Development of Heuristics for Multi-Skill Resource-Constrained Project Scheduling

Submitted in

fulfillment of the requirements for the degree of

Doctor of Philosophy

by

DHEERAJ JOSHI

ID: 2011RME7143

Under the supervision of

Prof. M. L. Mittal

DEPARTMENT OF MECHANICAL ENGINEERING

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR

August 2019

Dedicated to

My family

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY
JAIPUR – 302017 (RAJASTHAN) INDIA

DECLARATION

I, Dheeraj Joshi, declare that this thesis titled, **"Development of Heuristics for Multi-Skill Resource-Constrained Project Scheduling"** and the work presented in it is my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this university.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself, jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: August, 2019

Dheeraj Joshi
(2011RME7143)

MALAVIYA NATIONAL INSTITUTE OFTECHNOLOGY
JAIPUR – 302017 (RAJASTHAN) INDIA

CERTIFICATE

This is to certify that the thesis entitled **"Development of Heuristics for Multi-Skill Resource-Constrained Project Scheduling"** being submitted by **Mr. Dheeraj Joshi (ID No.: 2011RME7143)** is a bonafide research work carried out under my supervision and guidance in fulfillment of the requirement for the award of the degree of **Doctor of Philosophy** in the Department of Mechanical Engineering, Malaviya National Institute of Technology, Jaipur, India. The matter embodied in this thesis is original and has not been submitted to any other University or Institute for the award of any other degree.

Prof. M. L. Mittal
(Supervisor)
Professor
Department of Mechanical Engineering,
MNIT, Jaipur
India

Place: Jaipur.
Date: August, 2019

ACKNOWLEDGEMENT

It is indeed a matter of great pleasure and proud privilege for me to introduce and present this thesis entitled **"Development of Heuristics for Multi-Skill Resource-Constrained Project Scheduling".**

On the very onset, I thank the Almighty for bestowing upon me His kind blessings and fortunate opportunity to undertake this work. I wish to extend my utmost gratitude and sincere thanks to my benevolent supervisor **Prof. M. L. Mittal** whose excellent guidance, constructive feedback and timely motivation have proved highly instrumental in bringing out this research work.

I am equally indebted to Prof. Dilip Sharma (Head, Mechanical Engineering Department) and my DREC members: Prof. A.P.S. Rathore, Prof. Rakesh Jain and Dr. Gunjan Soni for providing me with their insightful comments that helped in streamlining this work.

I would like to extend sincere thanks to the management of my working organization Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur for permitting me to carry out this research work. I owe overwhelming debt to Prof. (Dr.) N.K.Banthiya, Prof. (Dr.) S.L.Surana and Prof. (Dr.) N.C. Bhandari for helping me a lot in balancing out my professional and academic assignments. My special appreciation goes to (Late) Prof. Alok Mathur who initiated a spark in me to start this research and also acted as a catalyst and key motivator till his last breadth for timely completion of this work.

I duly acknowledge and thank all my friends and colleagues Mr. Manish Kumar, Dr. Manu Augustine, Dr. Manoj Kumar Sain, Mr. Ramkaran Yadav and Mr. Pal Manojkumar Ramchandra for all their help and support. I want to express my heartiest thanks to my parents and all family members for their sacrifices, moral support and constant encouragement. Above all, I express my utmost indebtedness to my beloved wife *Lovely* and cute little daughter *Kiddu* whose love and energy kept me going.

Dheeraj Joshi
(2011RME7143)

Abstract

Due to the increased market volatility and uncertainties in today's competitive environment, there is a dire need for organizations to be more proactive and responsive in fulfilling customers' demand. Over the past few decades, achieving flexibility and robustness has been a key and strategic objective of most of the business organizations to handle these uncertainties. Due to its extensive breadth and sheer diversity, the term 'flexibility' has crossed its traditional locus from manufacturing to various other fields including project scheduling. Amongst the different dimensions and facets of flexibility, labour flexibility (also termed as workforce flexibility or human resource flexibility) has been a subject of much academic enquiry in the recent years. This work focuses on project scheduling problems involving resource flexibility.

Resources are inevitably necessary for the execution and realization of any project. The scheduling problems under the limited resource environment have been studied in the literature as the resource-constrained project scheduling problem (RCPSP). Due to both its theoretical relevance and practical applicability, there has been a significant and conspicuous research in the area of RCPSP in the last few decades. In this research, one of the recent and practical extensions of the RCPSP termed as the multi-skill resource constrained project scheduling problem (MSRCPSP) has been considered for investigation. Unlike the RCPSP, in these problems resources are flexible in nature i.e. each resource has the functionality of various renewable resources. This scenario is highly pronounced particularly in call centers, software development companies, consultancy firms, maintenance or construction agencies where the team members are skilled to perform a variety of jobs. The scheduling decision, therefore, is twofold i.e. allocation of a particular resource to an activity; and the specific skill for which the resource is allocated.

A comprehensive literature review revealed that there exists a good literature pertaining to resource flexibility in shop-floor and job-shop scheduling but same has not been sufficiently addressed in the area of project scheduling. In most of the current works, it has been assumed that each activity require one skill and one resource unit in form of a staff member possessing the stated skill. Moreover, majority of these studies consider that staff members possess different skills with same proficiency levels. However, this is not true in real life. Usually in

organizations, a staff member possessing various skills may be expert in one (or more) particular skill(s) but may not be able to exhibit the same level of expertise in all skill types. The proficiency or expertise of a skill refers to the degree of sophistication, ease or superiority by which a staff member can deliver a particular skill. Also, is has been found that in majority of these studies only one scheduling objective (mostly the makespan) is considered. However, in many practical situations decision-makers are concerned about several objectives simultaneously which give rise to what is known as multi-objective multi-skill resource-constrained project scheduling problem (MO-MSRCPSP).

The MSRCPSP is considered to be NP-hard in the strong sense hence it is difficult to find optimal solutions for large sized real life problems in reasonable time. During the past few decades, metaheuristic approaches have been an indispensable choice to achieve near optimal solutions for many of the combinatorial optimization and NP-hard problems, project scheduling being no exception. One of the recent metaheuristics for optimization problems is teaching-learning- based algorithm (TLBO). The algorithm mimics the teaching-learning process commonly seen in classrooms. It has been successfully applied on mathematical benchmark functions and mechanical design optimization problems of continuous nature. To the best of the knowledge attained from literature survey, there is no reported work in literature having application of TLBO on standard RCPSP and its multi-skill version with finite resource requirements and consideration of skill proficiencies.

Under this motivation, this study presents a TLBO algorithm with some modifications as an alternative metaheuristic approach for solving the general class of the RCPSP as well as for the MSRCPSP and its multi-objective case. The thesis starts by application of the TLBO for the RCPSP with some modifications from its conventional form. In addition to teacher and learner phase, two additional phases namely *self-study phase* and *examination phase* have also been appended in the conventional TLBO for improving the exploration and exploitation capabilities of the algorithm. The comparative results on standard benchmark problems show that performance of the proposed TLBO is quite competitive with other approaches available in the literature.

Subsequently in the second phase of the research, a TLBO for the MSRCPSP with modified encoding and decoding schemes has been developed that can accommodate the multi-skilled resource assignment information in the solution. In addition, a genetic algorithm (GA) with similar configuration has also been developed for comparing with the TLBO. The test instances have been developed using the methodology proposed in the literature and comparative results show that the proposed TLBO is quite effective to solve the MSRCPSP.

To fill another research gap, unlike most of the current research work, a practical scenario has been considered wherein staff members possess different skills with different proficiency levels. More specifically, a multi-objective TLBO has been introduced for the MO-MSRCPSP with two objectives including minimization the project makespan along with minimization of total time elapsed with less-skilled resource assignments. A weighted-sum or scalarization method has been employed to design MO-TLBO and MO-GA to solve this problem. The results on the test instances having different project architectures establish that MO-TLBO can be an effective metaheuristic approach to tackle the MO-MSRCPSP.

Some typical beneficiaries of this research may include software development companies, consultancy firms, R & D based organizations, maintenance firms, big construction houses etc. which incorporate multi-skilled staff members to accomplish different client orders simultaneously. Future research in this area may be directed towards consideration of project scheduling under stochastic or non-deterministic environment with varying activity times and interrupted skill availabilities.

Contents

List of Figures

List of Tables

Abbreviations

ABC	Artificial Bee Colony
ACO	Ant Colony Optimisation
AL	Activity List
ARV	Average Response Variable
B&B	Branch & Bound
BM	Boctor Mutation
CII	Construction Industry Institute
CP	Constraint Programming
CS	Cuckoo Search
CPM	Critical Path Method
DBH	Decomposition Based Heuristic
DE	Differential Evolution
DEGR	Differential Evolution and Greedy Algorithm
DOE	Design of Experiment
EST	Earliest Start Time
EFT	Earliest Finish Time
GA	Genetic Algorithm
GRPW	Greatest Rank Positional Weight
HM	Hartmann Mutation
HAntCO	Hybrid Ant Colony Optimization
HBD	Hybrid Benders Decomposition
HS	Harmony Search
iMOPSE	Intelligent Multi-Objective Project Scheduling Environment
IT	Information Technology
LST	Latest Start Time
MC-NFP	Minimum Cost Network Flow Problem
MILP	Mixed-Integer Linear Programming
MIP	Mixed-Integer Programming
MMRCPSP	Multi-Mode Resource-Constrained Project Scheduling Problem

MOFOA	Multi-Objective Fruit Fly Optimization Algorithm
MOIWO	Multi-Objective Invasive Weeds Optimization Algorithm
MOPSO	Multi-Objective Particle Swarm Optimization
MO-MSRCPSP	Multi-Objective Multi-Skill Resource-Constrained Project Scheduling Problem
MRS	Modified Resource Strength
MSPSP	Multi-Skill Project Scheduling Problem
MSRCPSP	Multi-Skill Resource-Constrained Project Scheduling Problem
NC	Network Complexity
NP-hard	Non-deterministic Polynomial-time hardness
NPV	Net Present Value
NSGA-II	Non-Dominated Sorting Genetic Algorithm
PERT	Programme Evaluation and Review Technique
PSO	Particle Swarm Optimisation
PSP	Project Scheduling Problem
PSPFR	Project Scheduling Problem with Flexible Resources
PSPLIB	Project Scheduling Problem Library
RBRS	Regret-based Biased Random Sampling
RCPSP	Resource-Constrained Project Scheduling Problem
RCPSP-FWP	Resource-Constrained Project Scheduling Problem with Flexible Work Profiles
RIP	Resource Investment Problem
RSM	Response Surface Methodology
R&D	Research & Development
SA	Simulated Annealing
SDS	Skill Divergence Span
SF	Skill Factor
SGS	Serial Schedule Generation Scheme
SFLA	Shuffled Frog-Leaping Algorithm
TLBO	Teaching-Learning-Based Optimization
TS	Tabu Search

Chapter 1

Introduction

Chapter 1

Introduction

1.1. Background

Projects are inevitable in our day-to-day life. In fact, the growth and development of human civilization can be largely attributed to the timely and successful completion of various projects. The term ‘project’ can be defined as a non-repetitive process with definite start and end time having specific objective(s) to be achieved under given constraints of time, cost and other resources.

