	58	58	0	0.75	77	0	0
J30,33-8	61	61	61.2	62	0.41	18.44	80	0	0
J30,33-9	65	65	65.13	67	0.51	20.4	229	0	0
J30,33-10	53	53	53	53	0	0.76	76	0	0
J30,34-1	68	68	68	68	0	0.69	77	0	0
J30,34-2	44	44	44	44	0	0.61	76	0	0
J30,34-3	69	69	69	69	0	1.28	76	0	0
J30,34-4	67	67	67	67	0	0.55	76	0	0
J30,34-5	63	63	63	63	0	0.64	76	0	0
J30,34-6	52	52	52	52	0	0.68	76	0	0
J30,34-7	58	58	58	58	0	0.8	77	0	0
J30,34-8	58	58	58	58	0	0.57	76	0	0
J30,34-9	60	60	60	60	0	0.57	76	0	0
J30,34-10	47	47	47	47	0	1.35	77	0	0
J30,35-1	57	57	57	57	0	0.6	76	0	0
J30,35-2	53	53	53	53	0	0.56	76	0	0
J30,35-3	60	60	60	60	0	0.74	76	0	0
J30,35-4	50	50	50	50	0	1.84	77	0	0
J30,35-5	60	60	60	60	0	0.81	77	0	0
J30,35-6	58	58	58.07	60	0.37	8.97	76	0	0
J30,35-7	61	61	61	61	0	0.63	76	0	0

J30,35-8	63	63	63	63	0	0.55	76	0	0
J30,35-9	59	59	59.2	62	0.76	7.97	77	0	0
J30,35-10	59	59	59	59	0	0.58	76	0	0
J30,36-1	66	66	66	66	0	0.53	76	0	0
J30,36-2	44	44	44	44	0	0.54	76	0	0
J30,36-3	61	61	61	61	0	0.57	76	0	0
J30,36-4	59	59	59	59	0	0.55	76	0	0
J30,36-5	64	64	64	64	0	0.54	76	0	0
J30,36-6	46	46	46	46	0	0.57	76	0	0
J30,36-7	56	56	56	56	0	0.54	76	0	0
J30,36-8	63	63	63	63	0	0.53	76	0	0
J30,36-9	59	59	59	59	0	0.56	76	0	0
J30,36-10	59	59	59	59	0	0.54	76	0	0
J30,37-1	80	80	80.6	82	0.77	77.28	11477	1.27	1.27
J30,37-2	69	69	69	69	0	1.34	76	0	0
J30,37-3	81	85	83.97	87	2.41	58.73	844	0	0
J30,37-4	83	83	83.3	86	0.92	15.22	115	0	0
J30,37-5	80	80	80.27	84	0.87	17.65	501	0	0
J30,37-6	73	76	75.53	76	0.68	67.11	837	0	0
J30,37-7	92	92	92.37	96	1.13	14.74	162	0	0
J30,37-8	72	72	72.27	80	1.46	10.33	272	0	0
J30,37-9	57	58	58.03	63	1.07	56.13	240	0	0
J30,37-10	82	83	82.53	83	0.51	68.88	11477	1.23	1.23
J30,38-1	48	48	48.2	49	0.41	20.28	153	0	0
J30,38-2	54	54	54	54	0	7.46	79	0	0
J30,38-3	59	60	59.77	61	0.57	55.1	194	0	0
J30,38-4	59	59	59	59	0	1.44	77	0	0
J30,38-5	71	71	71.17	72	0.38	34.28	272	0	0
J30,38-6	63	63	63.1	66	0.55	12.68	270	0	0
J30,38-7	65	66	66.07	68	1.11	54.53	153	0	0
J30,38-8	61	61	61.07	63	0.37	5.74	78	0	0
J30,38-9	63	63	63	63	0	2.42	77	0	0
J30,38-10	60	60	60	60	0	1.18	76	0	0
J30,39-1	55	55	55	55	0	0.94	76	0	0
J30,39-2	54	54	54	54	0	0.69	76	0	0
J30,39-3	54	54	54	54	0	2.69	76	0	0
J30,39-4	53	53	53.17	54	0.38	13.5	77	0	0
J30,39-5	55	55	55.53	57	0.9	27.14	77	0	0
J30,39-6	69	69	69	69	0	1.55	76	0	0
J30,39-7	56	56	56	56	0	0.7	77	0	0
J30,39-8	67	67	67	67	0	0.62	76	0	0
J30,39-9	64	64	64.03	65	0.18	12.19	120	0	0
J30,39-10	60	60	60	60	0	0.67	76	0	0
J30,40-1	51	51	51	51	0	2.11	203	0	0
J30,40-2	56	56	56	56	0	2.03	203	0	0
J30,40-3	57	57	57	57	0	2.04	203	0	0
J30,40-4	57	57	57	57	0	1.99	203	0	0
J30,40-5	65	65	65	65	0	2.05	203	0	0
J30,40-6	60	60	60	60	0	2.05	203	0	0
J30,40-7	46	46	46	46	0	1.92	203	0	0
J30,40-8	57	57	57	57	0	2.04	203	0	0
J30,40-9	64	64	64	64	0	2.11	203	0	0
J30,40-10	51	51	51	51	0	2.13	203	0	0

J30,41-1	86	88	88.33	91	1.37	153.62	1674	0	0
J30,41-2	89	89	89.83	93	1.18	71.33	658	0	0
J30,41-3	85	85	85.33	86	0.48	73.28	555	0	0
J30,41-4	78	78	79.13	82	1.46	100.91	1044	0	0
J30,41-5	99	99	99.7	103	1.12	85.53	542	0	0
J30,41-6	103	107	106.13	108	1.76	119.62	420	0	0
J30,41-7	92	96.5	95.7	97	1.82	140.75	783	0	0
J30,41-8	88	90	89.93	94	1.39	164.71	2032	0	0
J30,41-9	92	95	95.67	102	2.72	163.3	6150	0	0
J30,41-10	99	100	100.07	105	0.98	139.44	991	0	0
J30,42-1	58	58	58.13	61	0.57	13.56	111	0	0
J30,42-2	50	50	50.33	51	0.48	65.63	577	0	0
J30,42-3	60	61.5	61.27	63	0.98	127.69	1880	0	0
J30,42-4	49	50	49.7	51	0.65	105.43	888	0	0
J30,42-5	52	52	52	52	0	2.18	104	0	0
J30,42-6	67	67	67	67	0	149.28	15712	1.52	1.52
J30,42-7	66	66	66	66	0	2.55	106	0	0
J30,42-8	82	82	82	82	0	5.8	106	0	0
J30,42-9	60	60	60.9	64	1.24	91.25	679	0	0
J30,42-10	75	75	75	75	0	1.9	105	0	0
J30,43-1	55	56	55.63	56	0.49	107.09	890	0	0
J30,43-2	43	43	43	43	0	8.02	108	0	0
J30,43-3	57	58	57.93	61	1.11	135.61	526	0	0
J30,43-4	67	67	67	67	0	2.83	105	0	0
J30,43-5	64	65	64.8	66	0.66	135.03	687	0	0
J30,43-6	58	58	58	58	0	9.3	160	0	0
J30,43-7	52	52	52	52	0	1.99	106	0	0
J30,43-8	62	63	62.83	65	0.75	151.47	3754	0	0
J30,43-9	57	57	57.33	58	0.48	69.89	109	0	0
J30,43-10	60	60	60	60	0	5.41	105	0	0
J30,44-1	50	50	50	50	0	1.28	104	0	0
J30,44-2	54	54	54	54	0	1.19	104	0	0
J30,44-3	51	51	51	51	0	1.09	104	0	0
J30,44-4	57	57	57	57	0	1.25	104	0	0
J30,44-5	55	55	55	55	0	1.15	104	0	0
J30,44-6	56	56	56	56	0	1.14	104	0	0
J30,44-7	42	42	42	42	0	1.16	104	0	0
J30,44-8	49	49	49	49	0	1.23	104	0	0
J30,44-9	64	64	64	64	0	1.18	104	0	0
J30,44-10	63	63	63	63	0	1.16	104	0	0
J30,45-1	82	84	83.8	87	1.24	157.76	1203	0	0
J30,45-2	125	125	125.43	128	1.04	32.39	159	0	0
J30,45-3	92	96	95.67	98	2.15	142.21	1043	0	0
J30,45-4	84	85	85.07	91	1.36	156.59	1774	0	0
J30,45-5	87	90	89.17	91	1.18	154.32	14791	1.16	1.16
J30,45-6	129	129	129.7	134	1.32	82.8	1060	0	0
J30,45-7	101	102	103.6	108	2.8	116.83	1828	0	0
J30,45-8	94	96	95.6	97	0.89	117.42	964	0	0
J30,45-9	82	87	86.5	90	2.35	192.78	12819	0	0
J30,45-10	90	91	91.5	95	1.33	131.89	776	0	0
J30,46-1	59	60	60.5	63	1.2	150.08	4372	0	0
J30,46-2	67	67	67.2	68	0.41	48.96	248	0	0
J30,46-3	65	66	66.5	68	0.86	153.95	914	0	0

J30,46-4	64	65	64.8	66	0.76	116.01	628	0	0
J30,46-5	57	57	57.83	59	0.95	101.26	1022	0	0
J30,46-6	59	60	59.8	60	0.41	137.94	1155	0	0
J30,46-7	59	60	59.7	61	0.6	123.5	342	0	0
J30,46-8	58	59	59.3	60	0.53	170.52	1880	0	0
J30,46-9	49	49	49.5	51	0.57	106.42	481	0	0
J30,46-10	55	55	55.1	56	0.31	30.66	148	0	0
J30,47-1	58	58	58	58	0	4	97	0	0
J30,47-2	59	59	59	59	0	1.76	96	0	0
J30,47-3	55	55	55	55	0	2.35	97	0	0
J30,47-4	49	49	49.43	50	0.5	78.52	626	0	0
J30,47-5	47	47	47	47	0	4.73	99	0	0
J30,47-6	53	53	53.57	55	0.68	95.93	821	0	0
J30,47-7	66	66	66	66	0	2.61	96	0	0
J30,47-8	48	48	48	48	0	1.25	96	0	0
J30,47-9	65	65	65	65	0	1.27	96	0	0
J30,47-10	60	60	60	60	0	12.31	198	0	0
J30,48-1	63	63	63	63	0	1.09	96	0	0
J30,48-2	54	54	54	54	0	1.16	96	0	0
J30,48-3	50	50	50	50	0	1.05	96	0	0
J30,48-4	57	57	57	57	0	1.14	96	0	0
J30,48-5	58	58	58	58	0	1.17	96	0	0
J30,48-6	58	58	58	58	0	1.15	96	0	0
J30,48-7	55	55	55	55	0	1.15	96	0	0
J30,48-8	44	44	44	44	0	1.14	96	0	0
J30,48-9	59	59	59	59	0	1.17	96	0	0
J30,48-10	54	54	54	54	0	1.13	96	0	0
Total Average					**0.47**	**32.37**	**1188.58**	**0.10**	**0.10**

Table 14

The detailed results of 480 problems of J60 obtained by bi-EA

The results are obtained from 30 runs with up to 50,000 fitness evaluations for each.

Prob. No	Best	Median	Mean	Worst	STD	t	FE	LB_{OP}	LB_{CP}
J60,1-1	77	77	77	77	0	4.39	201	0	4.62
J60,1-2	68	69	69.87	73	2.03	227.95	784	0	1.49
J60,1-3	68	71.5	71.27	76	1.7	319.85	6146	0	15.19
J60,1-4	91	91	91.07	93	0.37	14.5	111	0	8.82
J60,1-5	74	75	75.53	79	1.17	288.77	16611	1.37	26.92
J60,1-6	66	66	66.07	67	0.25	27.8	111	0	23.33
J60,1-7	74	76	76.33	82	1.73	334.28	16725	2.78	5.63
J60,1-8	75	78	78.1	82	1.69	307.4	5735	0	13.33
J60,1-9	85	85	85.6	92	1.59	55.38	110	0	5.26
J60,1-10	80	80	80	80	0	5.34	107	0	0
J60,2-1	65	65	65	65	0	2.48	106	0	0
J60,2-2	82	82	82	82	0	2.11	106	0	1.3
J60,2-3	78	78	78	78	0	2.44	106	0	0
J60,2-4	78	78	78	78	0	3.24	106	0	1.89
J60,2-5	54	54	54.03	55	0.18	31.03	107	0	4.92
J60,2-6	64	64	64	64	0	6.8	111	0	8.16
J60,2-7	53	53	53	53	0	4.55	108	0	0
J60,2-8	66	66	66	66	0	2.29	106	0	0
J60,2-9	65	65	65.53	69	1.22	62.66	108	0	7.81
J60,2-10	69	69	69.07	70	0.25	21.61	106	0	0
J60,3-1	60	60	60	60	0	3.95	106	0	0
J60,3-2	69	69	69	69	0	2.1	106	0	2.94
J60,3-3	105	105	105	105	0	2.11	107	0	0
J60,3-4	81	81	81	81	0	2.09	106	0	0
J60,3-5	83	83	83	83	0	2	106	0	0
J60,3-6	57	57	57.07	58	0.25	32.96	107	0	1.72
J60,3-7	59	59	59	59	0	2.2	106	0	5.77
J60,3-8	55	55	55.33	57	0.55	106.73	108	0	6.35
J60,3-9	67	67	67	67	0	6.33	108	0	2.99
J60,3-10	69	69	69.3	70	0.47	83.66	108	0	0
J60,4-1	84	84	84	84	0	1.99	106	0	0
J60,4-2	60	60	60	60	0	2.05	106	0	0
J60,4-3	58	58	58	58	0	2.04	106	0	0
J60,4-4	65	65	65	65	0	2.08	106	0	0
J60,4-5	75	75	75	75	0	2.11	106	0	0
J60,4-6	71	71	71	71	0	2.03	106	0	0
J60,4-7	67	67	67	67	0	2.01	106	0	0
J60,4-8	65	65	65	65	0	2.05	106	0	0
J60,4-9	75	75	75	75	0	2.04	106	0	0
J60,4-10	77	77	77	77	0	2.05	106	0	33.9
J60,5-1	79	84	84.17	89	2.32	368.68	14819	3.95	43.42

J60,5-2	109	113	113.97	123	3.02	371.21	14491	2.83	42.37
J60,5-3	84	87	88	101	3.44	324.81	11133	5	41.51
J60,5-4	75	80	80.13	86	2.6	264.06	10607	4.17	40.51
J60,5-5	111	118	117.87	124	3.51	252.73	10644	2.78	23.44
J60,5-6	79	84	84.6	91	3.29	265.04	10646	6.76	60.78
J60,5-7	82	84.5	84.43	89	1.87	262.71	11132	9.33	26.15
J60,5-8	82	87	86.5	91	2.43	251.78	10891	5.13	7.32
J60,5-9	88	92	92.57	97	2.06	180.26	10572	6.02	25
J60,5-10	85	89	88.87	91	1.59	220.61	11416	4.94	0
J60,6-1	60	62	62.03	64	1.13	179.45	8933	0	4.55
J60,6-2	69	71	70.43	72	0.97	165.46	10572	2.99	0
J60,6-3	72	72	72.57	77	1.19	54.01	71	0	0
J60,6-4	67	69	69.2	72	1.61	151.75	879	0	0
J60,6-5	78	78	78	78	0	10.28	70	0	5.66
J60,6-6	56	60	59.3	62	1.53	170.24	10748	1.82	3.33
J60,6-7	62	63	63.6	66	0.89	155.14	10642	1.64	0
J60,6-8	72	74	73.67	76	1.49	99.27	73	0	0
J60,6-9	64	65	65.07	69	1.36	118.96	109	0	0
J60,6-10	74	74	74.03	75	0.18	7.82	70	0	0
J60,7-1	77	77	77	77	0	1.17	70	0	0
J60,7-2	85	85	85	85	0	1.19	70	0	0
J60,7-3	62	62	62	62	0	1.19	70	0	0
J60,7-4	63	63	63	63	0	5.87	70	0	0
J60,7-5	71	71	71	71	0	1.21	70	0	0
J60,7-6	65	65	65	65	0	3.46	70	0	0
J60,7-7	89	89	89	89	0	1.14	70	0	0
J60,7-8	66	66	66	66	0	1.13	70	0	0
J60,7-9	44	44	44.4	46	0.62	59.04	72	0	0
J60,7-10	82	82	82	82	0	1.07	70	0	0
J60,8-1	64	64	64	64	0	1.04	70	0	0
J60,8-2	61	61	61	61	0	1.07	70	0	0
J60,8-3	79	79	79	79	0	1.1	70	0	0
J60,8-4	64	64	64	64	0	1.07	70	0	0
J60,8-5	83	83	83	83	0	1.08	70	0	0
J60,8-6	56	56	56	56	0	1.04	70	0	0
J60,8-7	62	62	62	62	0	1.04	70	0	0
J60,8-8	66	66	66	66	0	1.09	70	0	0
J60,8-9	58	58	58	58	0	1.07	70	0	0
J60,8-10	97	97	97	97	0	1.06	70	0	57.63
J60,9-1	93	95.5	95.57	99	1.7	232.47	11097	6.9	23.94
J60,9-2	88	92.5	92.47	97	2.19	210.95	11066	7.32	66.15
J60,9-3	108	112	112.47	121	2.61	228.58	12990	8	41.54
J60,9-4	92	97.5	98.23	106	3.43	221.57	11099	5.75	76.92
J60,9-5	92	96	95.83	100	2.21	271.8	12149	8.24	30
J60,9-6	117	122	121.47	125	2.15	258.95	11976	5.41	58.11
J60,9-7	117	122	122.07	126	2.07	273.57	11206	7.34	56.92
J60,9-8	102	108	107.27	114	2.7	221.35	10994	6.25	33.75
J60,9-9	107	111	110.73	114	2.07	206.64	11131	8.08	55.38
J60,9-10	101	106	105.7	109	2.48	236.48	11308	8.6	0
J60,10-1	85	85	85.03	86	0.18	7.19	70	0	0
J60,10-2	62	65	64.87	68	1.55	156.32	841	0	0
J60,10-3	72	72	72.6	75	0.77	90.76	422	0	0
J60,10-4	80	80	80	80	0	2.22	70	0	0

J60,10-5	79	79	79.17	83	0.75	18.45	72	0	0
J60,10-6	67	69	69.13	74	1.81	135.45	71	0	1.45
J60,10-7	70	72	72.53	75	1.46	182.63	10853	1.45	1.56
J60,10-8	65	68.5	68.93	72	2.07	162.49	5887	0	4.11
J60,10-9	76	78	78.27	81	1.6	169.81	10572	4.11	0
J60,10-10	73	73	73	73	0	1.7	70	0	0
J60,11-1	71	71	71	71	0	1.09	70	0	0
J60,11-2	61	61	61	61	0	1.04	70	0	0
J60,11-3	76	76	76	76	0	1.01	70	0	0
J60,11-4	69	69	69.07	70	0.25	24.95	71	0	0
J60,11-5	65	65	65	65	0	1.15	70	0	0
J60,11-6	70	70	70.07	72	0.37	16.6	70	0	0
J60,11-7	70	70	70	70	0	1.16	70	0	0
J60,11-8	69	69	69	69	0	1.12	70	0	0
J60,11-9	62	62	62	62	0	1.71	70	0	0
J60,11-10	58	58	58	58	0	1.31	70	0	0
J60,12-1	59	59	59	59	0	1.08	70	0	0
J60,12-2	58	58	58	58	0	1.19	70	0	0
J60,12-3	75	75	75	75	0	1.15	70	0	0
J60,12-4	69	69	69	69	0	1.16	70	0	0
J60,12-5	63	63	63	63	0	1.14	70	0	0
J60,12-6	54	54	54	54	0	1.11	70	0	0
J60,12-7	71	71	71	71	0	1.09	70	0	0
J60,12-8	60	60	60	60	0	1.09	70	0	0
J60,12-9	59	59	59	59	0	1.1	70	0	0
J60,12-10	79	79	79	79	0	1.08	70	0	73.91
J60,13-1	120	126	126.1	130	2.48	382.05	12310	7.14	71.21
J60,13-2	113	117	117.87	123	2.62	337.15	10235	6.6	61.4
J60,13-3	92	97	96.97	102	2.53	366.16	10630	4.55	77.42
J60,13-4	110	113	113.27	118	1.66	331.21	10402	6.8	98.11
J60,13-5	105	109	108.97	114	2.47	385.44	10200	8.25	63.93
J60,13-6	100	103	103.1	107	1.65	379.29	10761	6.38	72.22
J60,13-7	93	97	97.13	101	1.76	430.21	10562	6.9	86.96
J60,13-8	129	133	132.47	136	1.87	376.82	11817	7.5	59.7
J60,13-9	107	111	111.27	116	2.16	349.95	10695	4.9	98.41
J60,13-10	125	130	129.7	134	2.44	381.93	10600	6.84	6.78
J60,14-1	63	64	64.03	67	1.03	315.23	10203	3.28	0
J60,14-2	65	65	65.77	68	1.14	145.09	597	0	3.28
J60,14-3	63	65	64.87	66	0.86	274.76	10198	3.28	3.08
J60,14-4	67	69	68.73	70	0.87	294.76	10267	3.08	0
J60,14-5	59	61	60.83	63	1.05	250.53	728	0	0
J60,14-6	65	65	65.33	67	0.61	106.76	434	0	0
J60,14-7	69	70	69.9	73	1.03	196.91	993	0	0
J60,14-8	88	88	88	88	0	1.78	66	0	0
J60,14-9	61	62	62.87	66	1.38	258.46	1626	0	11.94
J60,14-10	75	77	76.87	80	1.25	292.45	10336	4.17	0
J60,15-1	84	84	84	84	0	1.64	66	0	0
J60,15-2	89	89	89	89	0	1.75	66	0	0
J60,15-3	72	72	72	72	0	1.61	66	0	0
J60,15-4	75	75	75	75	0	1.66	66	0	0
J60,15-5	70	70	70	70	0	1.74	66	0	0
J60,15-6	76	76	76	76	0	1.8	66	0	0
J60,15-7	64	64	64	64	0	2.64	66	0	0

J60,15-8	79	79	79	79	0	1.66	66	0	0
J60,15-9	72	72	72	72	0	1.53	66	0	0
J60,15-10	61	61	61	61	0	1.93	66	0	0
J60,16-1	64	64	64	64	0	1.54	66	0	0
J60,16-2	64	64	64	64	0	1.48	66	0	0
J60,16-3	53	53	53	53	0	1.45	66	0	0
J60,16-4	60	60	60	60	0	1.55	66	0	0
J60,16-5	66	66	66	66	0	1.53	66	0	0
J60,16-6	66	66	66	66	0	1.56	66	0	0
J60,16-7	82	82	82	82	0	1.6	66	0	0
J60,16-8	68	68	68	68	0	1.5	66	0	0
J60,16-9	54	54	54	54	0	1.53	66	0	0
J60,16-10	68	68	68	68	0	1.58	66	0	13.16
J60,17-1	86	87.5	89.17	96	3.53	140.19	663	0	4.48
J60,17-2	70	73	73.5	78	2.01	193.28	9969	1.45	15.19
J60,17-3	91	92	93.1	99	2.01	175.95	9969	2.25	2.9
J60,17-4	71	71	71.27	78	1.28	42.48	67	0	15.38
J60,17-5	60	61	61.53	64	1.41	194.51	9973	1.69	2.99
J60,17-6	69	70	72	79	3.24	149.17	436	0	2.47
J60,17-7	83	86	85.7	92	2.2	161.97	696	0	28.79
J60,17-8	85	89	90.03	96	3.12	202.66	5750	0	8.57
J60,17-9	76	81	80.9	86	3.22	175.98	206	0	10.77
J60,17-10	72	73.5	74.6	79	2.57	152.72	464	0	1.25
J60,18-1	81	81	81	81	0	2.94	66	0	0
J60,18-2	69	69	69	69	0	1.69	66	0	0
J60,18-3	77	77	77	77	0	1.42	66	0	0
J60,18-4	71	71	71.47	78	1.48	39.94	70	0	0
J60,18-5	80	80	80	80	0	1.38	66	0	0
J60,18-6	61	61	61.17	65	0.75	14.97	67	0	6.9
J60,18-7	93	95	94.33	97	1.06	116.86	67	0	0
J60,18-8	78	78	78	78	0	1.47	66	0	0
J60,18-9	69	69	71.57	75	2.99	85.45	66	0	0
J60,18-10	97	97	97	97	0	1.37	66	0	0
J60,19-1	62	62	62.03	63	0.18	11.52	66	0	0
J60,19-2	83	83	83	83	0	1.42	66	0	2.47
J60,19-3	83	83	83	83	0	1.46	66	0	0
J60,19-4	67	67	67	67	0	1.34	66	0	0
J60,19-5	73	73	73	73	0	1.41	66	0	1.47
J60,19-6	69	69	69	69	0	3.54	66	0	0
J60,19-7	60	60	60	60	0	1.35	66	0	6.1
J60,19-8	87	87	87	87	0	1.32	66	0	2.99
J60,19-9	69	69	69.4	73	0.97	43.32	70	0	0
J60,19-10	78	78	78	78	0	1.39	66	0	0
J60,20-1	60	60	60	60	0	1.35	66	0	0
J60,20-2	78	78	78	78	0	1.34	66	0	0
J60,20-3	69	69	69	69	0	1.32	66	0	0
J60,20-4	86	86	86	86	0	1.35	66	0	0
J60,20-5	71	71	71	71	0	1.33	66	0	0
J60,20-6	97	97	97	97	0	1.31	66	0	0
J60,20-7	74	74	74	74	0	1.31	66	0	0
J60,20-8	65	65	65	65	0	1.38	66	0	0
J60,20-9	74	74	74	74	0	1.35	66	0	0
J60,20-10	70	70	70	70	0	1.34	66	0	36.84

J60,21-1	104	109	109.37	113	2.24	255.02	10132	0.97	25
J60,21-2	115	119.5	120.7	132	4.2	261.9	9968	6.48	44.62
J60,21-3	94	97.5	98.67	107	3.68	264.88	10004	8.05	53.13
J60,21-4	98	104	104.53	112	3.42	260.86	10066	3.16	21.33
J60,21-5	91	95	95.27	103	2.35	252.16	10528	2.25	33.33
J60,21-6	84	92	92.1	102	3.49	240.1	5485	0	31.25
J60,21-7	105	108	108.6	116	2.77	245.16	10365	1.94	32.94
J60,21-8	113	121	120.73	129	3.84	202.69	10233	2.73	34.78
J60,21-9	93	97	96.9	103	2.41	259.55	10266	4.49	60.78
J60,21-10	82	86.5	88.07	100	4.25	241.36	10001	2.5	3.23
J60,22-1	64	65	64.8	67	0.85	268.06	690	0	0
J60,22-2	83	84	84.47	87	1.46	268.29	400	0	0
J60,22-3	70	71	71.37	73	0.93	291.84	186	0	15.63
J60,22-4	74	77	76.73	80	1.55	407.53	14943	1.37	0
J60,22-5	76	76	76	76	0	6.8	108	0	0
J60,22-6	79	79	79.5	83	0.82	152.72	112	0	0
J60,22-7	69	70	70.13	73	1.14	250.52	113	0	0
J60,22-8	59	59	59.8	62	1.03	249.51	1047	0	0
J60,22-9	65	65	65.43	68	0.73	124.72	110	0	1.45
J60,22-10	70	72	72.23	75	1.38	367.36	475	0	0
J60,23-1	75	75	75	75	0	2.5	108	0	0
J60,23-2	69	69	69	69	0	2.66	108	0	4
J60,23-3	78	78	78	78	0	2.47	108	0	0
J60,23-4	83	83	83	83	0	2.46	108	0	0
J60,23-5	72	72	72	72	0	2.5	108	0	0
J60,23-6	81	81	81	81	0	2.49	108	0	0
J60,23-7	60	60	60.1	61	0.31	37.04	108	0	0
J60,23-8	72	72	72	72	0	2.49	108	0	0
J60,23-9	64	64	64	64	0	2.67	108	0	0
J60,23-10	68	68	68	68	0	3.3	108	0	0
J60,24-1	65	65	65	65	0	2.46	108	0	0
J60,24-2	55	55	55	55	0	2.44	108	0	0
J60,24-3	67	67	67	67	0	2.44	108	0	0
J60,24-4	78	78	78	78	0	2.43	108	0	0
J60,24-5	76	76	76	76	0	2.38	108	0	0
J60,24-6	75	75	75	75	0	2.39	108	0	0
J60,24-7	68	68	68	68	0	2.47	108	0	0
J60,24-8	81	81	81	81	0	2.39	108	0	0
J60,24-9	80	80	80	80	0	2.5	108	0	0
J60,24-10	66	66	66	66	0	2.45	108	0	70.42
J60,25-1	121	126	125.9	134	3.03	506.83	18146	6.14	53.62
J60,25-2	106	110	110.27	115	2.42	521.04	15450	8.16	34.44
J60,25-3	121	126	126.2	133	2.51	481.49	15662	7.08	46.15
J60,25-4	114	119	119.13	124	2.21	522.6	15950	5.56	68.85
J60,25-5	103	106	105.6	109	1.73	457.45	16168	5.1	56
J60,25-6	117	124	123.47	130	2.47	553.27	16526	4.46	43.94
J60,25-7	95	99	98.63	102	1.67	494.67	17482	5.56	82.46
J60,25-8	104	110	110.8	117	3.09	527.96	13636	5.05	52.17
J60,25-9	105	108.5	108.27	111	1.64	473.13	14458	6.06	48.05
J60,25-10	114	119	118.33	124	2.37	467.5	14049	5.56	0
J60,26-1	80	80	80.23	82	0.57	94.56	104	0	6.35
J60,26-2	67	69	69.3	72	1.24	396.13	14519	1.52	8.45
J60,26-3	77	80	79.83	82	1.09	370.52	13637	1.32	4.62

J60,26-4	68	69	69.3	73	1.18	410.59	14248	1.49	0
J60,26-5	61	61	61.6	64	0.97	157.43	102	0	4.11
J60,26-6	76	78	77.93	81	1.14	418.19	13648	2.7	0
J60,26-7	72	72	72	72	0	2.62	102	0	0
J60,26-8	89	89	89	89	0	22.99	102	0	6.45
J60,26-9	66	68	68.43	72	1.43	383.56	13845	1.54	0
J60,26-10	85	85	85.03	86	0.18	46.01	103	0	0
J60,27-1	96	96	96	96	0	2.53	102	0	0
J60,27-2	74	74	74	74	0	8.47	102	0	0
J60,27-3	76	76	76	76	0	5.04	102	0	0
J60,27-4	60	60	60	60	0	32.01	102	0	0
J60,27-5	78	78	78	78	0	2.51	102	0	0
J60,27-6	64	64	64	64	0	2.65	102	0	0
J60,27-7	83	83	83	83	0	2.47	102	0	0
J60,27-8	88	88	88	88	0	2.56	102	0	0
J60,27-9	76	76	76	76	0	2.48	102	0	0
J60,27-10	57	57	57	57	0	2.77	102	0	0
J60,28-1	92	92	92	92	0	2.38	102	0	0
J60,28-2	64	64	64	64	0	2.39	102	0	0
J60,28-3	72	72	72	72	0	2.47	102	0	0
J60,28-4	84	84	84	84	0	2.49	102	0	0
J60,28-5	71	71	71	71	0	2.35	102	0	0
J60,28-6	89	89	89	89	0	2.46	102	0	0
J60,28-7	75	75	75	75	0	2.35	102	0	0
J60,28-8	62	62	62	62	0	2.37	102	0	0
J60,28-9	74	74	74	74	0	2.4	102	0	0
J60,28-10	74	74	74	74	0	2.35	102	0	88.14
J60,29-1	111	115	114.53	119	2.37	262.19	15213	7.77	60.23
J60,29-2	141	145	145.07	150	2.15	235.37	13114	6.02	80.56
J60,29-3	130	134.5	134.3	138	1.91	260.5	12994	7.44	90.67
J60,29-4	143	147.5	147.9	153	2.73	243.25	13231	6.72	54.55
J60,29-5	119	123	123.53	129	2.29	240.84	15064	8.18	114.29
J60,29-6	165	170	170.03	175	2.59	253.97	14075	7.14	79.45
J60,29-7	131	134	134.3	140	2.38	211.89	14013	6.5	50
J60,29-8	108	110.5	110.7	114	1.51	224.39	13294	4.85	68.57
J60,29-9	118	124	123.9	129	2.98	226.55	12939	5.36	81.43
J60,29-10	127	132	131.83	137	2.42	260.81	15513	6.72	2.86
J60,30-1	72	73	73.53	78	1.59	209.07	12213	2.86	10.77
J60,30-2	72	75	74.73	77	1.2	201.08	13712	2.86	10
J60,30-3	88	89	89.23	92	1.43	207.25	12878	7.32	0
J60,30-4	76	76	76	76	0	1.39	90	0	8.33
J60,30-5	78	80	79.63	81	0.89	186.08	12632	2.63	0
J60,30-6	68	68	68.6	71	0.86	80.96	514	0	14.29
J60,30-7	88	90	89.6	92	1.04	201.59	14253	2.33	1.59
J60,30-8	64	66	65.93	68	1.11	162.85	12153	1.59	0
J60,30-9	98	98	98.43	100	0.63	89.27	153	0	11.11
J60,30-10	90	92	92.4	96	1.33	204.4	12934	4.65	0
J60,31-1	65	65	65	65	0	1.15	90	0	0
J60,31-2	74	74	74.33	76	0.55	58.41	571	0	0
J60,31-3	66	66	66	66	0	1.3	90	0	0
J60,31-4	68	68	68	68	0	1.3	90	0	0
J60,31-5	72	72	72	72	0	1.11	90	0	0
J60,31-6	72	72	72	72	0	1.1	90	0	0