Many of the modern projects involve hundreds or even thousands of inter-related activities and their timely execution demands a formal and structured approach. Project management, a continuously growing management discipline, primarily involves *planning*, *scheduling* and *controlling* of projects. During the *planning* phase the activities and their characteristics (such as duration, nature, resource requirements, inter-relationships etc.) which must be carried out to achieve the stated project objective(s) are defined. In addition, the constraints definitions are also formalized in this phase. In the *scheduling* phase, the start and end times of various activities are determined, of course, by honoring the given constraints. Finally, in the *controlling* phase, monitoring and expediting of the project is carried out to ensure that activities are executed as scheduled. In addition, during control phase the project managers may also carry out corrective actions such as schedule repair or re-scheduling if unacceptable aberrations are found during project execution. Undoubtedly, the competitiveness and success of a business organization depend heavily on how effectively it employs project management tools to the projects. The applications of project management are manifold and can be found in constructional activities, public amenities and infrastructure, software development, process engineering, research and development, repair and maintenance activities and many more.

The focus of this work is on *scheduling* aspect of project management which demands timely and effective allocation of resources to the project activities. This has been increasingly difficult and challenging in recent years primarily due to the dearth of skilled and sophisticated resources,

ever alarming energy crisis and increased geographical and political constraints. The scheduling problems of projects have been studied under a generic name called Project Scheduling Problem (PSP). A PSP involves a whole class of problems comprising different variations in fundamental project characteristics such as resources, activities, their inter-relationships and also the stated objective(s). The project scheduling function aims to determine the start/finish times of the activities involved in a project so that some predefined performance metric (e.g. makespan, profit, cost, NPV etc.) can be optimized for a given set of constraints. The nature and types of resources which have a profound effect in the execution and completion of a project will be the focus of discussion in the next section.

1.2. Project scheduling under limited resources

The early studies of project scheduling namely Programme Evaluation and Review Technique (PERT) by Malcolm et al. (1959) and Critical Path Method (CPM) by Kelley and Walker (1959) considered unlimited availability of resources which is, no doubt, an unrealistic assumption. In real life situations, different resources required by activities such as machines, humans, energy, budget etc. are indispensably limited in quantity and this demands their efficient and proper utilization. The scheduling problems under the limited resource environment have been studied in literature as the resource-constrained project scheduling problem (RCPSP). Due to both its theoretical relevance and practical applicability in diverse fields like research and development, maintenance, construction and software development, the RCPSP has attracted the attention of a large number of researchers. For a holistic and comprehensive view of the nature, variants and solution approaches of the RCPSP, reader is referred to surveys by Kolisch and Hartmann (2006), Hartmann and Briskorn (2010) and Habibi et al. (2018).

Resources are inevitably necessary for the execution and realization of project activities (except dummy) and can be broadly classified at three levels: *categories*, *types* and *units* (Weglarz et al., 2011). The *category* classification is established on the view-point of resource limitations based on time and can be further divided into following three basic sub-categories:

- *Renewable* resources which are constrained only on a period basis i.e. the amount available of such a resource is fixed and constant for the entire planning horizon.

This is because the units of such resource are released immediately as soon as the execution of an activity is over. Examples include human resources, machines and equipment etc.

- *Non-renewable* resources which are constrained only on total consumption over the entire planning horizon but having no restriction on per period availability. Units of non-renewable resource once consumed cannot be assigned to other activities. Examples are capital budget, raw material etc.
- *Doubly constrained* resources which are constrained both on a period-to-period basis as well as for the entire project time interval. It is either used (e.g. tools, machinery, equipment etc.) or consumed (e.g. raw material, money etc.) by an activity during its execution. It is important to note that each doubly constrained resource can be suitably substituted by a single renewable and another non-renewable resource. (Talbot, 1982).

According to the *type* classification, resources are distinguished as per their functional capabilities. In other words, two or more resources of same type are equally capable to execute a given activity. Finally, the amount of resource availability can be defined under classification by *unit* basis which simply identifies the number of units (for discretely-divisible like machines, tools, workers etc.) or amount of resources (for continuously-divisible like money, energy, liquid etc.) required to execute a given activity.

Besides these basic categories, several other categories of resources can also be found in literature such as preemptable and non-preemptable resources (Blazewicz et al., 1986), reusable resources (Shewchuk and Chang, 1995), dedicated resources (Bianco et al., 1998), spatial resources (De Boer, 1998), partially renewable resources (Bottcher et al., 1999), cumulative resources (Neumann et al., 2002), *multi-skilled* resources (Neron, 2002), synchronizing resources (Schwindt and Trautmann, 2003), adjacent resources (Duin and Van Der Sluis, 2006), changeover resources (Neumann et al, 2006) and heterogeneous resources (Tiwari et al., 2009) etc.

Among the above mentioned categories, this work particularly focuses on ‘multi-skilled’ resources wherein flexible resources have been considered i.e. each resource has the

functionality of various renewable resources. This scenario is highly pronounced particularly in call centers, software development organizations, consultancy firms, maintenance or construction agencies where the team members are skilled to perform a variety of jobs. For example, a coder in a software development project can perform both coding and debugging; a mason in construction projects can perform leveling, plumbing etc.in addition to plastering. In this context, a given task which requires certain skill(s) can be accomplished by any staff member possessing the stated skill(s). Besides human resources, multi-skilled resources also include multi-purpose machines, robots, automatic machining centers etc. The problem in literature has been studied under various names viz. "multi-skill project scheduling problem (MSPSP)" (Bellenguez-Morineau & Néron, 2007), "project scheduling problem with flexible resources (PSPFR)" (Correia et. al., 2012) and "multi-skill resource-constrained project scheduling problem (MSRCPSP)" (Myszkowski et al. 2015). The underlying and common factor between all these problem classes is that a resource is considered to possess more than one skill. The scheduling decision, therefore, is twofold i.e. allocation of a particular resource to an activity; and the specific skill for which the resource is allocated.

Due to its obvious applications in a lot of real-life projects, the MSRCPSP has gained quick attention of researchers since its advent and it still presents a potential area of research. This work focuses both on single-objective and multi-objective MSRCPSP which is again a scarcely treated work in literature. The concept of resource flexibility and its variants in project scheduling has been discussed in sufficient length in the next chapter. For the sake of brevity and convenience, the detailed discussion on the other resource categories mentioned in this section has been omitted.

1.3. Solution approaches in project scheduling

The RCPSP is combinatorial in nature and considered to be NP-hard (Blazewicz et al., 1983) in the strong sense hence it is difficult to find optimal solutions for large sized real life problems in reasonable time. The MSRCPSP, being a practical extension of the RCPSP, are more complex in the sense that number of possible ways of resource allocation can be exceptionally large.

During the past few decades, heuristics approaches have been an indispensable choice to achieve near optimal solutions for many of the combinatorial optimization and NP-hard problems, project scheduling being no exception. Unlike traditional mathematical programming techniques, heuristics are simple and intuitive in nature that determines sub-optimal or 'good' solutions in reasonable amount of computational time. The early heuristics proposed for the RCPSP were constructive in nature and primarily based on priority rules and a schedule generation scheme (Kolisch, 1996). In pursuit of finding better solutions for practical optimization problems researches have proposed a variety of advanced heuristics called metaheuristics. These are a recent class of heuristics which start from one or more initial feasible solutions and imitate some natural or physical phenomena (e.g. genetic algorithms, simulated annealing etc.) to achieve convergence towards a global optimum solution. The metaheuristics have proved far efficient than rule-based heuristics in many fields and including project scheduling.

Although a number of metaheuristic approaches exist in literature but it is interesting to note that no single approach can be guaranteed to give better results for all types of scheduling problems. This is primarily due to the inherent nature and solution mechanisms that these approaches employ, while one may be good at exploration, other at exploration. Nevertheless, this has constantly motivated practitioners and researchers to employ and explore the behaviour of newly designed metaheuristics based on various physical or natural phenomena.

1.4. Motivation for the work

A comprehensive literature survey revealed that although resource limitation aspect has been sufficiently addressed by researchers in project scheduling, the notion of skilled resources still presents a potential area of research. There exists good literature pertaining to resource flexibility in shop-floor and job-shop scheduling (Dauzere et al., 1998) but same has not been sufficiently addressed in the area of project scheduling. More specifically, numerous heuristic and metaheuristic approaches can be found for the RCPSP (Hartmann and Briskorn, 2010) but this is not true in case of MSRCPSP. The work by Neron (2002) wherein two lower bounds have been proposed can be attributed as the initial contribution to the MSRCPSP. Bellenguez-Morineau & Néron (2007) introduced branch-and-bound method for MSRCPSP which can tackle problems of only small scale. Later, Correia et al. (2012) developed an MILP formulation for the PSPFR and

introduced few more inequalities in order to solve it more effectively using a general solver. The concept of skilled resources has also been harnessed by researchers in in the area of personnel assignment (Ernst et al., 2004) but there is still a lack of efficient metaheuristics or solution approaches for solving large scale MSRCPSP.

Moreover, in the above studies, only one scheduling objective (mostly the makespan) is considered. However, in many practical situations decision-makers need to optimize several objectives functions simultaneously which give rise to what is known as multi-objective multi-skill resource-constrained project scheduling problem (MO-MSRCPSP). Recently, some work has been reported in literature in this growing area. Maghsoudlou et. al (2016) proposed multi-objective invasive weeds optimization algorithm (MOIWO) with a new chromo-some structure guaranteeing feasibility of solutions to solve this problem. Besides minimizing project's makespan, the two other objectives considered were minimizing total cost of allocating workers to skills and maximizing total quality of processing the activities. In addition, a knowledge-guided multi-objective fruit fly optimization algorithm (MOFOA) was proposed by Wang et. al (2018) for MO-MSRCPSP with the criteria of minimizing the makespan and the total cost simultaneously.

In most of these works, it has been assumed that a staff member possess different skills with same proficiency levels. However, this is not true in real life. Usually in organizations, a staff member possessing various skills may be expert in one (or more) particular skill(s) but may not be able to exhibit the same level of expertise in all skill types. To elaborate further, a coder in a software developing project may be highly expert to code in JAVA platform but may have moderate level of proficiency in Python, Elixir, TypeScript or other programming languages. The concept of this varied level of proficiencies has been studied in literature under the name 'hierarchical skills levels' for single-objective problems but no substantial work can be found particularly for the multi-objective MSRCPSP.

One of the recent metaheuristics for optimization problems is teaching-learning-based algorithm (TLBO). The algorithm, introduced by Rao et al. (2011), mimics the teaching-learning process commonly seen in classrooms. It has been successfully applied on mathematical benchmark functions and mechanical design optimization problems of continuous nature. The TLBO has

been reported to have high convergence rate and it also inherits a merit of few algorithm specific parameters to tune (Rao et al., 2011). Inspired by the performance of TLBO on continuous non-linear problems, researchers have also applied it on discrete optimization problems. However, to the best of the knowledge there is no reported work in literature having application of TLBO on standard RCPSP and its multi-skill version with finite resource requirements and consideration of skill proficiencies. Under this motivation, this thesis presents a TLBO algorithm with some modifications as an alternative metaheuristic approach for solving general class of the RCPSP as well as for the MSRCPSP and its multi-objective case.

1.5. Research objectives and the scope of the work

The discussions made in section 1.4 highlight some relatively less studied areas in the domain of "project scheduling with flexible resources". This has been a key motivation to develop efficient solution techniques for the multi-skill resource-constrained project scheduling problem (MSRCPSP) and also for multi-objective multi-skill resource-constrained project scheduling problem (MO-MSRCPSP) taking skill proficiencies into account. The behaviour of the developed metaheuristic has been tested on the standard resource-constrained project scheduling problem (RCPSP) as a preliminary approach.

In particular, following research objectives have been formulated in this work:

1. *To develop metaheuristic for the resource-constrained project scheduling problem (RCPSP).*
2. *To develop metaheuristic for the multi-skill resource-constrained project scheduling problem (MSRCPSP).*
3. *To develop metaheuristic for the multi-objective multi-skill resource-constrained project scheduling problem (MO-MSRCPSP).*

As explained earlier, there are a number of variations and extensions of the classical RCPSP and hence no approach can be generalized and suitable for all problem class. Having said this, the scope of the current work is limited by following assumptions:

- Single-mode for execution of activity and renewable type multi-skilled resources have been considered in this research work.
- The prerequisite activity relations are of the finish-to-start type i.e. there exist zero time lags between activities.
- The deterministic and static scheduling is assumed which implies that all activity times, resource requirements and their availabilities are known a priori and do not alter during execution.
- The preemption of activities is not permitted and all skills needed by an activity need to be available at its start.

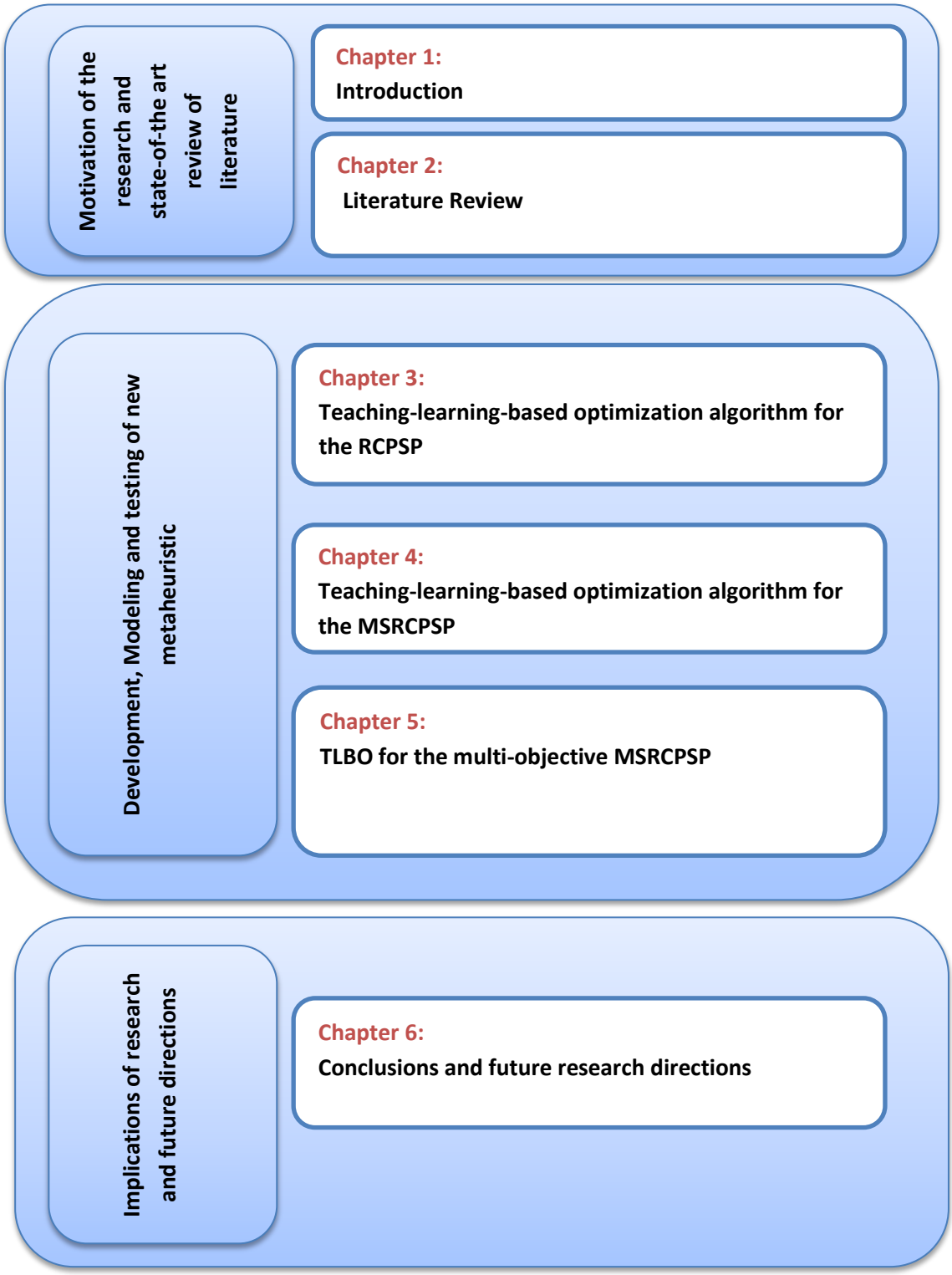
1.6. Organization of the thesis

This section discusses a brief overview of the contents included in different chapters of this thesis. Figure 1.1 gives a pictorial view of the organization of the thesis. The core contents and inclusions of each chapter have been mentioned below:

In **chapter 1,** which is the current chapter, an introduction to the resource-constrained project scheduling along with relevance of metaheuristics as potential solution approach in this area is presented. In addition, it also brings out motivation and justification of incorporating multi-skilled nature of resources in the current work. Finally, research objectives and scope of the work have been exhibited.

Chapter 2, reviews the state-of-the-art literature on the RCPSP and MSRCPSP with special emphasis on concept of resource flexibility or multi-skill resources. The research gaps have been ascertained in the end as an outcome of the survey.

In **chapter 3**, a recently introduced metaheuristic teaching-learning-based optimization (TLBO) has been proposed for the RCPSP with some modifications from its conventional form. The key philosophy of TLBO and its framework of application for the problem in hand have been discussed in detail. In addition the chapter also presents the results of the comprehensive computational experiments conducted to test the behaviour of the developed algorithm.



.Figure 1.1: Pictorial view of organization of the thesis

In **chapter 4**, a TLBO algorithm for the multi-skill version of the RCPSP, i.e., MSRCPSP has been introduced. In particular, a mathematical model for the problem has been formulated and a novel decoding scheme for the MSRCPSP has been proposed to suit the multi-skill nature of resources. As another contribution of this work, the chapter also presents a modified schedule generation scheme to avoid resource conflicts between competing activities and ensuring resource feasibility.

In **chapter** 5, another scarcely treated work in literature has been targeted namely multi-objective multi-skill resource-constrained project scheduling problem (MO-MSRCPSP) by giving due consideration to different skill proficiencies of resources. A multi-objective mathematical formulation has been presented for this problem which aims to minimize two time estimates (1) the project makespan and (2) the total time elapsed with under-skill resource assignments denoted as *Skill Divergence Span* (SDS). To solve this complex problem, a priori approach of multi-objective optimization has been used as it is simple and intuitive. The TLBO algorithm developed in chapter 5 for MSRCPSP has been appended with some modifications to facilitate multi-objective optimization.

In **chapter** 6, finally summarizes the significant contributions and highlights of the research. Besides, an insight has also been given of important directions for accomplishing future research in the area of project scheduling.

Chapter 2

Literature Review

Chapter 2

Literature Review

This chapter presents a comprehensive and state-of-the-art review on contributions made by researchers in the project scheduling area with special focus on scheduling under flexible resource profiles. More precisely, it investigates the multi-skilled resource-constrained project scheduling problem (MSRCPSP) which has been realized as a recent extension of the classical resource-constrained project scheduling problem (RCPSP). A brief overview of the generic concept of flexibility with special emphasis on workforce flexibility as an important facet and dimension of flexibility has also been presented in this chapter. The key contributions and available solution approaches in this area that may steer new research potentials have been discussed in sufficient length. In the later section, significant contributions in the multi-objective optimization under multi-skilled resources have been highlighted. For the purpose of completeness and coherence, the review of the standard RCPSP is presented first and subsequently it proceeds towards the multi-skilled extension of this problem. Figure 2.1 presents the systematic flow of literature review that has been adopted in this work.

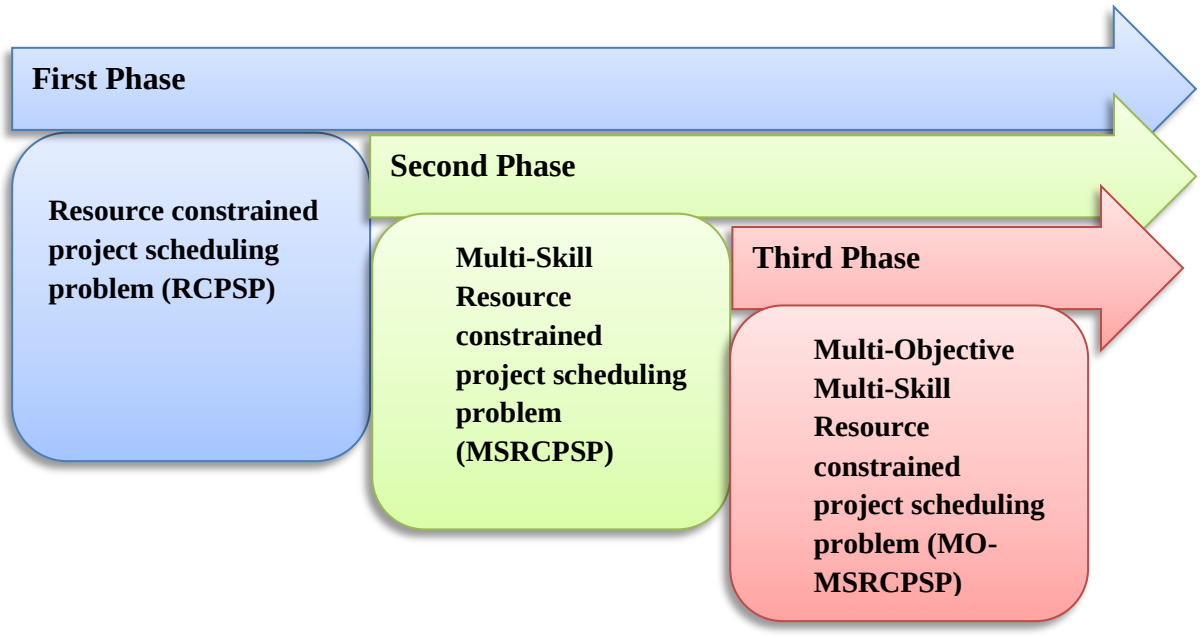


Figure 2.1: Flow of literature review

2.1 Resource-constrained project scheduling problem

Since its advent, the RCPSP has been considered as one of the most intractable problems in operations research community and hence attracted ample attention of researchers and practitioners. The reason can be attributed to its sheer diversity and applicability in various fields like R & D, maintenance, manufacturing, construction and software development etc. Over the last few decades, it has been a standard problem for project scheduling which deals with determination of start/finish times of the activities involved in the project by respecting the precedence and resource constraints and fulfilling a specific objective (s).