J60,31-7	76	76	76	76	0	1.26	90	0	0
J60,31-8	75	75	75.07	76	0.25	29.12	92	0	0
J60,31-9	86	86	86	86	0	1.13	90	0	0
J60,31-10	56	56	56.27	57	0.45	63.9	273	0	0
J60,32-1	69	69	69	69	0	1.06	90	0	0
J60,32-2	114	114	114	114	0	1.09	90	0	0
J60,32-3	85	85	85	85	0	1.04	90	0	0
J60,32-4	56	56	56	56	0	1.07	90	0	0
J60,32-5	77	77	77	77	0	1.1	90	0	0
J60,32-6	93	93	93	93	0	1.12	90	0	0
J60,32-7	76	76	76	76	0	1.09	90	0	0
J60,32-8	76	76	76	76	0	1.04	90	0	0
J60,32-9	74	74	74	74	0	1.1	90	0	0
J60,32-10	77	77	77	77	0	1.13	90	0	16.67
J60,33-1	105	105	105.9	109	1.67	41.64	213	0	0
J60,33-2	100	100	100.03	101	0.18	14.28	90	0	2.6
J60,33-3	79	80	79.83	82	0.83	90.37	214	0	8
J60,33-4	81	81	81.23	84	0.73	18.12	91	0	11.34
J60,33-5	108	114	112.27	114	2.29	102.36	518	0	15.38
J60,33-6	75	78.5	77.97	80	1.33	117.55	4440	0	18.18
J60,33-7	78	79	79.4	82	1.45	101.31	274	0	9.46
J60,33-8	81	83	83.03	87	1.47	131.01	12030	2.53	5.88
J60,33-9	108	108	108	108	0	5.82	91	0	21.74
J60,33-10	84	85	85.33	87	0.84	108.09	152	0	7.46
J60,34-1	72	72	72	72	0	2.72	91	0	4.62
J60,34-2	68	68	68.63	72	1.38	58.41	91	0	0
J60,34-3	61	62	62.2	65	0.85	119.56	573	0	0
J60,34-4	83	83	83	83	0	1	90	0	0
J60,34-5	80	80	80	80	0	1.03	90	0	5.19
J60,34-6	81	81	81	81	0	7.98	92	0	2.41
J60,34-7	85	85	85.13	86	0.35	16.37	90	0	3.28
J60,34-8	63	63	63.03	64	0.18	16.31	212	0	0
J60,34-9	77	77	77	77	0	1.03	90	0	0
J60,34-10	92	92	92.03	93	0.18	4.67	90	0	0
J60,35-1	78	78	78	78	0	6.14	90	0	0
J60,35-2	77	77	77	77	0	2.67	91	0	2.3
J60,35-3	89	89	89	89	0	2.43	90	0	0
J60,35-4	72	72	72	72	0	1.01	90	0	1.33
J60,35-5	76	76	76	76	0	1.71	90	0	0
J60,35-6	79	79	79	79	0	1.51	91	0	0
J60,35-7	73	73	73	73	0	2.73	90	0	4
J60,35-8	78	78	78	78	0	0.99	90	0	4.11
J60,35-9	76	76	76	76	0	1.36	90	0	1.43
J60,35-10	71	71	71	71	0	1.24	90	0	0
J60,36-1	61	61	61	61	0	0.99	90	0	0
J60,36-2	75	75	75	75	0	1	90	0	0
J60,36-3	81	81	81	81	0	1.02	90	0	0
J60,36-4	85	85	85	85	0	0.98	90	0	0
J60,36-5	57	57	57	57	0	1.03	90	0	0
J60,36-6	76	76	76	76	0	1	90	0	0
J60,36-7	71	71	71	71	0	1.01	90	0	0
J60,36-8	69	69	69	69	0	1	90	0	0
J60,36-9	86	86	86	86	0	1	90	0	0

J60,36-10	77	77	77	77	0	1.01	90	0	41.43
J60,37-1	99	102	102.77	107	1.91	180.49	13651	2.06	44.12
J60,37-2	98	102	102.67	108	2.12	175.68	12573	3.16	46.39
J60,37-3	142	153	152.7	160	4.59	178.74	13651	2.16	36
J60,37-4	102	107	107.33	113	2.63	199.54	12339	0.99	26.25
J60,37-5	101	107	106.8	111	2.04	166.9	13056	3.06	67.74
J60,37-6	104	109	108.9	114	2.72	184	13112	1.96	49.33
J60,37-7	112	118.5	118.57	126	3.65	180.32	12931	1.82	22.78
J60,37-8	97	100	100.1	104	2.06	188.41	13658	4.3	27.27
J60,37-9	98	102.5	102.73	108	2.33	182.99	12691	2.08	12.79
J60,37-10	97	104	103.3	109	2.77	171.73	12273	1.04	0
J60,38-1	73	73	73.37	77	0.85	62.91	91	0	9.86
J60,38-2	78	80	79.9	83	1.35	179.18	12273	2.63	1.32
J60,38-3	77	79	79	82	1.23	124.17	1355	0	0
J60,38-4	58	60	59.8	63	1.37	136.09	576	0	0
J60,38-5	103	103	103	103	0	1.9	90	0	0
J60,38-6	86	86	86.13	88	0.51	29.69	90	0	1.35
J60,38-7	75	77	77.07	79	1.23	151.5	12031	1.35	5.88
J60,38-8	72	73.5	73.7	76	1.32	170.08	12331	1.41	0
J60,38-9	66	66	66.7	69	0.95	98.61	1234	0	6.35
J60,38-10	67	68	67.9	70	0.96	169.54	12154	1.52	0
J60,39-1	80	80	80	80	0	1.08	90	0	0
J60,39-2	84	84	84	84	0	1.15	90	0	0
J60,39-3	83	83	83	83	0	1.11	90	0	0
J60,39-4	92	92	92	92	0	10.09	91	0	0
J60,39-5	73	73	73	73	0	10.14	90	0	1.2
J60,39-6	84	84	84.07	85	0.25	17.47	92	0	0
J60,39-7	68	68	68	68	0	3.56	91	0	0
J60,39-8	77	77	77	77	0	1.08	90	0	0
J60,39-9	72	72	72	72	0	1.07	90	0	0
J60,39-10	74	74	74	74	0	1.09	90	0	0
J60,40-1	86	86	86	86	0	1.03	90	0	0
J60,40-2	81	81	81	81	0	1.04	90	0	0
J60,40-3	70	70	70	70	0	1.05	90	0	0
J60,40-4	87	87	87	87	0	1.08	90	0	0
J60,40-5	83	83	83	83	0	1.06	90	0	0
J60,40-6	69	69	69	69	0	1.09	90	0	0
J60,40-7	68	68	68	68	0	1.04	90	0	0
J60,40-8	80	80	80	80	0	1.06	90	0	0
J60,40-9	90	90	90	90	0	1.04	90	0	0
J60,40-10	73	73	73	73	0	1.04	90	0	41.76
J60,41-1	129	135	134.77	145	3.05	213.73	12754	5.74	46.91
J60,41-2	119	122	122.13	126	1.89	231.38	13112	5.31	79.31
J60,41-3	104	108	107.53	111	1.81	250.5	15033	6.12	36
J60,41-4	136	141	141.5	148	2.86	189.98	15121	2.26	83.58
J60,41-5	123	126	126.2	130	1.67	223.12	14071	6.96	70.73
J60,41-6	140	144	144.57	149	1.99	204.14	13714	4.48	77.22
J60,41-7	140	147.5	146.83	154	3.58	213.43	13112	6.06	43.16
J60,41-8	136	146	146.53	158	4.72	245.79	12697	0.74	75
J60,41-9	140	144.5	144.87	151	2.79	202.03	13059	6.87	60.56
J60,41-10	114	119	119.67	125	2.26	224.05	13234	2.7	0
J60,42-1	83	83	83.3	85	0.6	61.12	931	0	0
J60,42-2	68	68	68.23	71	0.63	62.12	273	0	9.59

J60,42-3	80	83	83.27	86	1.41	194.5	13416	2.56	10.64
J60,42-4	104	108	107.73	111	1.64	173.7	12813	0.97	0
J60,42-5	73	74	73.97	77	1	125.9	878	0	0
J60,42-6	82	83.5	83.67	89	1.63	124.31	1172	0	9.09
J60,42-7	60	61.5	61.63	64	0.85	180.29	12754	1.69	6.41
J60,42-8	83	86	85.63	88	1.13	177.34	13231	1.22	2.86
J60,42-9	72	74	73.87	76	1.11	177.45	12571	1.41	0
J60,42-10	87	87	87.57	90	0.82	96.91	814	0	0
J60,43-1	108	108	108	108	0	1.11	90	0	0
J60,43-2	85	85	85	85	0	1.14	90	0	0
J60,43-3	74	74	74	74	0	1.11	90	0	2.7
J60,43-4	76	78	77.5	80	0.82	152.79	12032	1.33	0
J60,43-5	64	64	64	64	0	1.52	90	0	0
J60,43-6	84	84	84	84	0	1.08	90	0	0
J60,43-7	89	89	89	89	0	1.13	90	0	0
J60,43-8	69	69	69	69	0	1.04	90	0	0
J60,43-9	70	70	70.37	71	0.49	64.95	213	0	0
J60,43-10	78	78	78	78	0	1.09	90	0	0
J60,44-1	84	84	84	84	0	1.06	90	0	0
J60,44-2	68	68	68	68	0	1.03	90	0	0
J60,44-3	87	87	87	87	0	1.03	90	0	0
J60,44-4	77	77	77	77	0	1.08	90	0	0
J60,44-5	74	74	74	74	0	1.08	90	0	0
J60,44-6	81	81	81	81	0	1.06	90	0	0
J60,44-7	76	76	76	76	0	1.07	90	0	0
J60,44-8	83	83	83	83	0	1.04	90	0	0
J60,44-9	65	65	65	65	0	1.05	90	0	0
J60,44-10	65	65	65	65	0	1.07	90	0	40.85
J60,45-1	100	104.5	104.27	109	2.41	234.41	15302	4.17	93.59
J60,45-2	151	155	155.77	162	2.6	270.41	15992	4.86	79.07
J60,45-3	154	159.5	159.43	165	2.73	236.95	13173	7.69	91.67
J60,45-4	115	118	118.5	124	2.18	243.6	13234	6.48	85
J60,45-5	111	116	115.4	120	2.14	291.21	13234	4.72	87.5
J60,45-6	150	157	156.9	162	2.82	280.9	13535	4.17	70.27
J60,45-7	126	131.5	131.13	135	2.33	250.52	13293	3.28	73.42
J60,45-8	137	139	139.5	144	2.1	298.21	14945	6.2	66.67
J60,45-9	130	136	135.8	140	2.71	277.21	12697	5.69	100
J60,45-10	122	124	124.83	128	1.7	269.58	14974	7.02	3.85
J60,46-1	81	83	83.37	85	1.22	189.21	12515	2.53	0
J60,46-2	78	80	79.63	82	0.93	174.17	3873	0	2.53
J60,46-3	81	83	83.37	86	1.25	211.34	13170	2.53	8.45
J60,46-4	77	79	78.57	80	0.94	223.33	12873	4.05	14.63
J60,46-5	94	96	95.73	97	1.23	192.77	12391	3.3	5.75
J60,46-6	92	95	94.87	97	1.07	195.54	12391	2.22	8
J60,46-7	81	82	82.23	84	0.94	223	13350	3.85	6.94
J60,46-8	77	80	79.73	84	1.62	195.49	12459	2.67	16.67
J60,46-9	70	72	72.23	74	1.04	188.19	12458	1.45	13.92
J60,46-10	90	92	92.33	98	1.73	211.14	12571	2.27	0
J60,47-1	75	75	75	75	0	1.27	90	0	0
J60,47-2	66	66	66.3	68	0.53	75.69	932	0	0
J60,47-3	69	69	69.6	72	1.04	81.15	992	0	0
J60,47-4	76	76	76	76	0	1.71	90	0	0
J60,47-5	87	87	87.23	88	0.43	41.68	91	0	0

J60,47-6	76	76	76	76	0	1.27	90	0	0
J60,47-7	68	68	68	68	0	4.83	91	0	0
J60,47-8	71	71	71.5	73	0.57	102.43	1172	0	0
J60,47-9	76	76	76	76	0	9.21	91	0	0
J60,47-10	66	67	66.9	69	0.76	145.4	573	0	0
J60,48-1	71	71	71	71	0	1.08	90	0	0
J60,48-2	87	87	87	87	0	1.16	90	0	0
J60,48-3	84	84	84	84	0	1.14	90	0	0
J60,48-4	62	62	62	62	0	1.12	90	0	0
J60,48-5	101	101	101	101	0	1.13	90	0	0
J60,48-6	66	66	66	66	0	1.12	90	0	0
J60,48-7	77	77	77	77	0	1.17	90	0	0
J60,48-8	88	88	88	88	0	1.19	90	0	0
J60,48-9	82	82	82	82	0	1.16	90	0	0
J60,48-10	70	70	70	70	0	1.24	90	0	0
Total Average					**0.80**	**96.96**	**3788.83**	**1.22**	**12.10**

Table 15

The detailed results of 480 problems of J90 obtained by bi-EA

The results are obtained from 30 runs with up to 50,000 fitness evaluations for each.

Prob. No	Best	Median	Mean	Worst	STD	t	FE	LB_{OP}	LB_{CP}
J90,1-1	76	81	81.07	86	2.3	559.66	15461	4.11	19.4
J90,1-2	92	96	95.33	100	3.1	319.19	105	0	4.55
J90,1-3	68	71.5	71.93	76	1.89	544.24	15663	3.03	18.64
J90,1-4	87	91	91.57	97	2.53	527.32	15413	1.16	17.11
J90,1-5	90	90	91.23	94	1.59	440.82	15561	3.45	5.95
J90,1-6	78	81	81	86	2.15	506.21	15408	5.41	27.87
J90,1-7	93	97.5	97.03	100	1.88	463.94	15202	2.2	16.87
J90,1-8	98	104.5	104	109	3.02	488.55	15101	3.16	20
J90,1-9	76	77	77.97	82	1.65	485.44	15206	5.56	13.64
J90,1-10	94	98	98	104	2.2	520.42	15103	4.44	11.49
J90,2-1	96	96	96	96	0	3.3	101	0	0
J90,2-2	114	114	114	114	0	3.26	101	0	0
J90,2-3	75	75	75	75	0	3.65	101	0	0
J90,2-4	70	70	70	70	0	3.24	101	0	0
J90,2-5	100	100	100.1	103	0.55	16.67	101	0	0
J90,2-6	67	67	67	67	0	4.8	101	0	0
J90,2-7	92	92	92.03	93	0.18	17.62	101	0	0
J90,2-8	82	82	82	82	0	3.79	101	0	0
J90,2-9	79	79	79.47	81	0.86	142.13	104	0	0
J90,2-10	80	80	80	80	0	4.28	101	0	0
J90,3-1	81	81	81	81	0	3.31	101	0	0
J90,3-2	84	84	84	84	0	3.26	101	0	0
J90,3-3	71	71	71	71	0	3.21	101	0	0
J90,3-4	104	104	104	104	0	3.32	101	0	0
J90,3-5	75	75	75	75	0	3.91	101	0	0
J90,3-6	68	68	68	68	0	3.24	101	0	0
J90,3-7	87	87	87	87	0	3.86	101	0	0
J90,3-8	86	86	86	86	0	3.25	101	0	0
J90,3-9	61	61	61	61	0	3.29	101	0	0
J90,3-10	65	65	65.07	67	0.37	22.77	103	0	0
J90,4-1	93	93	93	93	0	3.16	101	0	0
J90,4-2	89	89	89	89	0	3.2	101	0	0
J90,4-3	67	67	67	67	0	3.15	101	0	0
J90,4-4	92	92	92	92	0	3.13	101	0	0
J90,4-5	88	88	88	88	0	3.21	101	0	0
J90,4-6	78	78	78	78	0	3.12	101	0	0
J90,4-7	80	80	80	80	0	3.14	101	0	0
J90,4-8	69	69	69	69	0	3.17	101	0	0
J90,4-9	79	79	79	79	0	3.22	101	0	0
J90,4-10	68	68	68	68	0	3.09	101	0	0
J90,5-1	86	90.5	90.8	95	2.55	599.39	15105	10.26	36.36

J90,5-2	104	111	111.1	117	2.94	622.2	12991	11.83	17.78
J90,5-3	95	99	99.1	105	2.45	564.34	13295	9.2	59.68
J90,5-4	112	118.5	119	127	3.31	546.25	12989	8.74	26.67
J90,5-5	122	126.5	126.93	134	2.74	640.83	12995	9.91	60.76
J90,5-6	93	98	98.93	108	3.64	565.98	13293	8.14	29.33
J90,5-7	114	119	118.77	122	2.01	638.23	14021	6.54	48.1
J90,5-8	111	115	115.27	122	2.24	663.87	12953	8.82	52
J90,5-9	128	133	133.63	142	3.24	624.16	12331	11.3	42.55
J90,5-10	104	108	108.37	114	2.25	586.97	12679	8.33	39.74
J90,6-1	82	82	82.27	85	0.69	94.24	81	0	0
J90,6-2	86	86	86	86	0	3.04	78	0	0
J90,6-3	81	84	84.17	88	1.68	547.42	12564	5.19	8
J90,6-4	80	80	80	80	0	8.16	78	0	0
J90,6-5	71	73	72.5	74	1.01	411.97	2191	0	1.41
J90,6-6	98	98	98	98	0	6.71	78	0	0
J90,6-7	71	71	71.03	72	0.18	44.41	81	0	0
J90,6-8	71	74	73.97	79	1.79	491.56	12177	4.41	7.46
J90,6-9	68	68	68.57	71	0.82	195.47	393	0	0
J90,6-10	94	94	94.1	97	0.55	25.59	78	0	0
J90,7-1	88	88	88	88	0	2.78	78	0	0
J90,7-2	77	77	77	77	0	2.81	78	0	0
J90,7-3	80	80	80.07	82	0.37	21.1	78	0	0
J90,7-4	86	86	86	86	0	2.95	78	0	0
J90,7-5	79	79	79	79	0	2.95	78	0	0
J90,7-6	90	90	90	90	0	2.92	78	0	0
J90,7-7	90	90	90	90	0	2.88	78	0	0
J90,7-8	60	60	60.4	62	0.67	131.67	79	0	0
J90,7-9	83	83	83	83	0	3.41	79	0	0
J90,7-10	98	98	98	98	0	2.89	78	0	0
J90,8-1	96	96	96	96	0	2.75	78	0	0
J90,8-2	78	78	78	78	0	2.85	78	0	0
J90,8-3	70	70	70	70	0	2.65	78	0	0
J90,8-4	77	77	77	77	0	2.78	78	0	0
J90,8-5	63	63	63	63	0	2.73	78	0	0
J90,8-6	70	70	70	70	0	2.73	78	0	0
J90,8-7	77	77	77	77	0	2.7	78	0	0
J90,8-8	68	68	68	68	0	2.7	78	0	0
J90,8-9	97	97	97	97	0	2.76	78	0	0
J90,8-10	88	88	88	88	0	2.75	78	0	0
J90,9-1	116	121	120.73	126	2.77	518.02	16139	11.54	52.5
J90,9-2	139	144	144	150	2.45	457.38	15542	8.59	56.52
J90,9-3	113	116	116.3	120	2.05	439.53	15135	10.78	74.63
J90,9-4	139	143.5	144.27	152	3.4	479.45	16322	9.45	80.25
J90,9-5	154	160	160.2	166	3.01	451.58	12933	10	80.9
J90,9-6	130	134	134	140	2.42	416.67	14555	9.24	46.59
J90,9-7	120	124	123.87	128	2.22	486.87	14138	10.09	66.22
J90,9-8	128	133	133.37	138	2.19	423.05	13834	10.34	76
J90,9-9	127	131	131.6	137	2.71	445.59	12338	9.48	71.43
J90,9-10	122	124	125	131	2.12	435.01	14853	9.91	61.04
J90,10-1	78	81	80.8	84	1.71	307.27	12033	1.3	3.9
J90,10-2	95	95	95.2	97	0.48	105.69	333	0	0
J90,10-3	112	112	112	112	0	2.09	90	0	0
J90,10-4	94	94	94	94	0	2.05	90	0	0

J90,10-5	78	78	78	78	0	2.63	90	0	0
J90,10-6	92	92	92	92	0	2.11	90	0	0
J90,10-7	83	83	83	83	0	12.27	90	0	0
J90,10-8	82	85	85.47	90	1.94	317.69	12152	1.23	3.7
J90,10-9	88	88	88	88	0	2.06	90	0	0
J90,10-10	77	79	78.57	82	1.1	334.25	12277	2.67	2.67
J90,11-1	86	86	86	86	0	2.04	90	0	0
J90,11-2	99	99	99.03	100	0.18	15.21	90	0	0
J90,11-3	69	69	69.03	70	0.18	27.83	90	0	0
J90,11-4	64	64	64	64	0	3.16	90	0	0
J90,11-5	81	81	81	81	0	2.05	90	0	0
J90,11-6	78	78	78	78	0	5.19	90	0	0
J90,11-7	95	95	95	95	0	2.02	90	0	0
J90,11-8	82	82	82	82	0	2.26	90	0	0
J90,11-9	81	81	81	81	0	2.02	90	0	0
J90,11-10	81	81	81	81	0	2.04	90	0	0
J90,12-1	71	71	71	71	0	2.05	90	0	0
J90,12-2	71	71	71	71	0	1.95	90	0	0
J90,12-3	93	93	93	93	0	1.95	90	0	0
J90,12-4	73	73	73	73	0	1.96	90	0	0
J90,12-5	83	83	83	83	0	1.93	90	0	0
J90,12-6	81	81	81	81	0	1.97	90	0	0
J90,12-7	77	77	77	77	0	1.98	90	0	0
J90,12-8	83	83	83	83	0	1.96	90	0	0
J90,12-9	77	77	77	77	0	1.93	90	0	0
J90,12-10	86	86	86	86	0	1.99	90	0	0
J90,13-1	150	154	154.03	158	2.16	527.89	14864	8.7	89.02
J90,13-2	140	143	143.13	146	1.91	535.85	14725	10.24	88.31
J90,13-3	119	124	123.4	128	2.27	425.42	14311	10.19	51.25
J90,13-4	122	126	125.63	130	2.09	428.61	13652	8.93	54.32
J90,13-5	127	134	133.13	138	3.17	439.02	13710	10.43	63.29
J90,13-6	135	139	139	145	2.35	428.76	14071	8.87	78.95
J90,13-7	137	141	140.8	150	2.94	419.74	12035	10.48	58.43
J90,13-8	125	129	129.5	135	2.13	455.9	13772	6.84	123.73
J90,13-9	136	140	140.13	144	2.22	423.74	12936	9.68	69.51
J90,13-10	129	132.5	132.83	138	2.1	444.17	14852	7.5	91.43
J90,14-1	89	90	90.13	94	1.28	198.92	90	0	0
J90,14-2	79	80	80.03	83	1.22	196.29	95	0	0
J90,14-3	94	94	94	94	0	1.99	90	0	0
J90,14-4	88	89	88.8	91	0.85	217.16	93	0	0
J90,14-5	84	84	84	84	0	2.18	90	0	0
J90,14-6	79	81	81.67	85	1.58	320.87	12521	3.95	3.95
J90,14-7	86	86	86	86	0	1.91	90	0	0
J90,14-8	80	80.5	81.13	84	1.31	226.91	91	0	0
J90,14-9	112	112	112	112	0	2.07	90	0	0
J90,14-10	85	87	87.53	92	1.55	320.26	2076	0	1.18
J90,15-1	76	76	76	76	0	1.97	90	0	0
J90,15-2	71	71	71	71	0	1.96	90	0	0
J90,15-3	82	82	82	82	0	1.96	90	0	0
J90,15-4	92	92	92	92	0	1.95	90	0	0
J90,15-5	93	93	93	93	0	2	90	0	0
J90,15-6	61	61	61.17	62	0.38	85.4	91	0	0
J90,15-7	82	82	82	82	0	1.94	90	0	0

J90,15-8	82	82	82	82	0	9.71	90	0	0
J90,15-9	83	83	83	83	0	2.46	90	0	0
J90,15-10	78	78	78	78	0	1.92	90	0	0
J90,16-1	85	85	85	85	0	1.86	90	0	0
J90,16-2	71	71	71	71	0	1.83	90	0	0
J90,16-3	73	73	73	73	0	1.91	90	0	0
J90,16-4	69	69	69	69	0	1.84	90	0	0
J90,16-5	71	71	71	71	0	1.89	90	0	0
J90,16-6	74	74	74	74	0	1.8	90	0	0
J90,16-7	65	65	65	65	0	1.87	90	0	0
J90,16-8	71	71	71	71	0	1.82	90	0	0
J90,16-9	66	66	66	66	0	1.87	90	0	0
J90,16-10	71	71	71	71	0	1.85	90	0	0
J90,17-1	96	101	100.5	105	2.1	252.73	12094	4.35	25.64
J90,17-2	101	107	106.77	113	2.56	258.69	12819	1	18.39
J90,17-3	89	94	93.93	98	2.08	251.94	2135	0	5.81
J90,17-4	94	97	96.6	101	1.98	234.87	991	0	2.17
J90,17-5	113	113	113	113	0	2.85	90	0	3.67
J90,17-6	95	98	98.23	102	2.28	276.33	12033	1.06	8.79
J90,17-7	80	83	83.87	90	2.93	214.53	1595	0	10.96
J90,17-8	113	116	116.4	125	2.63	228.1	882	0	11.88
J90,17-9	97	101	100.73	104	2.35	247.78	12213	1.04	22.5
J90,17-10	91	94	94.4	99	1.79	258.7	12274	2.25	14.81
J90,18-1	101	101	101	101	0	1.84	90	0	0
J90,18-2	94	94	94	94	0	3.22	90	0	0
J90,18-3	83	83	83	83	0	1.83	90	0	0
J90,18-4	98	98	98	98	0	1.83	90	0	0
J90,18-5	90	90	90	90	0	9.43	90	0	0
J90,18-6	83	83	83.1	84	0.31	30.96	90	0	0
J90,18-7	73	73	73	73	0	5.48	90	0	0
J90,18-8	92	92	92	92	0	1.85	90	0	0
J90,18-9	79	79	79	79	0	1.8	90	0	0
J90,18-10	94	94	94	94	0	1.83	90	0	0
J90,19-1	98	98	98	98	0	1.8	90	0	0
J90,19-2	83	83	83	83	0	2.4	90	0	0
J90,19-3	89	89	89	89	0	1.82	90	0	0
J90,19-4	77	77	77	77	0	1.81	90	0	0
J90,19-5	66	66	66	66	0	1.8	90	0	0
J90,19-6	136	136	136	136	0	1.79	90	0	0
J90,19-7	66	66	66	66	0	1.94	90	0	0
J90,19-8	91	91	91	91	0	1.82	90	0	0
J90,19-9	121	121	121	121	0	1.81	90	0	0
J90,19-10	85	85	85	85	0	1.82	90	0	0
J90,20-1	85	85	85	85	0	1.78	90	0	0
J90,20-2	76	76	76	76	0	1.83	90	0	0
J90,20-3	86	86	86	86	0	1.81	90	0	0
J90,20-4	86	86	86	86	0	1.84	90	0	0
J90,20-5	88	88	88	88	0	1.82	90	0	0
J90,20-6	83	83	83	83	0	1.92	90	0	0
J90,20-7	82	82	82	82	0	1.83	90	0	0
J90,20-8	85	85	85	85	0	1.88	90	0	0
J90,20-9	76	76	76	76	0	1.85	90	0	0
J90,20-10	89	89	89	89	0	1.82	90	0	0

J90,21-1	123	126.5	126.93	131	2.1	813.68	17246	11.82	61.84
J90,21-2	128	132	132.67	137	2.83	735.72	14438	10.34	37.63
J90,21-3	134	142	141.83	148	3.78	714.9	15003	8.06	32.67
J90,21-4	117	123	123.43	132	3.35	563.88	12572	10.38	30
J90,21-5	121	126.5	126.37	132	2.85	498.43	13053	8.04	51.25
J90,21-6	115	121	120.93	126	2.88	444.87	12274	8.49	32.18
J90,21-7	119	125.5	125.83	130	2.73	460.98	13952	9.17	30.77
J90,21-8	119	128	127.43	134	3.41	502.9	14258	7.21	40
J90,21-9	128	135	135.7	145	3.74	494.2	13114	5.79	40.66
J90,21-10	117	124	123.83	130	2.73	467.06	13352	7.34	44.44
J90,22-1	108	108	108	108	0	2.6	90	0	0
J90,22-2	85	85	85	85	0	15.65	90	0	0
J90,22-3	86	87	87.4	90	1.43	415.37	12154	3.61	3.61
J90,22-4	96	96	96.33	99	0.76	121.61	90	0	0
J90,22-5	96	96	96.17	98	0.53	53.64	91	0	0
J90,22-6	71	71	71.03	72	0.18	57.38	93	0	0
J90,22-7	90	90	90	90	0	10.52	90	0	0
J90,22-8	97	97	97	97	0	2.53	90	0	0
J90,22-9	104	108.5	108.33	113	2.45	416.69	12031	2.97	7.22
J90,22-10	78	81	80.83	84	1.6	396.33	12032	4	4
J90,23-1	90	90	90	90	0	2.6	90	0	0
J90,23-2	84	84	84	84	0	2.44	90	0	0
J90,23-3	116	116	116	116	0	2.53	90	0	0
J90,23-4	85	85	85	85	0	2.83	90	0	0
J90,23-5	95	95	95	95	0	2.49	90	0	0
J90,23-6	87	87	87	87	0	2.89	90	0	0
J90,23-7	77	77	77	77	0	2.52	90	0	0
J90,23-8	92	92	92	92	0	2.5	90	0	0
J90,23-9	126	126	126	126	0	2.56	90	0	0
J90,23-10	87	87	87	87	0	2.48	90	0	0
J90,24-1	84	84	84	84	0	2.44	90	0	0
J90,24-2	92	92	92	92	0	2.5	90	0	0
J90,24-3	69	69	69	69	0	2.39	90	0	0
J90,24-4	81	81	81	81	0	2.45	90	0	0
J90,24-5	85	85	85	85	0	2.45	90	0	0
J90,24-6	79	79	79	79	0	2.43	90	0	0
J90,24-7	87	87	87	87	0	2.41	90	0	0
J90,24-8	88	88	88	88	0	2.44	90	0	0
J90,24-9	80	80	80	80	0	2.49	90	0	0
J90,24-10	89	89	89	89	0	2.54	90	0	0
J90,25-1	134	140	140.07	146	2.9	401.06	14671	8.06	42.55
J90,25-2	143	148.5	148.4	155	2.79	379.14	14130	9.16	72.29
J90,25-3	137	143	142.77	149	2.74	374.84	14914	11.38	77.92
J90,25-4	148	154	153.3	158	2.73	423.99	16803	7.25	55.79
J90,25-5	126	129.5	129.7	135	2.26	351.61	13714	10.53	77.46
J90,25-6	134	137	137	143	2.3	375.54	13113	10.74	69.62
J90,25-7	143	148	147.67	156	3.09	385.48	13953	10	92.21
J90,25-8	159	163	163.93	171	3.2	438.51	15454	13.57	62.38
J90,25-9	115	119	119.2	122	2.02	349.92	13114	8.49	51.25
J90,25-10	142	147	147.9	155	2.95	389.41	14312	9.23	56.25
J90,26-1	90	90	90	90	0	30.43	92	0	0
J90,26-2	89	93	93.4	97	1.75	316.1	12512	4.71	9.41
J90,26-3	80	81	80.8	82	0.76	180.55	91	0	0