Since the early work on the RCPSP by Weist (1967), a number of researchers have contributed significantly in this area. It is important to note that initial solution techniques were mainly based on optimal or exact approaches such as dynamic programming, zero-one programming (Patterson and Roth, 1976), implicit enumeration (Patterson et al., 1990), branch and bound (Demeulemeester and Herroelen, 1992; Brucker et al., 1998) etc. Figure 2.2 classifies the various exact solution approaches employed by researchers to tackle the RCPSP. It is worthwhile to mention that exact approaches were found to be incompetent to provide good solutions for practical size problems which involve large number of activities and high network complexity.

As the RCPSP is a generalized form of job-shop scheduling problem and proved to be NP-hard (Blazewicz et al., 1983) in the strong sense, the optimal solution is difficult to achieve for problems where number of activities exceeds around 60. This has motivated a lot of researchers in the past few decades to search for the near-optimal or heuristic solution approaches for solving the RCPSP.

The early heuristics for the RCPSP were constructive in nature and primarily based on priority rules applied in a single or multi-pass fashion. A serial or parallel schedule generation scheme (Kolisch, 1996) is generally used to obtain a feasible solution from a given priority list. The study based on single pass priority rules and their relative comparison can be found in Boctor (1990). In this work it was shown that no single heuristic consistently perform well for a given problem and the performance may depend upon type of problem, network complexity and other related parameters.

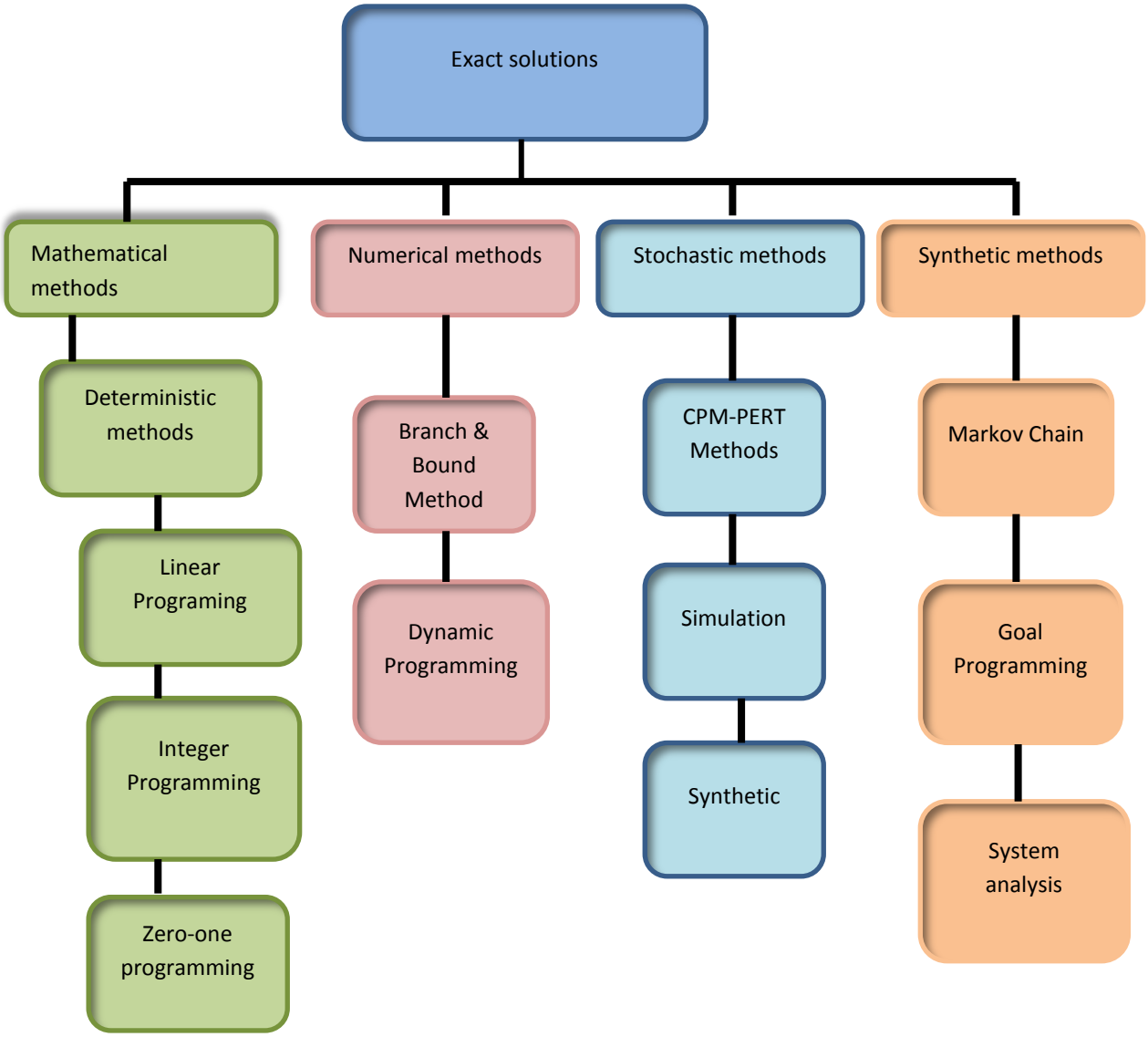


Figure 2.2: Exact solution approaches for the RCPSP

This motivated the researchers to devise another class of heuristics called improvement heuristics which basically employ multi-priority rules to reach at better solution. Besides the conventional multi-priority rule methods, forward-backward methods (Li and Willis, 1990) and sampling techniques (Tormos and Lova, 2001) have also shown good results for the RCPSP. The sampling techniques, in particular, being based on random device and probabilistic concepts have proved better than other priority rules and forward-backward based techniques (Kolisch, 1996) for regular measure of minimizing makespan. Figure 2.3 presents a pictorial overview of the different heuristics for the RCPSP available in the literature (Abdolshah, 2014).

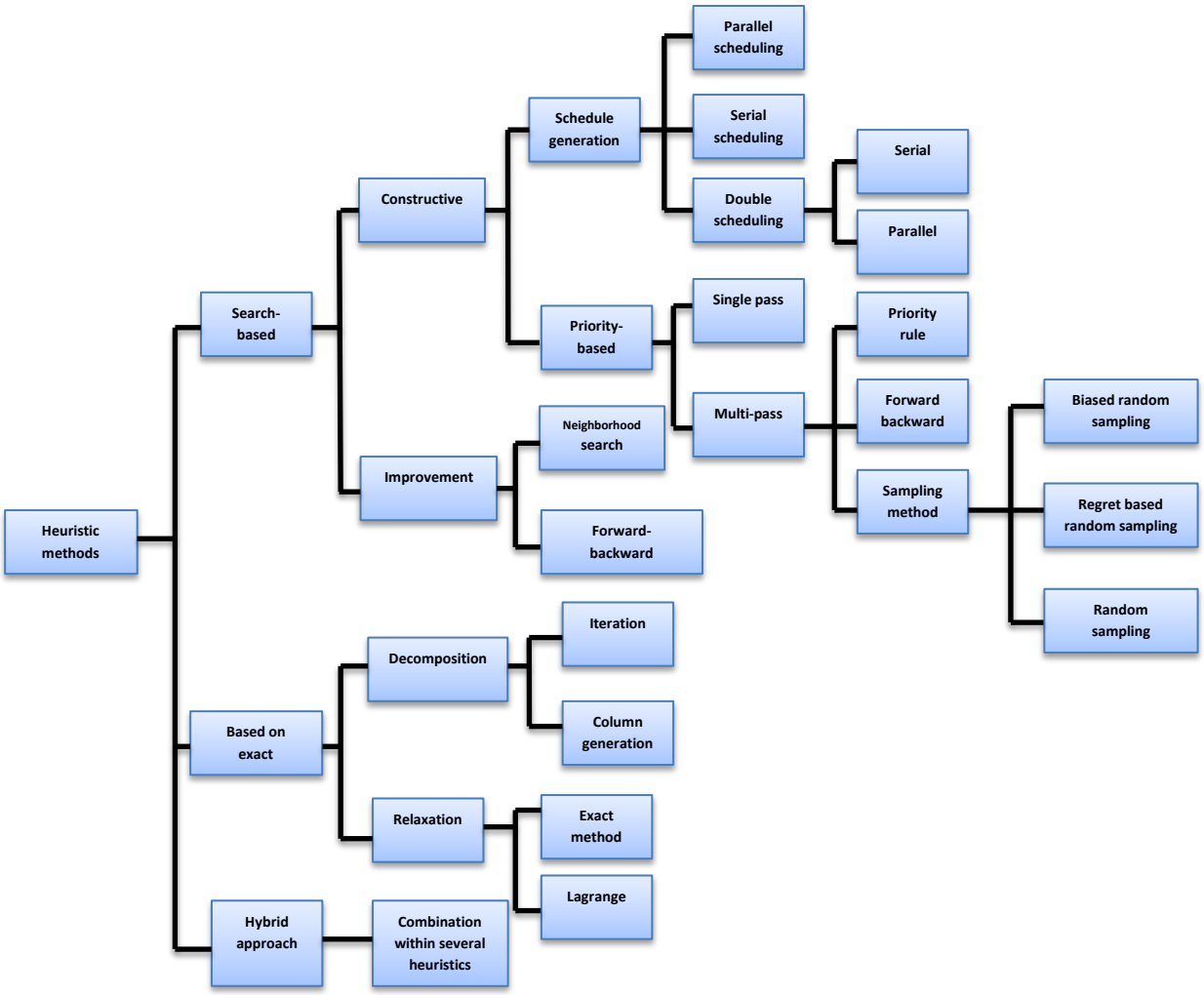


Figure 2.3: Classification of heuristics methods for the RCPSP

As mentioned earlier in chapter 1, in pursuit of finding better solutions for real life practical optimization problems of large and complex nature, researchers have proposed a latest class of heuristics called metaheuristics. The metaheuristics have proved far efficient than conventional heuristics in many fields and the RCPSP is no exception. These are generally inspired from nature or based on some physiological phenomenon. Figure 2.4 shows a summary of different metaheuristics that researchers have tested for the RCPSP.

Among the most widely applied population based metaheuristic for the RCPSP is genetic algorithm (GA). Some of these include GAs developed by Hartmann (1998, 2002), Alcaraz and Maroto (2001) and Mendes et al. (2009).