J90,26-4	103	105.5	105.7	108	1.37	316.82	13051	6.19	10.42
J90,26-5	90	94	93.37	96	1.69	334.63	12931	5.88	12.2
J90,26-6	108	108	108.03	109	0.18	26.79	90	0	0
J90,26-7	85	87	87.03	89	1.19	271.34	12690	3.66	6.1
J90,26-8	87	89	89.33	91	0.99	294.85	12696	6.1	9.76
J90,26-9	88	92	92.53	96	2.11	316.76	12151	1.15	4.6
J90,26-10	92	92	92	92	0	1.91	90	0	0
J90,27-1	96	96	96	96	0	1.77	90	0	0
J90,27-2	81	81	81.03	82	0.18	12.55	90	0	0
J90,27-3	91	91	91	91	0	1.75	90	0	0
J90,27-4	79	79	79	79	0	2.91	90	0	0
J90,27-5	99	99	99	99	0	1.74	90	0	0
J90,27-6	87	87	87	87	0	1.76	90	0	0
J90,27-7	73	73	73.37	76	0.76	54.07	90	0	0
J90,27-8	72	72	72	72	0	1.76	90	0	0
J90,27-9	84	84	84.43	86	0.63	119.73	634	0	0
J90,27-10	97	97	97	97	0	1.78	90	0	0
J90,28-1	80	80	80	80	0	1.7	90	0	0
J90,28-2	76	76	76	76	0	1.65	90	0	0
J90,28-3	86	86	86	86	0	1.68	90	0	0
J90,28-4	78	78	78	78	0	1.74	90	0	0
J90,28-5	88	88	88	88	0	1.69	90	0	0
J90,28-6	102	102	102	102	0	1.69	90	0	0
J90,28-7	97	97	97	97	0	1.65	90	0	0
J90,28-8	110	110	110	110	0	1.71	90	0	0
J90,28-9	120	120	120	120	0	1.76	90	0	0
J90,28-10	68	68	68	68	0	1.72	90	0	0
J90,29-1	149	153.5	154	161	2.77	405.7	12158	10.37	61.46
J90,29-2	138	142	141.6	145	1.87	355.98	13353	9.52	86.84
J90,29-3	157	161	160.77	166	2.42	407.75	15933	9.03	85.06
J90,29-4	161	164.5	164.73	168	1.93	447.85	13772	8.05	94.25
J90,29-5	131	136.5	136.47	141	2.83	387.53	13833	8.26	42.71
J90,29-6	133	138	137.97	144	2.57	386.21	12452	7.26	81.82
J90,29-7	184	189.5	189.53	196	3.08	466.96	14082	8.24	85.44
J90,29-8	165	173	172.93	179	2.94	428.53	15030	7.14	88.17
J90,29-9	137	142	141.97	145	1.87	433.15	13533	7.87	77.78
J90,29-10	134	139	138.9	145	2.28	372.27	14013	7.2	76.54
J90,30-1	102	102	102	102	0	1.79	90	0	0
J90,30-2	76	79	78.7	82	1.53	280.43	93	0	2.63
J90,30-3	102	105	105.4	109	1.65	302.63	13113	0	2.94
J90,30-4	104	104	104.83	109	1.18	223.86	3032	0	0
J90,30-5	88	92	91.9	96	1.75	322.72	13594	6.02	13.25
J90,30-6	90	90	90	90	0	2.1	90	0	0
J90,30-7	89	92	92.7	96	1.64	285.64	12875	5.95	9.52
J90,30-8	83	86	86.07	90	1.48	281.91	12341	1.22	3.66
J90,30-9	100	103	102.87	107	1.43	331.19	13711	6.38	26.83
J90,30-10	90	93	93.37	97	1.61	326.35	12992	0	4.44
J90,31-1	79	79	79	79	0	1.8	90	0	0
J90,31-2	69	71	71.07	73	0.83	253.85	10805	0	2.9
J90,31-3	106	106	106	106	0	1.73	90	0	0
J90,31-4	79	79	79	79	0	1.73	90	0	0
J90,31-5	79	79	79	79	0	1.72	90	0	0
J90,31-6	80	80	80	80	0	3.52	90	0	0

J90,31-7	97	97	97	97	0	1.7	90	0	0
J90,31-8	83	83	83	83	0	1.83	90	0	0
J90,31-9	72	72	72	72	0	1.74	90	0	0
J90,31-10	99	99	99	99	0	4.84	90	0	0
J90,32-1	78	78	78	78	0	1.61	90	0	0
J90,32-2	78	78	78	78	0	1.62	90	0	0
J90,32-3	89	89	89	89	0	1.65	90	0	0
J90,32-4	104	104	104	104	0	1.65	90	0	0
J90,32-5	93	93	93	93	0	1.66	90	0	0
J90,32-6	86	86	86	86	0	1.74	90	0	0
J90,32-7	87	87	87	87	0	1.7	90	0	0
J90,32-8	79	79	79	79	0	1.68	90	0	0
J90,32-9	95	95	95	95	0	1.71	90	0	0
J90,32-10	91	91	91	91	0	1.82	90	0	0
J90,33-1	99	105	104.4	109	2.36	218.98	5074	0	25.61
J90,33-2	112	113	112.53	113	0.51	135.46	92	0	4.67
J90,33-3	108	112.5	112.27	119	2.69	217.22	3422	0	3.81
J90,33-4	93	98	97.23	101	1.77	194.57	12032	1.09	11.76
J90,33-5	111	113	113.83	118	2.26	230.69	12031	1.83	5.66
J90,33-6	88	89	89.43	91	1.36	166.31	2617	0	6.02
J90,33-7	109	109	110.5	118	2.49	112.39	92	0	17.02
J90,33-8	110	111	111.5	114	1.22	189.33	454	0	11.11
J90,33-9	97	101.5	101.8	108	2.68	234.87	12031	2.11	12.79
J90,33-10	116	118	118.37	122	1.9	232.5	12036	1.75	2.65
J90,34-1	83	83	83	83	0	7.34	90	0	0
J90,34-2	89	89	89	89	0	1.69	90	0	0
J90,34-3	82	82	82	82	0	5.62	91	0	0
J90,34-4	81	83	82.67	85	1.12	172.5	156	0	6.58
J90,34-5	83	85	84.87	87	1.38	167.72	512	0	3.75
J90,34-6	89	89	89	89	0	1.6	90	0	0
J90,34-7	92	92	92	92	0	1.55	90	0	0
J90,34-8	81	82	81.73	84	0.78	134.91	813	0	3.85
J90,34-9	109	109	109	109	0	1.59	90	0	0
J90,34-10	101	101	101	101	0	2.05	90	0	0
J90,35-1	98	98	98	98	0	1.5	90	0	1.03
J90,35-2	92	92	92	92	0	1.58	90	0	0
J90,35-3	96	96	96	96	0	1.55	90	0	0
J90,35-4	86	86	86	86	0	1.54	90	0	0
J90,35-5	103	103	103	103	0	1.54	90	0	0
J90,35-6	72	72	72	72	0	4.07	90	0	0
J90,35-7	78	78	78	78	0	2.79	90	0	0
J90,35-8	85	85	85	85	0	1.54	90	0	0
J90,35-9	76	76	76	76	0	1.79	90	0	0
J90,35-10	82	82	82	82	0	1.53	90	0	0
J90,36-1	97	97	97	97	0	1.48	90	0	0
J90,36-2	114	114	114	114	0	1.61	90	0	0
J90,36-3	84	84	84	84	0	1.55	90	0	0
J90,36-4	79	79	79	79	0	1.53	90	0	0
J90,36-5	98	98	98	98	0	1.51	90	0	0
J90,36-6	99	99	99	99	0	1.52	90	0	0
J90,36-7	89	89	89	89	0	1.55	90	0	0
J90,36-8	84	84	84	84	0	1.53	90	0	0
J90,36-9	102	102	102	102	0	1.5	90	0	0

J90,36-10	109	109	109	109	0	1.51	90	0	0
J90,37-1	117	123	122.73	126	2.36	292.83	13353	6.36	39.29
J90,37-2	122	128	128.2	134	2.91	356.06	13114	6.09	35.56
J90,37-3	139	147	147.03	155	3.52	313.08	12995	5.3	20.87
J90,37-4	132	140.5	140.43	149	3.52	335.87	13296	7.32	53.49
J90,37-5	136	142.5	141.8	147	3.22	321.96	13051	7.94	47.83
J90,37-6	141	145.5	145.87	151	2.69	323.18	14491	7.63	46.88
J90,37-7	130	136.5	135.9	143	3.21	293.54	13772	5.69	34.02
J90,37-8	130	134	134.2	139	2.38	354.49	14942	9.24	46.07
J90,37-9	133	140	141.03	147	3.21	291.95	14132	8.13	43.01
J90,37-10	133	139.5	139.53	147	3	311.72	13475	8.13	49.44
J90,38-1	87	88	88.5	93	1.28	244.69	12033	2.35	4.82
J90,38-2	78	82	81.37	85	1.38	222.88	10053	0	1.3
J90,38-3	90	93	93.43	97	1.38	267.1	12281	1.12	2.27
J90,38-4	89	89	89.5	92	0.82	119.5	92	0	0
J90,38-5	89	91.5	91.47	95	1.55	265.07	12455	3.49	5.95
J90,38-6	89	91	91.13	93	1.11	257.57	12035	1.14	1.14
J90,38-7	85	85	85.03	86	0.18	50.26	91	0	0
J90,38-8	92	97.5	97.07	100	2.26	245.01	12032	1.1	1.1
J90,38-9	96	99	99.27	104	2.08	265.93	12030	1.05	1.05
J90,38-10	108	108	108	108	0	4.65	90	0	0
J90,39-1	106	106	106	106	0	1.68	90	0	0
J90,39-2	119	119	119	119	0	1.93	90	0	0
J90,39-3	83	83	83	83	0	5.23	90	0	0
J90,39-4	81	81	81.4	84	0.77	93.05	91	0	0
J90,39-5	85	85	85.3	87	0.6	78.81	333	0	0
J90,39-6	102	102	102	102	0	4.57	90	0	0
J90,39-7	85	85	85	85	0	2.63	90	0	0
J90,39-8	81	81	81.13	85	0.73	21.69	90	0	0
J90,39-9	79	79	79	79	0	1.68	90	0	0
J90,39-10	100	100	100	100	0	1.71	90	0	0
J90,40-1	95	95	95	95	0	1.64	90	0	0
J90,40-2	91	91	91	91	0	1.63	90	0	0
J90,40-3	77	77	77	77	0	1.64	90	0	0
J90,40-4	106	106	106	106	0	1.65	90	0	0
J90,40-5	92	92	92	92	0	1.66	90	0	0
J90,40-6	86	86	86	86	0	1.63	90	0	0
J90,40-7	87	87	87	87	0	1.64	90	0	0
J90,40-8	79	79	79	79	0	1.6	90	0	0
J90,40-9	98	98	98	98	0	1.64	90	0	0
J90,40-10	86	86	86	86	0	1.68	90	0	0
J90,41-1	153	157	157.3	162	2.29	399.67	14130	7.75	50
J90,41-2	186	192	192.03	201	3.47	443.1	15662	10.71	72.22
J90,41-3	177	182	182.03	190	3.34	421.18	14133	9.94	66.98
J90,41-4	166	172.5	171.87	177	2.99	478.06	15480	7.79	95.29
J90,41-5	138	144	143.97	153	3.59	420.8	13176	8.66	55.06
J90,41-6	147	151	151	158	2.72	349.65	12572	8.89	70.93
J90,41-7	172	180	179.07	185	3.02	414.16	15003	9.55	63.81
J90,41-8	176	182.5	182.8	191	3.11	481.02	14558	7.98	62.96
J90,41-9	132	137	137	144	3.17	412.51	14945	10.92	71.43
J90,41-10	164	169	168.63	172	2.46	428.99	15305	9.33	56.19
J90,42-1	106	108.5	108.7	113	1.73	357.64	4997	0	0
J90,42-2	108	110	110.7	116	1.7	339.2	12039	5.88	17.39

J90,42-3	95	98	97.9	101	1.35	268.45	12212	1.06	1.06
J90,42-4	102	102	102.37	105	0.72	86.49	91	0	0
J90,42-5	106	107	107.13	111	1.22	266.72	12034	0.95	0.95
J90,42-6	89	91	90.63	91	0.72	194.12	2555	0	0
J90,42-7	91	92	92.23	94	1.04	282.91	12393	4.6	9.64
J90,42-8	105	105	105	105	0	1.68	90	0	0
J90,42-9	87	89	89.47	93	1.36	298.55	13416	4.82	6.1
J90,42-10	96	98.5	98.6	105	2.01	285.03	12395	6.67	12.94
J90,43-1	99	99	99.33	104	0.99	90.79	93	0	0
J90,43-2	91	91	91	91	0	5.66	90	0	0
J90,43-3	102	102	102	102	0	1.66	90	0	0
J90,43-4	94	94	94	94	0	1.96	90	0	0
J90,43-5	98	98	98	98	0	1.65	90	0	0
J90,43-6	114	114	114	114	0	1.64	90	0	0
J90,43-7	88	88	88.1	89	0.31	49.94	91	0	0
J90,43-8	100	100	100	100	0	26.48	90	0	0
J90,43-9	88	88	88	88	0	1.72	90	0	0
J90,43-10	92	92	92.03	93	0.18	24.72	90	0	0
J90,44-1	100	100	100	100	0	1.61	90	0	0
J90,44-2	92	92	92	92	0	1.59	90	0	0
J90,44-3	110	110	110	110	0	1.62	90	0	0
J90,44-4	89	89	89	89	0	1.58	90	0	0
J90,44-5	84	84	84	84	0	1.6	90	0	0
J90,44-6	96	96	96	96	0	1.62	90	0	0
J90,44-7	93	93	93	93	0	1.59	90	0	0
J90,44-8	99	99	99	99	0	1.6	90	0	0
J90,44-9	96	96	96	96	0	1.62	90	0	0
J90,44-10	86	86	86	86	0	1.61	90	0	0
J90,45-1	157	162	162.13	169	2.78	430.81	13296	7.53	68.82
J90,45-2	159	164.5	164.57	173	3.17	480.86	13835	7.43	78.65
J90,45-3	169	174.5	174.23	182	3.32	446.18	14133	9.74	67.33
J90,45-4	145	150	150.5	155	2.62	411.41	13111	7.41	90.79
J90,45-5	192	196	196.43	204	2.93	491.26	15364	10.34	95.92
J90,45-6	190	197.5	196.9	206	4.27	450.49	15124	8.57	79.25
J90,45-7	148	153	153.43	157	1.92	406.25	14191	8.82	82.72
J90,45-8	172	179	178.87	185	3.37	440.6	14672	7.5	77.32
J90,45-9	168	174	173.8	180	2.81	502.29	14911	6.33	75
J90,45-10	179	184	184.47	190	2.74	496.61	15511	9.15	103.41
J90,46-1	110	112	112.13	115	1.57	330	12034	5.77	6.8
J90,46-2	98	101	100.87	104	1.74	285.11	4651	0	0
J90,46-3	115	118	117.77	119	1.1	276.52	12816	1.77	2.68
J90,46-4	99	102	102.13	106	1.96	304.86	12154	6.45	12.5
J90,46-5	91	95	95.47	98	1.72	288.56	10201	0	0
J90,46-6	86	88	88.07	91	1.14	281.08	12339	3.61	3.61
J90,46-7	92	96	95.8	99	1.45	311.38	12933	3.37	3.37
J90,46-8	102	104	103.9	107	1.32	359.42	14255	6.25	17.24
J90,46-9	92	94	94	96	0.98	310.15	12634	3.37	24.32
J90,46-10	114	114	114	114	0	2.84	90	0	0
J90,47-1	82	82	82	82	0	1.91	90	0	0
J90,47-2	90	90	90	90	0	1.74	90	0	0
J90,47-3	102	103	103.2	106	1.21	190.23	92	0	0
J90,47-4	93	93	93	93	0	10.01	90	0	0
J90,47-5	93	93	93	93	0	6.02	90	0	0

J90,47-6	98	98	98	98	0	1.86	90	0	0
J90,47-7	94	94	94	94	0	5.56	90	0	0
J90,47-8	98	98	98	98	0	1.82	90	0	0
J90,47-9	86	86	86	86	0	2.63	90	0	0
J90,47-10	65	65.5	65.73	69	0.94	165.27	91	0	0
J90,48-1	83	83	83	83	0	1.66	90	0	0
J90,48-2	89	89	89	89	0	1.66	90	0	0
J90,48-3	86	86	86	86	0	1.65	90	0	0
J90,48-4	91	91	91	91	0	1.66	90	0	0
J90,48-5	75	75	75	75	0	1.65	90	0	0
J90,48-6	114	114	114	114	0	1.71	90	0	0
J90,48-7	103	103	103	103	0	1.65	90	0	0
J90,48-8	74	74	74	74	0	1.66	90	0	0
J90,48-9	89	89	89	89	0	1.68	90	0	0
J90,48-10	93	93	93	93	0	1.64	90	0	0
Total Average					**0.86**	**145.78**	**4400.43**	**2.06**	**13.09**

Table 16

The detailed results of 600 problems of J120 obtained by bi-EA

The results are obtained from 30 runs with up to 50,000 fitness evaluations for each.

Prob. No	Best	Median	Mean	Worst	STD	t	FE	LB_{OP}	LB_{CP}
J120,1-1	113	118.5	118.17	124	2.6	855.74	15690	7.62	14.14
J120,1-2	119	125	124.67	130	2.68	736.43	14803	9.17	38.37
J120,1-3	132	135.5	136.1	143	3.17	793.13	14805	5.6	60.98
J120,1-4	104	107	107.5	112	2.37	722.27	14955	7.22	31.65
J120,1-5	119	123.5	124	131	2.98	705.42	15094	6.25	26.6
J120,1-6	87	92	91.7	97	2.25	717.05	15195	3.57	33.85
J120,1-7	122	126.5	127	133	2.98	738.91	13821	4.27	24.49
J120,1-8	114	119	119.17	126	3.14	618.18	13637	4.59	34.12
J120,1-9	122	129.5	129.83	144	4.82	704.64	13593	8.93	37.08
J120,1-10	113	123.5	123.8	134	5.64	765.46	13591	4.63	26.97
J120,2-1	91	96	95.47	99	2.58	664.26	13912	4.6	30
J120,2-2	79	81	81.5	85	1.66	735.88	13772	5.33	8.22
J120,2-3	100	105	104.83	109	2.56	690.49	13908	8.7	28.21
J120,2-4	97	105	104.37	113	3.51	692.79	13598	2.11	10.23
J120,2-5	111	113	113.53	121	2.37	701.8	13592	7.77	21.98
J120,2-6	96	101.5	101.63	106	2.94	607.66	13681	4.35	28
J120,2-7	95	100	100.03	105	2.61	705.03	13592	5.56	13.1
J120,2-8	88	93.5	93.77	99	3.1	659.27	13594	6.02	14.29
J120,2-9	100	102.5	103.13	108	2.3	664.31	13782	6.38	8.7
J120,2-10	104	111	111.03	119	3.61	730.42	13593	8.33	31.65
J120,3-1	84	87	87.37	93	1.99	682.57	13684	5	6.33
J120,3-2	88	88	88.23	90	0.5	121.6	91	0	0
J120,3-3	100	100	101.07	105	1.55	293.92	90	0	0
J120,3-4	75	79	78.7	84	2.22	718	13592	5.63	5.63
J120,3-5	86	90	89.8	93	1.94	650.1	13592	2.38	6.17
J120,3-6	102	102	102.23	103	0.43	151.35	91	0	0
J120,3-7	93	96	95.27	97	1.66	527.35	595	0	0
J120,3-8	78	82	82.77	90	2.84	734.66	13599	1.3	1.3
J120,3-9	86	86	86	86	0	4.59	90	0	0
J120,3-10	103	103	103	103	0	12.04	90	0	0
J120,4-1	74	75	76.17	80	2.21	465.07	138	0	5.71
J120,4-2	107	107	107	107	0	4.15	90	0	0
J120,4-3	95	95	95.07	97	0.37	66.64	92	0	4.4
J120,4-4	75	77	77.13	80	1.53	552.74	461	0	0
J120,4-5	74	76	76.1	80	1.84	519.35	1041	0	0
J120,4-6	90	90	90.8	96	1.54	207.02	92	0	5.88
J120,4-7	82	85	84.77	88	1.36	637.43	13594	1.23	1.23
J120,4-8	90	90	90	90	0	4.62	90	0	0
J120,4-9	79	79	79.4	83	0.89	155.96	91	0	0
J120,4-10	77	77	77	77	0	4.27	90	0	0
J120,5-1	92	92	92	92	0	4.1	90	0	0

J120,5-2	80	80	80.1	81	0.31	83.07	90	0	0
J120,5-3	73	74	74.3	77	1.37	613.69	13599	1.39	1.39
J120,5-4	97	97	97	97	0	4.11	90	0	0
J120,5-5	77	77	77.53	84	1.55	118.19	90	0	0
J120,5-6	88	89	88.7	90	0.75	406.93	594	0	0
J120,5-7	84	84	84	84	0	5.3	91	0	0
J120,5-8	78	78	78.3	81	0.7	144.77	92	0	0
J120,5-9	106	106	106	106	0	20.19	92	0	0
J120,5-10	92	92	92	92	0	4.02	90	0	0
J120,6-1	166	174	173.2	181	3.46	1190.39	15141	15.28	121.33
J120,6-2	153	162	161.5	168	3.88	1011.45	14659	13.33	112.5
J120,6-3	153	161	160.27	166	3.5	1087.17	17038	13.33	80
J120,6-4	177	184	185.1	194	4.67	1019.88	17453	14.94	80.61
J120,6-5	142	148	147.47	152	2.61	1071.32	17647	13.6	97.22
J120,6-6	171	185	185.2	192	4.27	1028.31	15183	10.32	140.85
J120,6-7	191	199	198.83	207	4.04	1009.98	13892	13.69	70.54
J120,6-8	171	179	179.7	187	4.69	921.76	12035	16.33	66.02
J120,6-9	186	191	191.47	197	2.86	899.36	15181	15.53	113.79
J120,6-10	196	202.5	202.63	207	2.87	1187.49	14611	13.95	110.75
J120,7-1	114	120	119.7	124	2.51	872.48	13415	11.76	50
J120,7-2	131	135.5	135.53	142	2.65	867.01	12393	14.91	33.67
J120,7-3	113	117.5	117.63	121	2.46	896.75	13594	13	25.56
J120,7-4	122	129	128.57	133	2.73	988.43	13534	8.93	43.53
J120,7-5	148	154	153.83	161	3.32	1061.03	13654	12.98	70.11
J120,7-6	138	145	145.3	151	2.89	976.29	13593	11.29	43.75
J120,7-7	132	137	136.5	141	2.7	899.72	13951	11.86	41.94
J120,7-8	106	112	111.8	117	2.58	850.83	13294	9.28	63.08
J120,7-9	101	104	104.67	110	1.95	872.43	12035	13.48	27.85
J120,7-10	128	134	134.17	140	2.95	989.03	13353	8.47	52.38
J120,8-1	101	108	107.67	116	3.13	760.22	12034	6.32	6.32
J120,8-2	112	121.5	121.53	129	3.12	808.71	12035	8.74	24.44
J120,8-3	105	110	109.9	115	2.2	784.25	12933	10.53	19.32
J120,8-4	106	109.5	109.67	118	2.86	867.53	13174	12.77	19.1
J120,8-5	114	117.5	118.17	123	2.32	822.94	14135	9.62	25.27
J120,8-6	96	100	99.9	106	2.32	820.63	12879	12.94	17.07
J120,8-7	96	99.5	99.5	103	1.78	890.63	12990	10.34	10.34
J120,8-8	94	97	97.07	102	1.8	815.05	12031	8.05	8.05
J120,8-9	104	107	107.07	111	1.86	812.07	13114	10.64	25.3
J120,8-10	102	105	104.63	109	1.77	779.09	12759	9.68	22.89
J120,9-1	91	93	93.23	97	1.57	930.94	13033	3.41	3.41
J120,9-2	94	95	95.27	98	1.11	337.39	372	0	0
J120,9-3	88	90	90.8	94	1.54	526.79	11455	1.15	1.15
J120,9-4	93	96	95.97	100	1.94	535.65	11133	6.9	16.25
J120,9-5	114	114	114	114	0	4.84	156	0	0
J120,9-6	103	108	107.63	112	2.19	561.36	12715	5.1	5.1
J120,9-7	80	83	82.73	85	1.28	479.2	8017	0	0
J120,9-8	82	84.5	84.3	86	1.24	507.06	10511	2.5	2.5
J120,9-9	92	94	94.07	97	1.34	527.2	11456	5.75	5.75
J120,9-10	85	88	88.23	91	1.94	563.94	12173	1.19	1.19
J120,10-1	111	111	111	111	0	14.87	287	0	0
J120,10-2	91	91	91	91	0	10.02	287	0	0
J120,10-3	100	102	102.03	105	1.69	652.99	13414	1.01	1.01
J120,10-4	96	100	99.77	105	2.01	604.02	12843	1.05	1.05

J120,10-5	97	97	97	97	0	20.81	96	0	0
J120,10-6	92	92	92	92	0	3.17	96	0	0
J120,10-7	81	83	83.2	86	1.52	561.91	14373	2.53	2.53
J120,10-8	114	116	116.07	120	1.8	428.39	1707	0	0
J120,10-9	77	77	77	77	0	45.56	97	0	0
J120,10-10	66	66	66.73	69	0.94	308.53	738	0	0
J120,11-1	194	199.5	199.27	205	3.34	1036.82	13890	12.14	115.56
J120,11-2	178	183.5	183.6	189	3.22	837.02	14194	12.66	128.21
J120,11-3	233	237	237.63	244	3.51	1004.93	13354	14.78	150.54
J120,11-4	224	233	232.4	240	4	996.39	13594	14.29	133.33
J120,11-5	242	249	249.33	258	4.42	829.05	12754	14.69	149.48
J120,11-6	239	249	249.6	256	3.8	845.72	12219	12.74	162.64
J120,11-7	185	190.5	190.13	198	3.35	915.22	13054	13.5	125.61
J120,11-8	182	188	187.63	192	2.28	854.68	14255	12.35	91.58
J120,11-9	194	201	200.73	210	3.77	855.87	16530	12.14	155.26
J120,11-10	205	211.5	212.53	220	3.89	1013.41	14258	13.26	127.78
J120,12-1	153	159	158.77	164	2.47	884.9	12031	10.87	62.77
J120,12-2	129	132	132.2	137	2.3	742.62	12574	10.26	79.17
J120,12-3	149	153	153.47	162	2.83	743.98	12817	9.56	77.38
J120,12-4	136	143	142.8	149	2.75	625.85	15005	8.8	44.68
J120,12-5	178	187.5	187.5	194	3.96	790.41	13893	9.88	81.63
J120,12-6	134	139.5	139.33	144	2.63	663.71	14075	10.74	63.41
J120,12-7	131	134.5	134.93	142	2.46	647.36	12333	9.17	57.83
J120,12-8	130	135	135.1	140	2.07	661.16	15571	9.24	80.56
J120,12-9	117	119.5	120.2	124	1.92	653.85	13950	11.43	51.95
J120,12-10	154	159	158.93	165	2.15	675.26	13118	7.69	83.33
J120,13-1	140	143	143.07	149	2.15	1158.57	15096	10.24	15.7
J120,13-2	94	96.5	96.73	99	1.36	891.47	14200	5.62	30.56
J120,13-3	131	136	136.67	142	2.54	930.08	13954	11.02	23.58
J120,13-4	120	126	125.4	130	2.55	955.2	14074	7.14	36.36
J120,13-5	100	103	103.03	107	1.63	854.57	12273	9.89	26.58
J120,13-6	111	113	113.57	120	2.1	969.62	13053	12.12	29.07
J120,13-7	119	124	123.9	128	2.51	994.28	13351	9.17	13.33
J120,13-8	101	104.5	104.83	114	2.87	1038.58	12035	7.45	16.09
J120,13-9	94	97	96.8	100	1.85	982.67	13414	10.59	22.08
J120,13-10	101	105	105.67	110	2.32	972.44	13655	9.78	29.49
J120,14-1	93	96	96.2	99	1.71	836.09	12033	9.41	14.81
J120,14-2	101	105	104.83	108	1.91	909.5	12754	8.6	17.44
J120,14-3	93	96	95.8	99	1.67	806.2	12092	5.68	5.68
J120,14-4	98	100	100.53	103	1.46	983.01	12035	11.36	15.29
J120,14-5	108	111	111.03	116	1.69	934.57	12879	11.34	20
J120,14-6	95	97	97.73	101	1.62	835.82	13175	4.4	4.4
J120,14-7	96	100.5	100.07	103	1.78	938.38	12753	5.49	7.87
J120,14-8	119	122	122.3	127	1.9	1075.99	13230	6.25	17.82
J120,14-9	102	105	104.8	107	1.19	910.25	13837	0.99	0.99
J120,14-10	87	89	88.97	92	1.27	912.28	14375	7.41	8.75
J120,15-1	81	81	81.47	84	0.9	318.06	90	0	0
J120,15-2	77	80.5	80.83	84	1.66	805.29	12454	2.67	2.67
J120,15-3	89	93	92.8	97	1.58	811.87	12871	2.3	2.3
J120,15-4	82	85	84.9	88	1.42	736.48	2140	0	0
J120,15-5	87	87	87	87	0	4.45	90	0	0
J120,15-6	97	97	97	97	0	7.03	91	0	0
J120,15-7	75	75	75	75	0	54.44	91	0	0

J120,15-8	126	126	126	126	0	4.5	90	0	0
J120,15-9	109	109	109	109	0	4.48	90	0	0
J120,15-10	91	94	94.13	97	1.46	776.57	91	0	0
J120,16-1	214	223	223.47	235	4.63	1671.18	14945	9.18	201.41
J120,16-2	257	265.5	265.27	275	4.59	1597.08	14074	10.78	202.35
J120,16-3	262	269	269.2	278	4.14	1703	15665	11.97	170.1
J120,16-4	222	228	228.17	234	3.26	1489.28	15183	11	177.5
J120,16-5	223	229.5	229.37	236	2.94	1457.53	15187	11.5	142.39
J120,16-6	226	233	233.3	239	3.49	1557.52	14941	9.71	186.08
J120,16-7	203	211	210.87	216	2.73	1428.97	15574	9.14	125.56
J120,16-8	217	223	223.1	229	3.32	1639.89	15785	11.28	181.82
J120,16-9	227	235	234.97	246	4.44	1444.88	12633	10.73	157.95
J120,16-10	241	244.5	245.33	251	3.35	1621.28	16178	13.15	145.92
J120,17-1	153	156	156.6	160	1.85	1237.36	13175	9.29	75.86
J120,17-2	134	137	137.07	145	2.23	1095.21	12513	8.94	81.08
J120,17-3	117	120	119.93	122	1.08	980.13	13173	8.33	62.5
J120,17-4	130	134	134.57	139	2.33	1052.06	15098	8.33	42.86
J120,17-5	144	147.5	147.57	153	2.11	1172.55	15005	11.63	60
J120,17-6	146	149.5	149.5	153	1.74	1111.34	13897	7.35	114.71
J120,17-7	160	163	163.3	168	1.99	1258.69	15213	9.59	63.27
J120,17-8	138	142	142.1	145	1.81	1187.89	13233	8.66	89.04
J120,17-9	147	151	151.4	156	2.46	1294.58	12693	9.7	81.48
J120,17-10	145	148	148.5	153	1.81	1172.99	12511	8.21	64.77
J120,18-1	145	149	148.97	153	1.73	1299.98	17574	5.07	43.56
J120,18-2	126	131	130.8	136	2.77	1108.52	14013	8.62	13.51
J120,18-3	105	108	108.17	112	1.68	955.3	12753	3.96	45.83
J120,18-4	108	111	111.27	115	2.02	987.56	12695	6.93	40.26
J120,18-5	128	130	130.53	136	1.93	1022.23	13354	8.47	45.45
J120,18-6	145	149	149.13	154	2.24	1195.41	12039	8.21	34.26
J120,18-7	127	131	130.53	134	1.57	1020.32	13594	7.63	42.7
J120,18-8	113	117	116.8	120	1.67	1101.6	13830	6.6	28.41
J120,18-9	98	100	100.3	103	1.32	883.55	13175	7.69	28.95
J120,18-10	108	111	111.3	115	2.07	918.43	13534	10.2	28.57
J120,19-1	89	95	94.17	98	2	940.58	12993	1.14	1.14
J120,19-2	89	92	91.97	95	1.5	922.53	12093	7.23	9.88
J120,19-3	90	92	92.2	95	1.27	928.29	12514	5.88	25
J120,19-4	116	119	119.47	124	1.81	962.42	12213	9.43	26.09
J120,19-5	111	116	115.7	120	2.17	1013.32	14500	7.77	30.59
J120,19-6	97	100.5	100.5	104	2.19	841.47	12033	7.78	21.25
J120,19-7	96	100	100.17	105	2.09	852.34	12093	3.23	3.23
J120,19-8	100	104	103.97	108	1.94	852.11	12154	7.53	7.53
J120,19-9	94	96	96.13	99	1.31	953.15	12031	5.62	25.33
J120,19-10	90	93	92.6	95	1.19	936.58	13234	2.27	2.27
J120,20-1	95	99	98.53	102	1.68	737.71	10729	6.74	6.74
J120,20-2	99	101	101.03	104	1.27	530.89	289	0	0
J120,20-3	82	85	84.67	89	1.45	587.88	9473	6.49	10.81
J120,20-4	89	89	89	89	0	42.45	169	0	0
J120,20-5	71	74	74.07	77	1.36	516.1	8475	2.9	2.9
J120,20-6	80	80	80	80	0	8.21	169	0	0
J120,20-7	81	81	81	81	0	22.27	169	0	0
J120,20-8	113	115.5	115.67	119	1.47	615.95	9746	5.61	5.61
J120,20-9	80	80	80	80	0	44.16	169	0	0
J120,20-10	81	84	84.07	88	1.44	591.25	6672	0	0