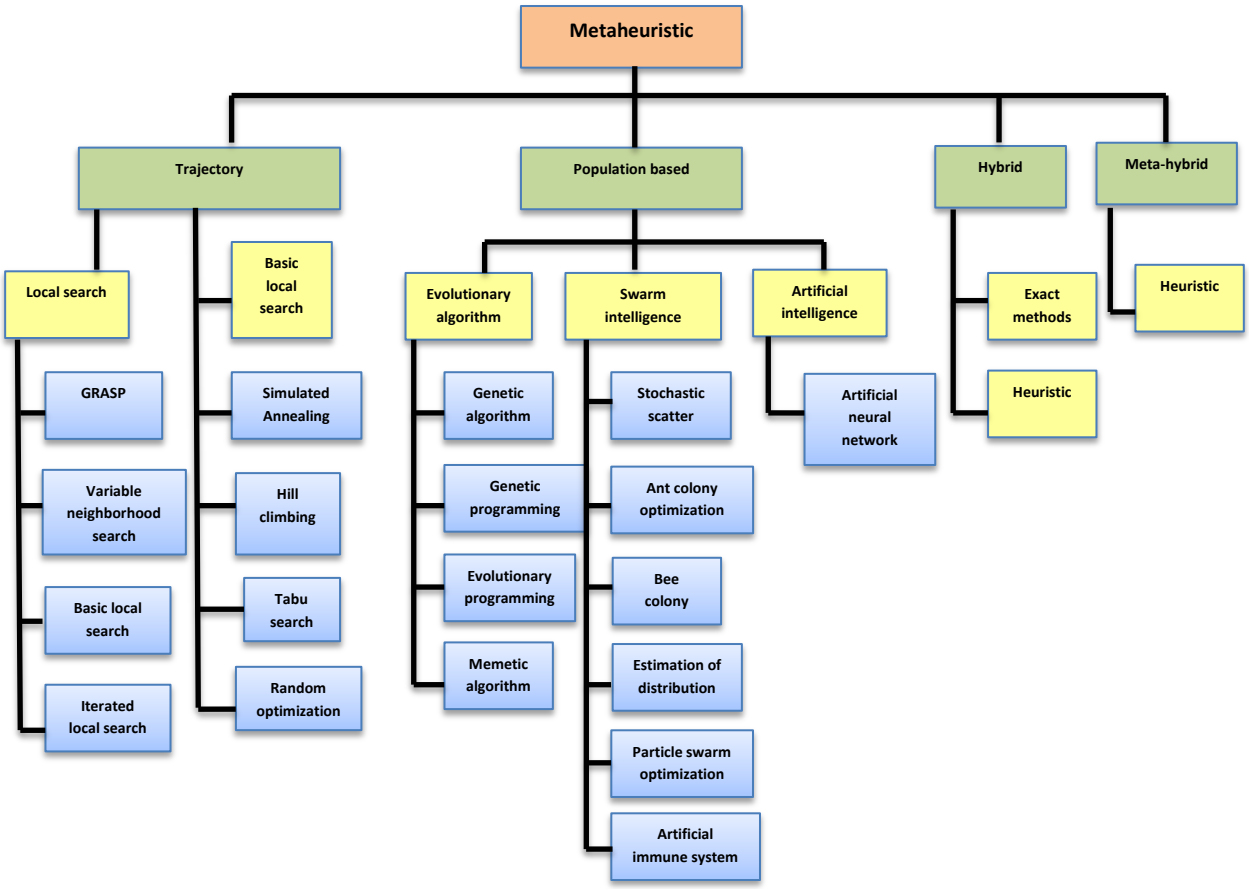


Figure 2.4: Metaheuristics methods for the RCPSP

In his GA, Zamani (2013) proposed a new magnet-based crossover operator which has been found very effective and competitive as compared to conventional crossover techniques. Simulated annealing (SA), which mimics the physical phenomena of cooling of metals or alloys, obtains an initial solution and repeatedly performs local alterations to achieve better solutions (Kirkpatrick et al., 1983). It has been successfully applied for RCPSP by Boctor (1996), Cho and Kim (1997) and Bouleimen and Lecocq (2003). Another metaheuristic is tabu search (TS), which also starts with a feasible solution and uses a tabu list to find the global solution by escaping from a local optimum. Some works that applied TS for the RCPSP includes those of Thomas and Salhi (1998), Klein (2000) and Pan et al. (2008). In another such work, Bukata et al. (2015) employed a parallel TS algorithm using graphics processing units (GPUs) to solve this problem which proved very effective in terms of quality of solutions and number of evaluated schedules. Under the applications of swarm intelligence optimization techniques, Zhang et al. (2005, 2006)

employed particle swarm optimization (PSO) wherein two different representations namely priority-based and permutation-based representations were considered for the RCPSP and their relative performance was analyzed.

Besides these well-known metaheuristics, researchers have also tested the performance of some other specific metaheuristics on the RCPSP. For example, Fang and Wang (2012) applied shuffled frog-leaping algorithm (SFLA) to solve this problem by encoding an activity list as virtual frog and using specific serial schedule generation scheme to decode the same. In the work of Eshraghi (2016), differential evolution (DE) algorithm embedded with local search techniques was employed to solve the RCPSP. In a recent application, the scheduling problem in construction industry was tackled by harmony search (HS) algorithm which is based on a musician's search process to achieve a better harmony (Giran et al. 2017). To tackle a practical extension of RCPSP, Lacomme et al. (2017) introduced a new shortest path algorithm that considers both routing and scheduling.

Due to the rapid development in the complexity theory and maturations in artificial intelligence techniques, research communities have begun to introduce more and more practical extensions of the RCPSP. As a matter of fact, project scheduling under limited resources has escaped from its conventional locus of operation research or management science to other applications including those in control theory, computer sciences, system simulation etc. In recent years, there has been an unprecedented growth not only in defining new paradigms of the RCPSP but also in devising its advanced solution approaches. In order to understand different solution approaches, variants and extensions of the RCPSP, reader is referred to surveys by and Hartmann and Briskorn (2010) and Weglarz et al. (2011).

Recently, Habibi (2018) identified four main characteristics of the RCPSP which may be attributed for possible variations in realizing practical extensions of the standard RCPSP. These are resources, concept of activities, objective functions and the availability level of information. Figure 2.5 depicts these characteristics along with their sub-classifications.

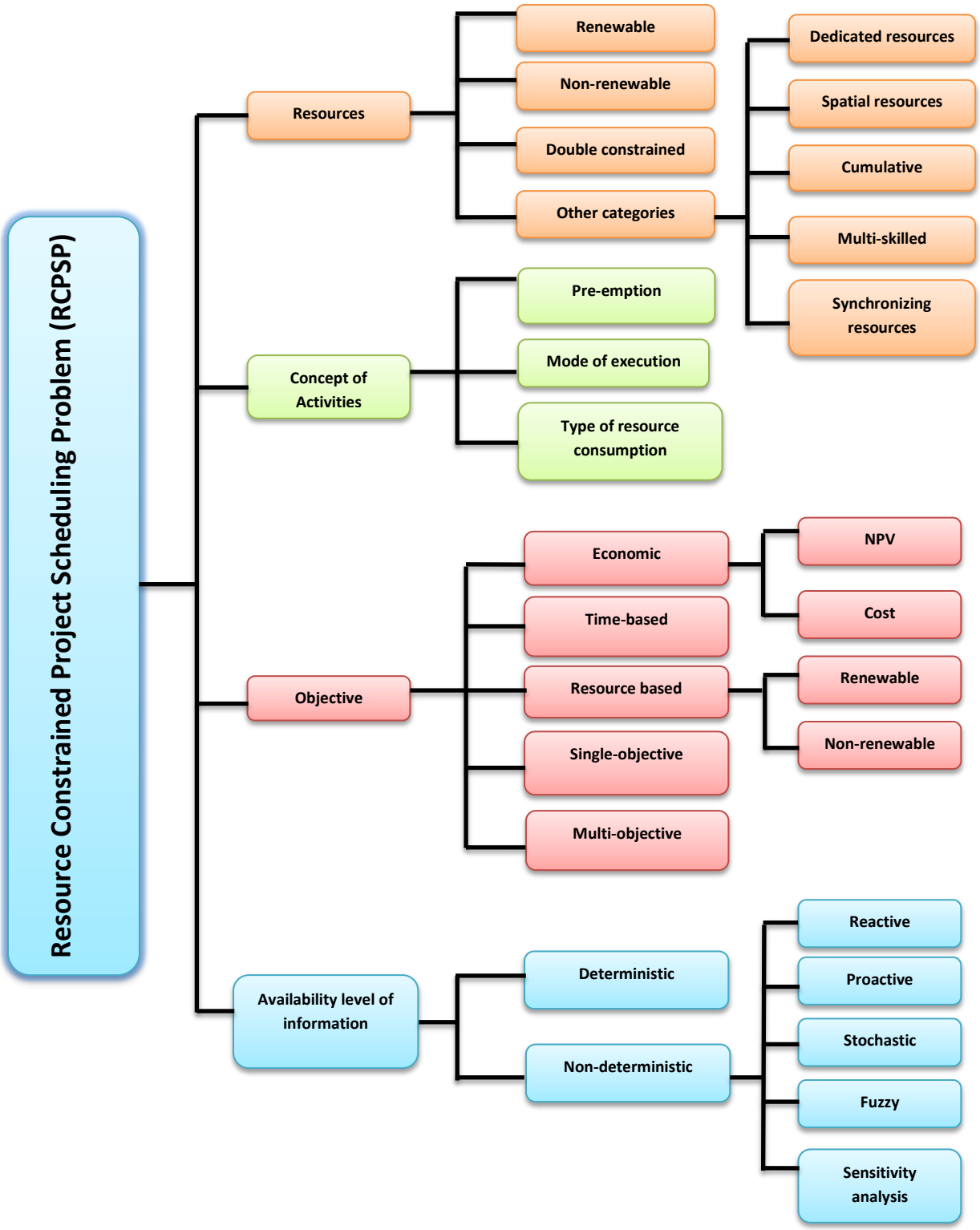


Figure 2.5: Variations and extensions of the RCPSP (Habibi et al., 2018)

Out of these various real-life generalizations, the current work particularly focuses on the concept of flexible resource or more specifically multi-skill resource categories for single mode project scheduling. The next section presents the facets and dimensions of flexibility that have been conceptualized by researchers in the area of manufacturing and thereby identifies their key applications in the context of project scheduling.

2.2 Flexibility in project management

2.2.1. Theoretical concept of flexibility

For the past few decades, the concept of *flexibility* has gained ample attention from researchers due to its importance and relevance in diverse fields such as design, manufacturing systems, production planning and control, construction, service centers, software development, etc. The generic concept of flexibility seems difficult to define due to extreme diversity in its connotation; however, it can be identified and realized by certain characteristics such as responsiveness, adaptability, resilience, rapidity, efficiency, reliability and sometimes complexity (Bordoloi et al., 1999). Olsson (2006) analyzed the dynamics related to project flexibility, both from a theoretical and an empirical perspective.