J120,21-1	121	128	127.97	133	2.7	562.38	9132	6.14	23.47
J120,21-2	122	127	126.73	131	2.32	479.84	8629	4.27	40.23
J120,21-3	146	153	153.27	159	2.56	556.26	9204	2.1	31.53
J120,21-4	141	146.5	145.83	150	2.35	444.91	8575	4.44	27.03
J120,21-5	114	121	120.5	127	3.27	535.45	8520	3.64	20
J120,21-6	119	123	123.53	129	2.32	497.81	8803	9.17	22.68
J120,21-7	118	122	122.13	127	2.13	561.15	9078	6.31	49.37
J120,21-8	135	142	141.7	151	3.29	456.6	8460	6.3	11.57
J120,21-9	109	113	113.43	119	2.66	533.17	8644	6.86	26.74
J120,21-10	109	114	113.8	120	2.55	573.48	8797	6.86	25.29
J120,22-1	105	111.5	111.37	116	2.74	528.06	8688	3.96	26.51
J120,22-2	110	115	114.5	120	2.29	522.37	8571	2.8	5.77
J120,22-3	102	105.5	105.67	110	1.9	551.55	8908	6.25	22.89
J120,22-4	92	95	95.2	99	1.75	498.38	8919	2.22	16.46
J120,22-5	97	100	100.37	104	1.88	560.75	8629	4.3	4.3
J120,22-6	107	111	110.63	114	1.99	574.82	8748	3.88	15.05
J120,22-7	133	133	133.07	134	0.25	56.32	169	0	7.26
J120,22-8	108	111	111.07	114	1.74	505.93	8635	4.85	16.13
J120,22-9	112	116	116.13	119	2.05	514.9	8855	2.75	23.08
J120,22-10	79	82	81.53	84	1.55	469.33	3425	0	11.27
J120,23-1	107	107	107	107	0	11.11	169	0	0
J120,23-2	116	116	116.2	119	0.66	63.76	169	0	0
J120,23-3	99	99	99	99	0	9.86	169	0	0
J120,23-4	106	106.5	107.4	112	1.96	326.89	1745	0	2.91
J120,23-5	100	100	100.5	103	0.82	423.22	8461	1.01	4.17
J120,23-6	108	110.5	110.5	114	1.72	456.53	8464	1.89	4.85
J120,23-7	104	104	104.8	107	1.03	229.64	170	0	0
J120,23-8	101	101	101	101	0	45.95	170	0	1
J120,23-9	107	110	110.27	114	1.64	453.8	3204	0	1.9
J120,23-10	100	100	100.03	101	0.18	118.16	171	0	0
J120,24-1	93	93	93.2	94	0.41	105.79	170	0	1.09
J120,24-2	91	94	93.77	97	1.61	398.46	3429	0	0
J120,24-3	89	89	89.2	91	0.48	150.15	285	0	0
J120,24-4	101	101	101.37	103	0.61	227.98	509	0	0
J120,24-5	86	86	86.6	89	1.07	190.15	620	0	0
J120,24-6	95	95	95.13	97	0.43	108.94	171	0	0
J120,24-7	112	112	112	112	0	38.19	169	0	2.75
J120,24-8	104	104	104.03	105	0.18	31.13	170	0	0
J120,24-9	82	83	83.07	85	0.87	330.49	734	0	7.89
J120,24-10	91	91	91.2	92	0.41	129.27	170	0	0
J120,25-1	82	85	84.4	86	1.07	437	846	0	0
J120,25-2	108	108	108	108	0	8.33	169	0	0
J120,25-3	100	100	100	100	0	7.8	169	0	0
J120,25-4	117	117	117	117	0	7.84	169	0	0
J120,25-5	100	100	100	100	0	7.91	169	0	0
J120,25-6	92	92	92.27	93	0.45	188.6	171	0	1.1
J120,25-7	92	95.5	94.93	97	1.41	405.81	1521	0	1.1
J120,25-8	81	83	83.37	86	1.47	467.22	8464	1.25	6.58
J120,25-9	94	94	94	94	0	7.87	169	0	0
J120,25-10	92	92	92	92	0	8.7	169	0	0
J120,26-1	194	204	203.5	210	4.13	747.69	9195	14.79	102.08
J120,26-2	193	199.5	200.13	209	4.11	730.68	7667	14.2	98.97
J120,26-3	190	197.5	197.5	203	3.5	664	7334	13.77	104.3

J120,26-4	191	202.5	202.53	215	4.7	669.93	7534	11.05	76.85
J120,26-5	177	183.5	183.7	192	3.84	680.01	7241	15.69	118.52
J120,26-6	215	223.5	223.33	237	4.56	658.24	7200	14.97	76.23
J120,26-7	181	191	190.2	196	4.11	624.65	7245	15.29	118.07
J120,26-8	198	207	205.8	214	4.07	699.4	8801	12.5	88.57
J120,26-9	201	208	207.77	218	4.12	680.04	7196	16.86	62.1
J120,26-10	212	222	221.97	231	4.42	534.26	7113	15.85	73.77
J120,27-1	118	122	122.17	125	1.88	610.88	8699	9.26	53.25
J120,27-2	128	133	133.07	139	2.94	553.07	7439	11.3	47.13
J120,27-3	155	159	160	167	2.99	603.28	7574	7.64	58.16
J120,27-4	118	122	122.13	129	2.46	580.95	7994	9.26	38.82
J120,27-5	119	126	125.73	130	2.63	612.9	7431	7.21	33.71
J120,27-6	164	169	169.13	177	3.13	713.51	8424	13.1	69.07
J120,27-7	142	148	147.63	154	2.43	511.35	7101	13.6	42
J120,27-8	156	161	161.63	167	3.25	726.14	8326	11.43	44.44
J120,27-9	142	148	147.63	157	2.74	640.13	8088	10.94	49.47
J120,27-10	128	135.5	135.1	142	3.29	576.16	7099	11.3	21.9
J120,28-1	117	122	122.13	128	2.39	581.38	7104	8.33	19.39
J120,28-2	117	120	120.7	128	2.89	573.9	7427	6.36	6.36
J120,28-3	106	110	110.17	114	2.04	516.84	7710	4.95	4.95
J120,28-4	128	130	130.03	135	1.63	563.41	7431	14.29	24.27
J120,28-5	102	104	104.23	109	1.76	481.77	1553	0	0
J120,28-6	112	115.5	115.57	120	1.99	548.15	7109	8.74	8.74
J120,28-7	118	122	122.03	126	2.17	532.92	7335	8.26	21.65
J120,28-8	109	114	113.7	118	2.07	525.43	7573	10.1	25.29
J120,28-9	108	112	111.9	116	1.95	545.22	7621	10.2	16.13
J120,28-10	127	130	130.67	135	2.22	583.16	7666	9.48	16.51
J120,29-1	104	105	104.9	108	1.06	298.15	141	0	0
J120,29-2	91	93.5	93.53	96	1.33	436.4	1746	0	0
J120,29-3	104	106	106.17	109	1.34	541.2	8331	6.12	30
J120,29-4	86	89	88.83	92	1.49	448.92	7291	7.5	11.69
J120,29-5	106	109	109.3	114	2.02	539.43	8043	3.92	3.92
J120,29-6	96	100	99.8	103	1.58	577.12	7953	5.49	9.09
J120,29-7	97	97	97.33	102	0.99	161.04	141	0	0
J120,29-8	83	85.5	85.37	88	1.38	483.36	7995	3.75	3.75
J120,29-9	97	99	99.4	102	1.25	465.46	2033	0	0
J120,29-10	96	98	98.07	101	1.39	404.52	142	0	0
J120,30-1	102	105	104.8	109	2.19	561.46	938	0	0
J120,30-2	112	112	112	112	0	6.32	160	0	0
J120,30-3	108	108	108	108	0	11.82	160	0	0
J120,30-4	83	84	83.9	85	0.76	552.56	910	0	0
J120,30-5	89	93	92.77	96	1.79	732.35	14885	7.23	9.88
J120,30-6	79	81.5	81.3	84	1.56	485.58	848	0	0
J120,30-7	95	99	99.03	102	1.73	544.22	13258	2.15	2.15
J120,30-8	79	80.5	80.8	84	1.32	468.35	1111	0	0
J120,30-9	93	93	93.17	95	0.53	132.91	137	0	0
J120,30-10	86	87	87.83	91	1.51	509.11	3658	0	0
J120,31-1	225	231.5	231.8	239	4.3	941.05	15516	13.64	144.57
J120,31-2	219	225	225.87	235	4.05	961.61	14948	13.47	167.07
J120,31-3	199	206.5	207.53	216	3.38	842.09	13051	14.37	148.75
J120,31-4	251	259.5	260.57	273	5.14	907.13	12037	14.61	124.11
J120,31-5	230	239.5	239.13	247	4.86	902.1	13653	15	137.11
J120,31-6	209	219.5	219.8	231	5.35	984.34	12275	8.85	106.93

J120,31-7	230	238	238.83	246	4.23	1009.69	14970	11.65	109.09
J120,31-8	217	227	226.7	234	3.84	874.1	13898	13.02	135.87
J120,31-9	215	220	220.8	233	4.57	956.15	12098	13.76	117.17
J120,31-10	261	274	272.8	284	5.54	886.82	15065	14.98	190
J120,32-1	160	164	164.2	168	1.94	812.23	16684	8.84	61.62
J120,32-2	147	151.5	151.13	155	2.33	665.31	14740	12.21	47
J120,32-3	159	167	167.17	176	3.84	780.31	15306	9.66	57.43
J120,32-4	148	155	154.6	161	3.21	708.96	12453	8.82	37.04
J120,32-5	153	158	158.17	163	2.15	722.82	15690	10.87	54.55
J120,32-6	141	145	145	150	2.48	756.58	15841	10.16	50
J120,32-7	135	139	139.17	145	2.36	688.43	14193	10.66	29.81
J120,32-8	148	153.5	153.47	157	2.54	740.54	15453	9.63	82.72
J120,32-9	137	141	141.53	147	2.42	730.36	14791	7.87	59.3
J120,32-10	144	149.5	148.93	153	2.29	814.36	14739	9.92	58.24
J120,33-1	120	124	123.93	128	2.3	608.57	13233	12.15	21.21
J120,33-2	124	127	126.9	130	1.63	605.93	13774	9.73	34.78
J120,33-3	117	122	122.03	126	2.25	685.49	13533	9.35	32.95
J120,33-4	122	127	127.07	132	2.41	748.18	13355	8.93	25.77
J120,33-5	161	164.5	164.73	170	2.15	806.4	13953	13.38	49.07
J120,33-6	126	129	128.97	134	2.09	659.75	15424	9.57	9.57
J120,33-7	136	140	140.5	145	2.26	622.08	13590	10.57	46.24
J120,33-8	122	125	125.73	130	2.02	803.37	13114	9.91	35.56
J120,33-9	124	127	127.37	131	1.9	729.66	14490	8.77	21.57
J120,33-10	116	120.5	120.73	127	2.48	623.06	14320	9.43	31.82
J120,34-1	86	89.5	89.7	93	1.93	546.58	12633	10.26	19.44
J120,34-2	115	117.5	117.8	120	1.49	626.37	13779	9.52	21.05
J120,34-3	107	112	111.77	117	2.01	661.62	12936	4.9	13.83
J120,34-4	103	108	107.63	113	2.58	580.32	12038	8.42	9.57
J120,34-5	111	114	114	118	1.93	705.58	13653	7.77	12.12
J120,34-6	110	113	113.4	120	2.06	540.27	12753	10	10
J120,34-7	112	116	115.8	123	2.4	595.2	12934	6.67	6.67
J120,34-8	97	102	101.97	106	2.06	619.47	14014	8.99	19.75
J120,34-9	101	105	105.07	108	1.93	705.52	14314	6.32	21.69
J120,34-10	104	108	108.2	113	2.16	602.63	12939	2.97	2.97
J120,35-1	87	87	87.03	88	0.18	30.76	91	0	0
J120,35-2	122	126	126.2	131	2.5	645.74	13653	9.91	9.91
J120,35-3	82	84	84.27	88	1.26	561.54	12698	6.49	6.49
J120,35-4	108	112	112	117	2.48	580.73	12036	6.93	6.93
J120,35-5	101	105	104.7	109	2.02	571.9	12033	8.6	9.78
J120,35-6	86	88	88.53	93	2.03	559	2554	0	0
J120,35-7	99	99	99.23	102	0.68	94.41	90	0	0
J120,35-8	103	105	105.33	108	1.58	543.02	12033	1.98	1.98
J120,35-9	96	97	97.73	101	1.36	581.83	12572	5.49	5.49
J120,35-10	86	88	88.27	92	1.48	509.41	399	0	0
J120,36-1	233	238.5	238.73	245	3.58	1125.04	16681	10.95	137.76
J120,36-2	253	258	257.9	265	2.78	946.37	15364	13.45	184.27
J120,36-3	258	263.5	263.57	270	3.29	990.66	15184	12.66	183.52
J120,36-4	265	270.5	270.97	277	3.41	1105.41	15007	12.29	176.04
J120,36-5	253	263	263.27	271	4.24	1035.13	17163	10.48	166.32
J120,36-6	255	263.5	263	272	4.31	911.89	12155	13.33	152.48
J120,36-7	230	238	238.3	246	4.15	936.35	13594	10.58	123.3
J120,36-8	191	196	196.37	205	3.45	999.65	13959	11.05	130.12
J120,36-9	248	257	257.4	266	4.33	984.24	12095	12.22	138.46

J120,36-10	236	246	246.3	254	4.15	1109.34	16536	9.26	159.34
J120,37-1	159	165	164.27	170	2.69	819.54	13657	9.66	65.63
J120,37-2	156	160	160.03	164	1.92	828.86	14075	7.59	81.4
J120,37-3	155	158	158.23	165	2.46	969.63	13658	11.51	53.47
J120,37-4	178	182	182.03	185	2.17	977.06	12937	9.2	87.37
J120,37-5	226	231	231.73	239	3.1	893.95	15184	9.71	98.25
J120,37-6	182	186.5	186.4	191	2.51	861.08	13292	11.66	73.33
J120,37-7	175	182	181.73	188	2.88	990.76	13897	8.7	113.41
J120,37-8	196	203	202.97	210	2.83	1035.91	17526	10.11	68.97
J120,37-9	158	162	161.93	167	2.29	782.36	13834	11.27	92.68
J120,37-10	143	146	146.47	152	1.81	856.14	12873	8.33	78.75
J120,38-1	117	122	121.57	126	2.34	1069.22	13419	8.33	13.59
J120,38-2	136	139	139.13	143	1.61	1102.26	13174	9.68	47.83
J120,38-3	165	168.5	169	173	2.33	1128.99	12751	6.45	57.14
J120,38-4	152	156.5	156.43	162	2.64	1164.98	14070	8.57	31.03
J120,38-5	121	123	123.7	127	1.58	1061.86	12032	6.14	27.37
J120,38-6	131	136	135.63	140	1.9	1145.73	14314	7.38	37.89
J120,38-7	114	118	118.5	123	2.27	1029.08	12458	8.57	25.27
J120,38-8	136	139	138.73	141	1.41	1108.84	13474	8.8	38.78
J120,38-9	145	152	152.5	157	2.81	1131.65	12033	8.21	8.21
J120,38-10	151	153	153.1	156	1.42	1250.96	14611	7.86	57.29
J120,39-1	101	104.5	105.1	110	2.37	898.23	12158	6.32	6.32
J120,39-2	119	123	123.07	127	2.12	1117.64	12693	10.19	14.42
J120,39-3	121	125	124.57	128	1.72	947.37	13476	9.01	21
J120,39-4	104	107	106.83	109	1.23	1066.81	13418	6.12	36.84
J120,39-5	107	111	110.7	115	1.91	1001.45	12811	0.94	0.94
J120,39-6	100	103	102.93	106	1.64	916.9	12031	5.26	5.26
J120,39-7	110	112	112.5	116	1.61	1013.03	14015	5.77	17.02
J120,39-8	104	108	107.8	112	2.02	981.38	12156	8.33	11.83
J120,39-9	98	100	100.4	102	1	1020.79	12994	6.52	27.27
J120,39-10	120	123	123.63	129	2.16	999.61	13893	9.09	21.21
J120,40-1	85	87	87	89	1.08	910.11	12457	4.94	8.97
J120,40-2	95	98	97.73	99	1.11	878.54	12518	5.56	5.56
J120,40-3	93	96	95.73	100	2.02	864.33	12271	6.9	6.9
J120,40-4	112	112	112.17	116	0.75	189.88	91	0	0
J120,40-5	102	106	105.5	109	1.93	926.49	12331	0.99	0.99
J120,40-6	90	91	91	94	1.02	529.61	1595	0	0
J120,40-7	91	91	91.93	97	1.44	502.75	90	0	0
J120,40-8	98	99	99.43	103	1.17	847.06	12033	1.03	1.03
J120,40-9	117	120	119.63	122	1.07	890.42	13235	0	0
J120,40-10	96	96	96.03	97	0.18	52.3	90	0	0
J120,41-1	130	138.5	138.03	144	3.29	769.32	12693	2.36	26.21
J120,41-2	146	150	150.77	157	3.51	846.92	12635	3.55	47.47
J120,41-3	145	152	152.7	162	4.21	827.88	12691	2.84	21.85
J120,41-4	121	128	127.97	134	3.03	733.88	12573	4.31	39.08
J120,41-5	139	144	144.1	151	2.38	752.76	12213	0.72	20.87
J120,41-6	119	124	124.33	129	2.29	803	12033	5.31	30.77
J120,41-7	115	120.5	120.63	127	3.13	824.17	12931	5.5	26.37
J120,41-8	141	146.5	146.73	154	2.96	825.65	12153	2.17	28.18
J120,41-9	127	132	132.27	138	2.53	776.65	12393	4.96	33.68
J120,41-10	143	150	150.37	157	3.66	857.07	13113	5.15	18.18
J120,42-1	114	118	118.1	124	2.59	491.16	12484	5.56	25.27
J120,42-2	126	130	129.87	134	2.15	368.59	129	0	0

J120,42-3	107	112	112.2	117	2.47	443.68	12090	0.94	7
J120,42-4	110	116	115.33	120	3.1	417.2	12211	5.77	7.84
J120,42-5	126	133	132.97	139	3.03	443.68	12007	5	16.67
J120,42-6	124	129	129.3	134	2.78	473.33	12329	4.2	40.91
J120,42-7	124	126	126.47	132	2.36	413.93	12284	0.81	5.08
J120,42-8	121	124	124.3	129	1.95	493	12366	7.08	19.8
J120,42-9	106	110	110.4	118	2.37	392.59	12369	1.92	10.42
J120,42-10	123	128	127.9	132	2.34	445.14	12045	4.24	20.59
J120,43-1	105	107	107.53	111	1.41	1083.05	12490	0	5
J120,43-2	120	120	121.2	126	1.73	633.5	148	0	4.35
J120,43-3	97	99	99.37	106	1.9	1093.98	18857	2.11	2.11
J120,43-4	107	111.5	111.8	117	2.72	762.63	12034	1.9	7
J120,43-5	107	111	111.3	115	1.74	714.1	12033	1.9	9.18
J120,43-6	102	105.5	105.57	114	2.74	752.17	12033	4.08	20
J120,43-7	123	128	128.87	135	3.4	771.89	12033	0.82	5.13
J120,43-8	115	115	115.43	119	0.97	210.56	91	0	0
J120,43-9	107	109	109.67	113	1.42	680.01	12033	1.9	5.94
J120,43-10	113	115	114.7	117	1.42	505.37	93	0	1.8
J120,44-1	100	100	100	100	0	21.81	90	0	0
J120,44-2	112	114	113.7	117	1.51	504.61	2199	0	9.8
J120,44-3	107	107	107	107	0	5.86	90	0	0
J120,44-4	96	98.5	98.47	101	1.61	669.81	12031	1.05	3.23
J120,44-5	99	100	100.23	103	1.04	645.23	12031	1.02	5.32
J120,44-6	106	106	106.43	108	0.68	338.37	94	0	0
J120,44-7	98	98	98.5	103	1.36	188.07	93	0	0
J120,44-8	109	114	113.53	116	2.24	671.69	12032	0.93	3.81
J120,44-9	91	92	91.93	94	0.98	384.62	94	0	0
J120,44-10	100	103	102.47	108	1.76	787.47	12034	2.04	2.04
J120,45-1	108	108	108	108	0	9.7	302	0	0
J120,45-2	91	91	91	91	0	54.41	250	0	0
J120,45-3	98	98	98	98	0	13.56	248	0	0
J120,45-4	103	103	103.13	104	0.35	161.13	185	0	0
J120,45-5	116	116	116	116	0	61.48	189	0	1.75
J120,45-6	125	125	125	125	0	6.5	185	0	0
J120,45-7	103	103	103	103	0	9.9	184	0	0
J120,45-8	103	103	103.53	106	0.86	347.88	173	0	0
J120,45-9	114	114	114	114	0	5.24	169	0	0
J120,45-10	99	99	99	99	0	10.16	170	0	0
J120,46-1	215	225	224.67	232	3.99	901.95	17404	14.36	80.67
J120,46-2	224	233	233.13	247	5.41	766.4	15128	13.13	91.45
J120,46-3	196	203.5	203.7	215	4.36	691.38	12870	12	88.46
J120,46-4	192	197.5	198.23	208	4.83	699.9	13893	12.94	113.33
J120,46-5	167	178	177.13	185	4.09	744.34	15782	12.08	72.16
J120,46-6	199	205.5	206.8	219	4.94	779.84	16596	11.8	105.15
J120,46-7	190	198.5	198.53	207	4.97	748.64	13114	11.76	79.25
J120,46-8	200	210	209.7	222	5.18	713.21	14614	12.99	108.33
J120,46-9	189	200	199.8	209	4.94	774.14	12753	13.86	98.95
J120,46-10	210	219	219.4	231	5.31	774.49	12511	11.7	100
J120,47-1	155	164	163.2	169	3.02	749.4	13293	13.14	44.86
J120,47-2	142	149.5	149.97	156	3.71	651.31	13651	7.58	32.71
J120,47-3	140	143	143.03	146	1.56	647.54	13355	12	41.41
J120,47-4	145	151	151.3	158	3.1	677.84	15301	9.85	36.79
J120,47-5	145	150	150.83	160	3.99	714.4	13895	14.17	36.79

J120,47-6	154	160	159.97	166	2.53	692.02	14434	12.41	54
J120,47-7	133	137.5	138.2	144	2.72	724.49	15993	12.71	54.65
J120,47-8	151	157	157.6	166	3.93	722.8	13479	13.53	86.42
J120,47-9	159	166.5	165.77	171	3.62	694.66	13415	10.42	52.88
J120,47-10	149	153	153.07	158	2.15	649.44	13955	12.88	46.08
J120,48-1	112	116	116.43	121	2.37	606.13	12573	12	17.89
J120,48-2	120	124.5	124.27	128	2.02	636.93	13533	6.19	36.36
J120,48-3	124	128	128.13	134	2.65	680.71	14794	10.71	19.23
J120,48-4	138	143	142.97	148	2.67	698.09	13415	8.66	31.43
J120,48-5	118	121	121.67	126	2.01	617.12	12633	7.27	26.88
J120,48-6	115	118	117.57	120	1.38	543.88	12874	9.52	22.34
J120,48-7	118	121	120.9	124	1.77	629.54	14071	10.28	15.69
J120,48-8	123	125	125.4	131	2.33	654.13	13234	6.03	19.42
J120,48-9	122	127.5	127.83	135	3.1	665.14	13292	7.96	20.79
J120,48-10	121	124.5	124.83	129	2.35	656.75	14134	9.01	18.63
J120,49-1	98	100	99.83	103	1.34	655.04	13366	2.08	2.08
J120,49-2	116	120	120.57	126	2.31	737.67	14890	6.42	16
J120,49-3	103	107	106.67	109	1.63	618.39	13829	7.29	10.75
J120,49-4	103	105	105.6	110	1.75	631.98	13566	7.29	14.44
J120,49-5	95	97	97.07	100	1.64	678.35	14030	6.74	15.85
J120,49-6	128	128	128.17	129	0.38	169.45	99	0	0
J120,49-7	105	108	107.8	111	1.97	683.26	13564	6.06	8.25
J120,49-8	121	125	124.77	129	2.05	681.96	14488	7.08	8.04
J120,49-9	102	105	105.37	112	1.87	598.04	13235	5.15	5.15
J120,49-10	102	106	105.67	109	1.86	660.04	13233	5.15	17.24
J120,50-1	116	116	116.43	118	0.63	462.7	166	0	0
J120,50-2	117	119	119.57	124	1.57	892.29	15018	4.46	4.46
J120,50-3	111	112.5	112.87	116	1.43	534.7	1161	0	0
J120,50-4	104	108	107.73	111	1.53	576.48	14628	4	10.64
J120,50-5	105	106	106.47	109	1.04	596.47	15354	5	5
J120,50-6	102	102	102.63	105	0.89	308.53	697	0	0
J120,50-7	137	137	137	137	0	3.19	99	0	0
J120,50-8	112	112	112.03	113	0.18	54.13	99	0	0
J120,50-9	101	102.5	102.9	106	1.71	420.24	2150	0	0
J120,50-10	111	114	113.73	116	1.39	539.49	13502	7.77	7.77
J120,51-1	235	245	244.83	258	5.23	1482.6	14313	14.08	135
J120,51-2	249	259.5	258.33	268	4.51	1550.35	16718	12.67	170.65
J120,51-3	246	252.5	251.93	259	3.93	1795.69	13776	11.82	164.52
J120,51-4	244	253	252.27	259	3.77	1529.63	16474	15.09	168.13
J120,51-5	263	276	275.7	286	5.57	1434.27	15604	14.35	157.84
J120,51-6	248	257	256.73	265	4.83	1669	17224	15.35	138.46
J120,51-7	243	251	250.3	259	3.65	1387.52	13953	14.62	167.03
J120,51-8	239	247	246.83	256	4.02	1524.21	12635	16.02	181.18
J120,51-9	242	249	249.83	261	5.15	1512.21	16983	14.69	132.69
J120,51-10	262	272.5	271.93	282	5.11	1576.76	16293	15.42	142.59
J120,52-1	197	203	203.23	209	2.99	1517.92	15454	11.93	74.34
J120,52-2	202	207.5	207.67	212	2.52	1557.2	15335	10.38	81.98
J120,52-3	148	152	152.37	158	2.47	1294.72	15213	10.45	45.1
J120,52-4	187	193	192.93	202	3.38	1467.84	12340	10	65.49
J120,52-5	185	194	192.87	200	3.99	1308.68	12755	10.12	63.72
J120,52-6	219	227	226.57	237	4.07	1573.3	15153	11.73	85.59
J120,52-7	161	168	168.13	175	3.66	1340.78	14734	8.05	69.47
J120,52-8	174	181	181	189	3.61	1369.09	16323	10.13	52.63

J120,52-9	165	171	170.77	178	3.11	1319.57	15454	10	87.5
J120,52-10	161	166	165.93	172	2.63	1347.17	13173	11.81	76.92
J120,53-1	159	164	164.2	169	2.63	1499.93	15611	10.42	29.27
J120,53-2	126	129	128.97	134	1.83	1280.45	15123	9.57	16.67
J120,53-3	121	126	125.9	131	2.11	1117.76	13114	8.04	17.48
J120,53-4	160	165	165.07	170	2.53	1138.83	12872	10.34	36.75
J120,53-5	121	127	126.9	131	2.4	1112.97	12995	7.08	26.04
J120,53-6	116	120	119.7	122	1.64	1156.61	12513	9.43	46.84
J120,53-7	127	132	131.97	138	3.02	1147.11	15033	6.72	9.48
J120,53-8	152	157	156.83	165	2.93	1225.3	13352	9.35	47.57
J120,53-9	188	194	195.3	204	3.87	1405.86	13594	13.94	44.62
J120,53-10	143	148	148.3	154	2.71	1148.23	14074	8.33	28.83
J120,54-1	113	117	116.63	121	1.85	1078.86	15395	7.62	15.31
J120,54-2	134	135	135.3	139	1.73	537.18	3100	0	0
J120,54-3	117	121.5	121.8	126	2.12	1024.39	12816	5.41	5.41
J120,54-4	127	134	133.8	138	2.22	1021.22	13593	5.83	6.72
J120,54-5	117	120	120.1	125	2.02	1033.27	12633	7.34	27.17
J120,54-6	117	121	120.67	124	1.71	923.38	12751	7.34	17
J120,54-7	123	126	126.73	130	2.05	940.21	13653	10.81	25.51
J120,54-8	110	113.5	113.83	118	2.21	1027.53	13834	7.84	23.6
J120,54-9	115	121	120.37	124	2.16	1092.61	14735	7.48	21.05
J120,54-10	115	118	118.47	123	1.85	994.48	13891	6.48	15
J120,55-1	108	110	110.1	114	1.6	635.85	12164	8	9.09
J120,55-2	83	84.5	84.67	87	1.18	416.78	1248	0	0
J120,55-3	126	126	126.17	128	0.46	130.84	122	0	0
J120,55-4	95	98	98.07	101	1.66	521.69	12328	5.56	5.56
J120,55-5	106	108	108.67	112	1.47	496.65	5691	0	0
J120,55-6	108	110	110.17	114	1.46	543.2	12486	8	10.2
J120,55-7	105	107	106.6	110	1.25	412.75	121	0	0
J120,55-8	107	109	109.23	113	1.63	614.75	13285	5.94	5.94
J120,55-9	97	99.5	99.67	103	1.45	528.06	12605	3.19	3.19
J120,55-10	101	103	103.8	108	1.52	472.21	12565	1	1
J120,56-1	261	273	272.5	279	4	707.48	10869	10.13	174.74
J120,56-2	231	238	238.03	247	3.44	645.94	10809	13.24	165.52
J120,56-3	270	276	276.97	289	4.16	651.4	9218	12.03	190.32
J120,56-4	249	255	255.23	262	3.04	681.27	10325	12.16	189.53
J120,56-5	314	323	323.83	339	5.86	779.33	10325	12.14	168.38
J120,56-6	238	249	247.5	256	4.67	688.07	9003	11.21	150.53
J120,56-7	309	324	323.13	334	4.98	796.48	11071	9.19	164.1
J120,56-8	324	334.5	332.47	345	5.42	773.3	9548	12.11	227.27
J120,56-9	318	331.5	332.3	344	6.1	749.86	9634	10.42	214.85
J120,56-10	294	300.5	301.2	310	4.48	675.18	10111	13.51	188.24
J120,57-1	203	210	210.43	217	3.32	743.89	10089	9.73	78.07
J120,57-2	176	181.5	181.27	189	2.9	707.87	9875	9.32	69.23
J120,57-3	200	209.5	209.33	218	3.5	680.71	9545	8.7	73.91
J120,57-4	214	225.5	224.8	231	3.78	713.92	10145	7	78.33
J120,57-5	195	201	201.37	208	2.93	682.77	9008	8.94	103.13
J120,57-6	205	213	212.8	220	3.84	711.85	10292	8.47	107.07
J120,57-7	182	190	189.23	193	3.06	624.52	10655	9.64	58.26
J120,57-8	178	183.5	182.77	188	3.14	594.3	9634	9.88	106.98
J120,57-9	184	191	190.63	196	3.01	706	9365	10.18	70.37
J120,57-10	184	188.5	188.7	194	2.61	583.25	10357	10.18	52.07
J120,58-1	155	159	158.77	162	2.24	528.62	9096	9.93	28.1

J120,58-2	137	141	140.4	146	2.27	552.1	9841	8.73	29.25
J120,58-3	129	132	132.17	135	1.23	522.84	9331	7.5	41.76
J120,58-4	157	161.5	161.5	165	1.74	578.6	9485	8.28	60.2
J120,58-5	129	132.5	132.43	137	1.79	530.9	9787	7.5	27.72
J120,58-6	153	159	159.17	169	3.15	526.85	9695	9.29	48.54
J120,58-7	161	165.5	165.63	169	1.85	546.45	9840	9.52	41.23
J120,58-8	146	149	149.1	154	2.12	532.06	9331	10.61	47.47
J120,58-9	141	144	144	147	1.93	600.97	11074	8.46	53.26
J120,58-10	143	147.5	147.53	153	2.5	563.56	9753	9.16	45.92
J120,59-1	120	122	122.17	124	1.29	431.55	9907	5.26	17.65
J120,59-2	114	116	116.57	119	1.22	447.12	9363	7.55	26.67
J120,59-3	116	121	120.77	124	1.85	407.45	9008	7.41	7.41
J120,59-4	118	120	120.17	124	1.51	410.69	9602	9.26	10.28
J120,59-5	115	118	117.97	121	1.54	416.49	9361	8.49	18.56
J120,59-6	123	128	127.93	132	2.05	490.09	9514	6.96	19.42
J120,59-7	120	122.5	122.53	126	1.55	452.12	10141	7.14	33.33
J120,59-8	118	120.5	120.63	124	1.45	462.61	9365	7.27	20.41
J120,59-9	129	133	133.23	137	2.05	493.13	9756	8.4	15.18
J120,59-10	143	146	146.07	150	1.91	540.32	9961	8.33	34.91
J120,60-1	101	103	103.43	106	1.48	444.19	3004	0	0
J120,60-2	87	89	88.7	90	0.99	424.1	9332	4.82	7.41
J120,60-3	95	97	96.73	99	1.05	413.07	9273	6.74	17.28
J120,60-4	111	113.5	113.7	116	1.56	456.3	9515	7.77	9.9
J120,60-5	111	115	114.5	117	1.48	442.06	9664	5.71	15.63
J120,60-6	113	116.5	116.57	120	1.76	427.12	9513	2.73	2.73
J120,60-7	102	104	103.9	106	1.27	457.33	9905	7.37	21.43
J120,60-8	106	108	107.53	109	0.86	406.21	9333	4.95	4.95
J120,60-9	101	103	103.07	106	1.2	395.78	152	0	0
J120,60-10	94	96	96.4	99	1.07	523.06	9815	5.62	10.59
Total Average					**2.17**	**684.50**	**9925.06**	**6.53**	**38.89**