2.2.2. Flexibility versus uncertainty

Organizations in today's world are exposed to various types of uncertainties on account of continuous and dynamic changes that often become quite significant in magnitude. Uncertainties in projects are usually one of a major concern of project stakeholders which may have unproductive repercussions and detrimental effects on the overall project growth and hence on the expected objective(s). Atkinson et al. (2006) categorized the overall uncertainty in a project relying on three attributes: uncertainty associated with estimation, uncertainty due to different project life cycles and uncertainty due to project stakeholders. Herroelen and Leus (2004) defined uncertainty in terms of magnitude of different project parameters viz. time, cost and quality as well as process intricacies like what, how, when, by whom and at what cost a process has to be done. In order to face these uncertainties, organizations are expected to be internally and/or externally flexible. Undoubtedly, the ever increasing demand of high quality goods and services and that too at low cost by customers impose an additional pressure on organizational

functionalities to develop robust and flexible systems that can handle uncertainties. In fact, the discipline of project risk management relies heavily on how accurately practitioners are able to ascertain possible reasons of uncertainties.

The application of concept of flexibility to handle uncertainties has attracted significant attention and recognition by researchers in the recent years. Groote (1994) demarcated flexibility as "a hedge against the diversity or uncertainty of the environment". The word 'diversity' was interpreted to represent variety, complexity, variability in the types and quantities demanded by customers. A flexible technology-enabled organization is capable of responding in favourable manner towards various environmental uncertainties in a better fashion as compared to its competitors. Beach et al. (2000) recommended reactive and proactive flexibility approaches to handle an uncertain environment. An early attempt depicting the association of flexibility types and uncertainty was made by Gerwin (1987) and shown in Table 2.1. It is interesting to note that each type of uncertainty needs a different and specific type of flexibility to accommodate it.

Table 2.1: Association of flexibility and uncertainty types (Gerwin , 1987)

Flexibility type	*Association to uncertainty type*
Mix	Uncertainty as to which products will be accepted by customers created a need for mix flexibility
Changeover	Uncertainty as to the length of product life cycles leads to changeover flexibility
Modification	Uncertainty as to which particular attributes customers want leads to modification flexibility
Re-routing	Uncertainty with respect to machine downtime makes for rerouting flexibility
Volume	Uncertainty with regard to the amount of customer demand for the products offered leads to volume flexibility
Material	Uncertainty as to whether the material inputs to a manufacturing process meet standards gives rise to the need for material flexibility
Sequence	Sequence flexibility arises from the need to deal with uncertain delivery times of raw materials

2.2.3. Flexibility dimensions in manufacturing

The concept of flexibility in manufacturing well existed in literature (Dauzere-Peres et al., 1998). Due to the very nature of diversity in flexibility definition, researchers have identified different traits and dimensions of manufacturing flexibility such as quantitative, qualitative, offensive, defensive, static, dynamic, internal and external. A firm's overall flexibility is a blend of these various dimensions, of course, with varied levels of each type. In their work, Golden and Powell (2000) identified four more dimensions of flexibility namely intension, focus, range, and temporal. The temporal flexibility can be further classified into three sub-categories namely strategic, tactical and operational flexibilities. Further, in the work of Koste and Malhotra (1999), operational flexibility of an organization was categorized into various other flexibilities such as labour flexibility, product flexibility, volume flexibility, mix flexibility, material handling flexibility, routing flexibility, expansion flexibility etc.

2.2.4. Human resource flexibility

Out of the various facets and dimensions of manufacturing flexibilities mentioned above labour flexibility will be of particular interest in this work. The term has been invariably used with workforce flexibility or human resource flexibility in the literature and may be defined in terms of number of heterogeneous or variety of jobs that a labour or worker is capable to perform with satisfactory level of cost, quality and performance. Due to their inherent mobility and ability to migrate from one work station to another, human resources can be considered as one of the key contributors in imparting flexibility to an organization. Table 2.2 (Goudswaard and De Nanteuil, 2000) shows the two basic forms of human resource flexibility namely external flexibility and internal flexibility. These have been further categorized into several sub-categories as shown. To achieve external flexibility the required workforce is balanced by hiring more workers on short term contracts from external market. This facilitates organizations to realize a rapid and costless strategic flexibility to cope up with the turbulent customers' demand.

There are negative impacts of developing external flexibility in terms of lack of core competences among employees, high hiring and layoffs costs, reduced morale and motivation of workers and also health and safety issues. Moreover, the flexibility achieved is a static one rather

than dynamic in nature. To overcome these issues, the organizations often strive for developing internal flexibility which is also referred as multi-skills flexibility. Multi-skills flexibility is defined as the ability of human resources to carry or execute variety of tasks rather than fixed or conventional ones. This kind of flexibility is also sometimes categorized into functional flexibility which requires rigorous training programme to incorporate in an organization. In this thesis focus is given on multi-skill flexibility of workers which is known to play a paramount role in handling the dynamic and static uncertainties of an organization.

Table 2.2 Different forms of flexibility (Goudswaard and De Nanteuil, 2000)

Forms of flexibility	*Quantitative flexibility*	*Qualitative flexibility*
External flexibility	*Employment status*	*Production system*
	• Permanent contracts	• Subcontracting
	• Fixed-term contracts	• Outsourcing
	• Temporary agency contracts	• Self employed
	• Seasonal work	
	• Work on demand/call	
	Numerical flexibility and/or contract flexibility	*Productive and/or geographical flexibility*
Internal flexibility	*Working time*	*Work organization*
	• Reduction of working hours	• Job enrichment/job rotation
	• Over-time/part-time work	• Teamwork/autonomous work
	• Night and shift work	• Multitasking, multi-skilling
	• Weekend work	• Project groups
	• Compressed working week	• Responsibility of workers over: planning, budget, innovation, technology
	• Varying working hours	
	• Irregular/unpredictable working time	
	Temporal flexibility	*Functional flexibility*

Although the flexibility concepts have been incorporated and tested in different fields of study, in the light of scope of this work, only review of the works pertaining to flexibility in the project scheduling environment is presented. Moreover, to keep the review focused, the resources

considered are only humans or staff members ignoring other flexible resources such as multi-purpose machines, automatic workcentres, etc.

2.3. Project scheduling under flexible resources

Realizing the importance and relevance of multi-skilled staff in increasing competitive advantage of an organization, a lot of researchers in recent years have formulated mathematical models and applied them in various scheduling problems that involve flexible resources. There are primarily two different scenarios or set-ups where flexible resources have been studied. In the former, the review of the staff assignment problems involving multi-skilled resources is done and later section focuses on state-of-the art in the multi-skill resource-constrained project scheduling area.

2.3.1. Multi-skill personnel and staff assignment problems

These problems are not essentially in a project architecture form, but largely concerned with optimum allocation and assignment of flexible resources in different applications such nurse rostering; school timetabling, workforce balancing etc. Cai and Li (2000) formulated a multi-criteria optimization model for a staff scheduling problem with three objectives: minimization of total cost incurred in assigning staff to meet manpower requirements; maximization of surplus of staff for same assignment cost; minimization of variations in surplus staff for different time periods. It is interesting to note that problem is not purely of multi-skill type in the sense that not all staffs members were assumed to possess multi-skills. More specifically, three types of staff members were considered such that type-I members were able to do type-I job, type-II were able to do type-II job and type-III staff members were able to perform both type-I and type-II jobs. In order to handle this complex problem a multi-point crossover based GA was proposed which proved effective in finding desirable solutions.

In the work of Cordeau (2007), a construction heuristic coupled with an adaptive large neighborhood search heuristic was proposed for scheduling of technicians and tasks in a telecommunication company. In the model, each technician was assumed to exhibit different levels of skill hierarchy in a number of skill domains. In addition, the tasks also vary in difficulty

in the sense that some tasks require more than one technician for their execution. The problem was given as the subject of 2007 challenge set-up conducted by French Operational Society (ROADEF) in collaboration with France Telecom. The algorithm stood second in the competition. Later, Firat and Hurkens (2011) improved the solution of the above problem by incorporating the opportunity of outsourcing some tasks. The solution approach introduced flexibility in matching model which resulted in improved packing of the experts requiring heterogeneous skill distribution.

In another variation of the staff scheduling problem with multi-skilled workforce, Li and Womer (2009) developed a hybrid benders decomposition (HBD) which combined the complementary strengths of both constraint programming (CP) and mixed-integer linear programming (MILP) to minimize the staffing costs under three constraint types: generalized temporal constraints, constraint in terms of project deadline on the makespan and lastly the multi-skilled resource constraints. It was shown by the computational study that hybrid MILP/CP algorithm performed significantly better and efficient than pure MILP or CP approaches when tested alone.

In one of the real-life applications of skilled workforce scheduling, Valls et al. (2009) considered various practical characteristics that are faced daily by Service Centers of the organizations. These include service quality agreements between client and company enforcing additional constraints on start and end dates for tasks with corresponding penalties in case of delay. Besides, other constraints considered were related to generalized precedence relations between tasks, time lags and variable task durations. Each worker was characterized by his/her efficiency levels and speed in executing different tasks. The objective chosen for the study was to obtain a quick feasible plan respecting the constraints of workers' timetable and maximum pre-established dates. A novel and hybrid GA which combined local search strategies with genetic techniques was proposed to solve this complex multi-objective staff scheduling problem.

Heimerl and Kolisch (2010) considered the problem of scheduling of IT projects involving multi-skilled resources with heterogeneous efficiencies with an objective of minimizing the labour costs. A MILP model with a sharp lower bound was proposed for this problem and benefits of applying it over simple heuristic in real-life projects were established.

Ranjbar and Kianfar (2010) used a different setting of skilled resources in their problem which consisted of only one renewable bottleneck resource and the processing time of activities and resource requirement was not pre-specified. Based on the total work content, all feasible work profiles were determined using a genetic algorithm approach. The problem was identified with the name resource-constrained project scheduling problem with flexible work profiles (RCPSP-FWP). For the same problem class, Naber and Kolisch (2014) proposed four discrete-time model formulations in which the resource usage of an activity can be adjusted from one period to other. The computational results indicated that the variable-intensity-based model performed significantly better than the other three models.

Gutjahr et al. (2008) proposed a non-liner MIP model for the project portfolio selection under multi-skilled resource environment. The model aimed to maximize the average profit of project selection, optimization of time and optimal assignment of persons to the selected projects. A greedy heuristic algorithm was employed to assign and schedule the persons and a metaheuristic to select the project. The solutions when compared with the lower bounds obtained by the exact solutions of simplified mathematical model proved very effective.

The concept of multi-skilled resources has also been suitably harnessed in the problems related to the area of construction engineering and management. For example, Hegazy et al. (2000) investigated a linear programming model to optimize the assignment and allocation process of a partially multi-skilled workforce. The transition of one worker from one activity or crew to another, known as 'switching' was minimized through this study. It was ascertained using Construction Industry Institute (CII) Model Plant data that paybacks of multi-skilling are not significant beyond 20% concentration of multi-skilled workforce in a given crew. In another study, Wongwai and Malaikrisanachalee (2011) proposed an alternative resource substitution approach through an augmented heuristic algorithm. It was shown by a number of case studies that this approach achieved shorter project durations as compared to other scheduling approaches. For optimization of linear scheduling problem Liu and Wang (2012) developed a constraint programming (CP) based optimization model along with heuristic rules and solved a standard bridge example available in the literature. The model also allowed the interruptions between two similar but repetitive activities. By this study managers can suitably select an

appropriate strategy for selecting crew and also ascertain the timing for temporary hiring of workforce.

It is worth mentioning here that the problem of scheduling with multi-skilled resources can also be classified as a special case of the multi-mode RCPSP (MMRCPSP) in which the permissible number of modes for each activity corresponds to the possible number of subsets of staff members that can perform the said activity. This number can be exceptionally large and hence traditional methods to solve MMRCPSP (Reyck et al., 1999, Josefowska et al., 2001) cannot be applied directly to MSRCPSP. More specifically, each activity is assumed to be processed in multiple modes such that activity duration in a particular mode is inversely proportional to the number of resource required by the activity. In literature the above problems are associated by the term 'elastic task modeling'. The term elastic is used as an analogous to metal spring elasticity where the length of a spring is inversely related to the force applied. Using same principle, the activity durations are in inverse relation to the allocated resources capacity.

For the above mentioned problem type, Kadrou and Najid (2007) proposed a novel tabu search algorithm (TS) with an objective of the minimization of overall project duration. The TS was embedded in decomposition based heuristic (DBH) with the purpose of reducing the search space. In order to test the behaviour of the developed algorithm, different standard benchmark instances were used and results were found competitive. In another work under this problem class, Santos and Tereso (2010) introduced a multi-objective cost-based mathematical model in which each activity required only one unit of resource but this resource may be employed at any of its specified levels.

Although notion of skill has been extensively applied in a lot of works related to personnel assignment, the literature in multi-skill project scheduling is still rare. The next section reviewes few of the contributions in this area that has steered and motivated other researchers to devise more efficient solution approaches for this complex problem.

2.3.2. Multi-skill resource-constrained project scheduling problem

This section specifically review the works in which researchers have applied the concept of human resource flexibility (termed as multi-skill resources in this context) in the project

scheduling environment. As explained earlier in chapter 1, unlike the conventional RCPSP, the resources in a MSRCPSP master one or more types of skills out of the several available skills. The resources are to be allocated from a pool of staff members with each member mastering one or more skill types. The scheduling decision, therefore, is twofold i.e. allocation of a particular resource to an activity; and the specific skill for which the resource is allocated. The usual objective of a typical MSRCPSP is to minimize the project makespan; however, in recent years researchers have also tackled the multi-objective cases of this problem. In the next section some of the recent contributions by researchers for both the single-objective and multi-objective cases of this problem are discussed.

2.3.2.1. Single-objective MSRCPSP

The motivation for the MSRCPSP was derived from considerable research in the field of workforce planning and staff scheduling mentioned in previous section. For example in the nurse rostering problem (Ernst et al., 2004), all needs of different working shifts have to be satisfied from the team of employees by giving fair consideration to employee's individual preferences and constraints. Similarly, in the problem of course time tabling (Alvarez et al., 1996) the concept of multi-skill staff was harnessed wherein a feasible time table for each lesson was determined respecting the constraints of teachers' availability and classrooms. To the best of the knowledge, the work by Néron (2002) represents the formal introduction and initial contribution in the area of the MSRCPSP. Later, Bellenguez and Néron (2004) extended the concept by introducing hierarchical levels of skill abilities where each staff member was assumed to possess the required skill(s) at different levels, thus imposing more complexity in the basic MSRCPSP. The objective chosen was the minimization of the makespan. To solve this complex problem, authors introduced the two lower bounds adapted from the lower bounds already available in literature for the RCPSP.

In another work, based on exact approach for this problem, Bellenguez and Néron (2007) introduced a branch-and-bound method which can tackle problems of small and average size instances up to 32 activities. There were no hierarchical levels for skills considered in the instances but a finite number of unavailability periods were assumed for the employees. After

comprehensive experimentations, it was established that neither the precedence junction nor the resource junction can be linked directly with the difficulty level to solve an instance.

Correia et al. (2012) modeled the MSRCPSP as a mixed-integer linear programming (MILP) and proposed sets of additional inequalities so as to solve the small and medium sized instances without the use of off-the-shelf general solver. The other contribution of this work can be seen in the form of developing standard benchmark instances for the MSRCPSP by taking motivation from instance characteristics mentioned in the project scheduling problem library (PSPLIB) (Kolisch and Sprecher, 1997). Analogous to a RCPSP instance, three fundamental characteristics of an MSRCPSP instance were identified namely the network complexity (NC), the skill factor (SF) and modified resource strength (MRS). Almeida et al. (2015) later proposed a detailed and comprehensive procedure for developing the various instances of MSRCPSP with varying complexity. In this thesis, this procedure will be used and coded to generate instances of required characteristics for computational studies. In the light of this aspect, a more detailed and elaborate methodology on the instance generation procedure will be presented in chapter 4.

Most of the above works were largely concerned with proposing optimal or exact solution approaches for the MSRCPSP in terms of effective lower bounds or enhancing theoretical MILP models by imposing additional inequalities. However, as already mentioned, the MSRCPSP being an extension of the RCPSP, is also NP hard which limits the possibility of solving large or practical instances in reasonable computational time. Moreover, the problem is a special case of multi-mode-RCPSP where the number of modes is given by the permutations or different subsets of staff members that may be assigned to a candidate activity and this may be exceptionally large. For example, a project instance with only three skills types and with only one level for each skill, among a pool of 10 employees may have 1000 different modes for executing some of the activities. Nevertheless, the initial solution techniques based on exact approaches are of great importance not only because of their academic insight but also for their ability to provide a reference or benchmark for evaluating the different newly designed heuristics.

Among few of the early applications of (meta) heuristics for the MSRCPSP, Kazemipoor et al. (2012) formulated the MSRCPSP as a novel linear mixed-integer programming (MILP) problem and proposed an efficient simulated annealing (SA) algorithm for finding its solution. In order to

test the behaviour of the proposed SA, the authors compared it from the optimal solutions obtained by LINGO solver for small sized instances. The results were quite promising in the sense that optimal or near-optimal solutions for small instances as well as 'good' solutions for larger instances were determined in a reasonable time using the SA.

In the work of Almeida et al. (2016), the problem was solved using a heuristic approach based on eight different priority rules and a modified parallel scheduling scheme. The author introduced two new concepts namely *resource weight* and *activity grouping* to accommodate multi-skilled nature of the resources. More specifically, each resource was assigned a weight depending upon the number of skills it masters as well as scarceness and demand of the particular skill. The activity grouping was made feasible by transforming the resource assignment problem into a minimum cost network flow problem denoted as MC-NFP. The computational experiments were performed on the test instances proposed by Correia et al. (2012) and deviation from the length of critical path was determined for various instances. It was concluded in the study that no rule performed significantly well simultaneously for all values of NC, SF and MRS. However, the hybrid of two activity priority rules namely LST (Latest Start Time) and GRPW (Greatest Rank Positional Weight) designated as LST+GRPW was found to exhibit better results among other ten alternatives chosen in the study. In another work, Almeida et al. (2018) employed a biased random-key based GA (based on Mendes et al., 2009) for the MSRCPSP and improved their previous results obtained by the priority rule-based heuristics.

A notable research can be comprehended by some authors in multi-skilled resource based scheduling area by considering the integration of resource investment problem (RIP) with the multi-skilled project scheduling. In a RIP, the limits on renewable resources have been treated as decision variables and the goal is to minimize the procurement cost of renewable resources with constraint on project completion deadline. Under this motivation, Javanmard et al. (2017) proposed two MILP models to minimize the total requirement cost for the different skill levels. To tackle this complex problem, the author proposed an innovative solution representation scheme and two metaheuristic approaches namely GA and particle swarm optimization (PSO) whose parameters were calibrated by response surface methodology (RSM). The performance of the algorithms was evaluated for different runs and compared with those obtained by GAMS

software. The results confirmed the applicability of the proposed methodology especially for real-life problem scales.

Recently, Myszkowski et al. (2018) applied a hybrid Differential Evolution and Greedy Algorithm (DEGR) with specialized indirect representation for the MSRCPSP. It is interesting to mention that the authors transformed the solution space from discrete (typical for the MSRCPSP) to continuous one (which is generally a feature of DE approaches). The authors compared the performance of DEGR with other reference methods such as hybrid ant colony (HAntCO), multiStart Greedy and GRASP on 36 benchmark instances which later became available in literature (Myszkowski et al. 2018). The results confirmed the robustness of the proposed algorithm. In fact, for 28 instances of iMOPSE dataset the best-known solutions were obtained by DEGR.

2.3.2.2. Multi-objective MSRCPSP

The various complex and realistic optimization problems in daily life or engineering applications demand optimization of more than a single objective such as maximizing the profit while minimizing the consumption of raw materials, improving a product quality and lowering the production cost or choosing a comfortable car at the minimum price and so on. These objectives are mutually conflicting or contradictory with each other and hence conventional solution approaches for single-objective optimization no longer prove feasible to solve these problems. There are two common techniques available in literature to handle the multi-objective problems; one is *priori* approach that combines the individual objective functions into a single function by allocating weights based on judicious and conceptual skill of practitioners and other is *posteriori* approach in which set of Pareto optimal solutions are obtained that are nondominated with respect to each other. For a holistic and comprehensive view of multi-objective optimization principles and theory, reader is referred to book by Kalyanmoy Deb, 2001.

In this section, the contributions made by researchers in the field of multi-objective project scheduling problems with flexible resources have been reviewed. Myszkowski et al. (2015) utilized a hybrid ant colony optimization (HAntCO) by linking classical heuristic priority rules with the ant colony optimization (ACO) philosophy to solve a special case of MSRCPSP.

However, it was assumed that each task can be executed exactly by one of the resource out of the available pool of resources. The authors proposed a novel method for updating the value of pheromone by memorizing the best and worst solutions by ants. Two objectives were chosen in this study namely duration optimization and cost optimization and tests were performed on the iMOPSE (intelligent multi-objective project scheduling environment) data set containing 36 instances created as benchmark instances for the bi-objective MSRCPSP. It was found that HAntCO outclassed classical ACO for both the objectives.

For the same type of multi-objective problem, Zheng et al. (2015) developed a teaching-learning-based optimization (TLBO) algorithm incorporating a task-resource list-based encoding scheme and a left-shift decoding scheme. To enhance the local intensification of the TLBO, the authors also appended it with a reinforcement phase. The parameter setting based on Taguchi's design of experiment was carried out to fine tune the algorithm and its performance was found effective than the HAntCO proposed by Myszkowski et al. 2015.

Similar to the MSRCPSP, the multi-objective MSRCPSP has also been realized by some researchers as a special case of MMRCPSP. For example, Maghsoudlou et al. (2016) studied the multi-skill multi-mode resource-constrained project scheduling problem with three objectives: (1) minimization of makespan of the project (2) minimization of total cost incurred in allocation of workers to required skills, and finally (3) maximizing the overall quality of activities which are executed. The authors encoded the solution into a novel chromosome structure that guaranteed the feasibility of solutions. A multi-objective invasive weeds optimization algorithm (MOIWO) was developed in this work to solve this tri-objective problem and results were compared and found competitive to the two other metaheuristic algorithms namely non-dominated sorting genetic algorithm (NSGA-II) and multi-objective particle swarm optimization (MOPSO) developed for the purpose. In another work by same authors (Maghsoudlou et al., 2017) three different modifications of Cuckoo Search algorithm (CS) were proposed for the bi-objective MSRCPSP problem with two conflicting objectives: (1) minimization of the total cost of processing the activities and (2) minimization of the reworking risks of the activities.

Recently, Wang and Zheng (2018) employed a knowledge-guided multi-objective fruit fly optimization algorithm (MOFOA) for the bi-objective MSRCPSP. The two objectives

considered were minimization of project makespan and total cost simultaneously. The minimum cost rule method was used for the initialization of the feasible solutions and both smell and vision-based search were adopted carry out the multi-objective optimization. The numerical tests conducted showed the effectiveness of the MOFOA over HACO (Myszkowski et al. 2015).

2.4. Summary and research gaps

In this chapter a thorough literature review related to the pertinent areas of project scheduling within the scope of this work is presented. Focus is given to the concept of flexibility and its elucidation in the area of project scheduling. The notion of skills in MSRCPSP is also highlighted and a brief outline of its solution approaches for both single and multi-objective optimization is presented. Although there has been a phenomenal growth in the classical RCPSP, there still seem few potential research gaps which have been mentioned below:

1. Most of the research has been focused on basic resource categories that is renewable, non-renewable and double constrained resources but adequate research about other recently introduced categories of resources such as dedicated resources, spatial resources, adjacent resources, cumulative resources, reusable resources, synchronizing resources, multi-skilled resources, changeover resources etc. or their combinations has not been tackled in sufficient length in the literature.
2. There seems a good scope of work in the area of modifying the activity characteristics in a project such as time-switch constraints, preemptibility, time-varying tasks, crashing etc. Also, a combination of these real-life activity variations can be studied in detail for a given scenario.
3. Unlike deterministic project scheduling, the work done in project scheduling under uncertainty is still rare. There is a good scope of developing efficient approaches for reactive scheduling, stochastic project scheduling, and fuzzy project scheduling and robust (proactive) scheduling.
4. Furthermore, research in sensitivity analysis in project scheduling environment has not been explored to the extent as compared to shop floor or production system scheduling. It can be fruitful to estimate the effect of change in parameters on the quality of solution obtained from an exact or (meta) heuristic approach.

5. More specifically, a lot of heuristic and metaheuristic approaches are available for the RCPSP in literature but efficient metaheuristics for the multi-skill resource-constrained project scheduling problem (MSRCPSP) are still rare.
6. Although hierarchical levels of skills exist in literature but to the best of the knowledge there is no reported work/algorithm involving different proficiency levels of skills attained by staff members (resources).
7. There is a substantial literature addressing the multi-objective RCPSP, however a very less work has been reported in literature regarding multi-objective multi-skill resource constrained project scheduling problem (MO-MSRCPSP) with consideration of proficiency levels in skills attained by staff members or human resources.

This thesis aims to fill some of these gaps, particularly related to the application of multi-skilled resources for achieving ‘good’ solutions for single and multi-objective cases. Moreover, the concept of varying skill proficiencies of staff members will also be taken into consideration during multi-objective optimization.

Chapter 3

Teaching-Learning-Based Optimization Algorithm for the RCPSP

Chapter 3

Teaching-learning-based optimization algorithm for the RCPSP

3.1 Introduction

As mentioned in chapter 1, in this work a modified version of teaching-learning-based optimization algorithm (TLBO) will be developed which is a recently introduced metaheuristic by Rao et al. (2011). The TLBO is a population based metaheuristic that mimics the teaching-learning process commonly seen in classrooms. It has been successfully applied on mathematical benchmark functions and mechanical design optimization problems of continuous nature. Interestingly, the algorithm has been reported to have high convergence rate and inherits a merit of having few algorithm specific parameters to tune (Rao et al., 2011). Inspired by the performance of the TLBO on continuous non-linear problems, a lot of researchers have applied it on discrete optimization problems in recent years. For a more comprehensive review of TLBO and its application readers are referred to a recent survey by Zou et al. (2018).

Under above motivation, this chapter presents the framework and details of implementation methodology of modified TLBO that has been conceptualized for the resource-constrained project scheduling problem (RCPSP). Inspired by the study of Zheng and Wang (2014), in addition to teacher and learner phase, the proposed work also applies two additional phases namely self-study phase and examination phase for improving the exploration and exploitation capabilities of the algorithm.

In this chapter, a formal introduction of the RCPSP is exhibited first with an illustrative example. It is followed by basic philosophy or working mechanism of the conventional TLBO. Afterwards, the encoding and decoding procedure designed to incorporate TLBO philosophy in this problem is explained in detail. Finally, the comparative results with other state-of-the art approaches available in literature to solve the RCPSP have been shown to validate the developed algorithm.

3.2. Problem description

Consider a directed acyclic graph $G(J,E)$ to represent a non-preemptive single mode resource constrained project scheduling problem (RCPSP) wherein J denotes the activities and E represents the precedence relations. The precedence relations which exist amongst activities due to technological requirements force each activity j to be scheduled after all its immediate predecessors (given by set E) are completely finished. There are total $n+2$ activities in the project represented by set J, $J= \{1, 2,\ldots\ldots n+2\}$ and K renewable resources given by set R so that $R= \{1, 2, \ldots\ldots, K\}$. It is important to mention that activities 1 and $n+2$ are dummy in nature i.e. they do not consume time and resources and represent start and finish activities of the project.

Each activity $j \in J$ needs $r_{j,k}$ units of resource k ($k \in R$) during each time period of its non-preemptive duration d_j (obviously $d_1 = d_{n+2} = 0$ and $r_{1,k} = r_{n+2,k} = 0$). Unlike PERT/CPM approach, in RCPSP a practical assumption that per-period availability of each resource $k \in R$ is limited and constant given by R_k is considered. The values of the parameters R_k , d_j and $r_{j,k}$ (total availability of resources, processing time of activities and resource requirement by activities) have been considered as deterministic, integer and non-negative. Let f_j denotes the finish time of activity j , then a schedule or solution of a RCPSP can be given as a vector of activities' finish times, $S=\{f_1, f_2, \ldots\ldots, f_{n+2}\}$.The objective function is to minimize the makespan f_{n+2} i.e. the finish time of last activity.

Mathematical formulation

Minimize f_{n+2} (3.1)

Subject to: $f_j - f_i \geq d_i \quad (i,j) \in E$ (3.2)

$\sum_{j \in A(t)} r_{j,k} \leq R_k \quad t \geq 0 \; ; k \in R$ (3.3)

$f_j \geq 0 \quad j \in J$ (3.4)

Eq. (3.1) represents the objective function which is to minimize the makespan of the schedule. The precedence constraints between activities are handled by Eq. (3.2). As explained earlier, the resources are limited in the RCPSP which is enforced by Eq. (3.3), where $A(t) = j \in J \mid f_j - d_j \leq t < f_j$ represents the set of activities which are active or being processed at time t. Finally, the constraint for decision variables to be non-negative integers is represented by Eq. (3.4).

Figure 3.1 depicts an example of a typical project having eight activities with activity 1 and activity 8 as dummy source and sink activity respectively. The objective is to generate a schedule having start (or finish) times of all activities respecting the precedence relations and per period availability of (renewable) resource R_1 with maximum per period availability of 6 units. A feasible (also optimal in this case) solution with a makespan of 15 time units is shown in Figure 3.2.

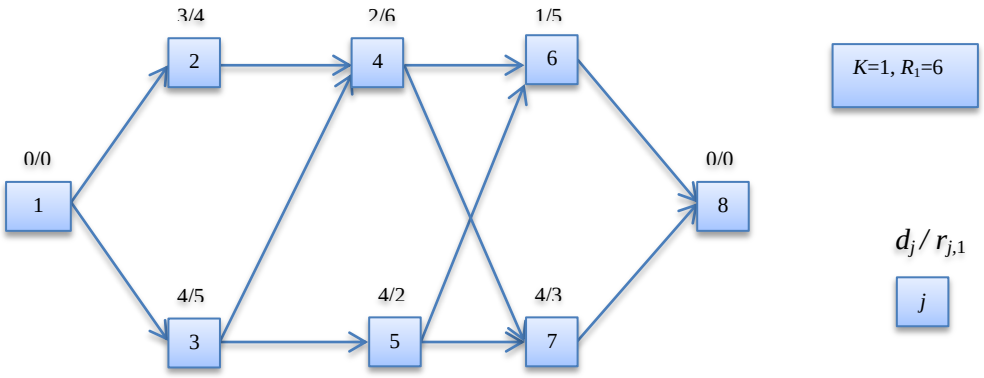


Figure 3.1: A project instance for the RCPSP

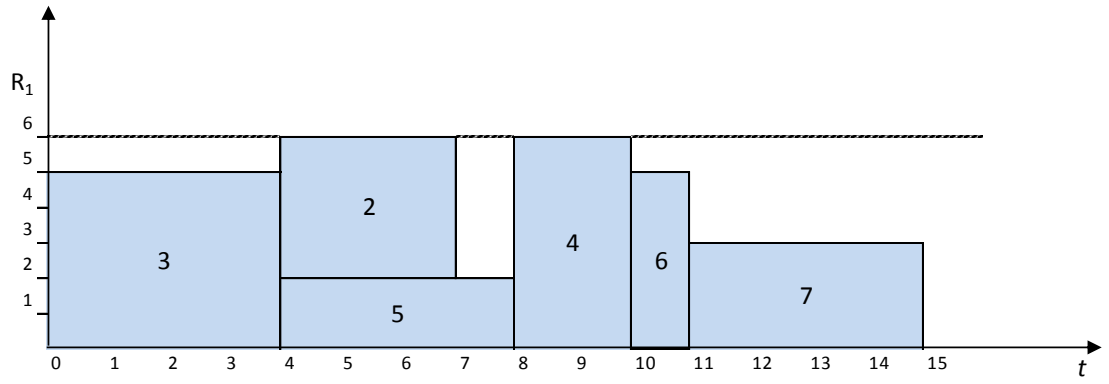


Figure 3.2: A feasible solution for the illustrative instance

3.3. The philosophy of TLBO

The TLBO algorithm is inspired by the teaching-learning process commonly seen in classrooms. It is a population- based algorithm and utilizes a group of students called learners as initial population to reach a global optimum. Figure 3.3 shows a simple model to understand its philosophy. The dotted curve represents the performance level or marks obtained by different learners in a class with mean $Mean_1$.The highest marks $Teacher_1$ shown in extreme right of the curve corresponds to that of a teacher because it is a teacher who is recognized as the most intelligent in any class. As the teacher imparts his/her knowledge to the learners, it is assumed that the mean $Mean_1$ of the class increases to a new value say $Mean_2$ by some probability depending upon the capabilities of the teacher and learners. At this stage to continue the teaching-learning process, the learners require a new teacher whose knowledge level is appreciably higher than this new mean $Mean_2$. The best learner from this new population having knowledge level $Teacher_2$ is selected as new teacher for next iteration. In this way the mean of class gets improved in subsequent iterations.

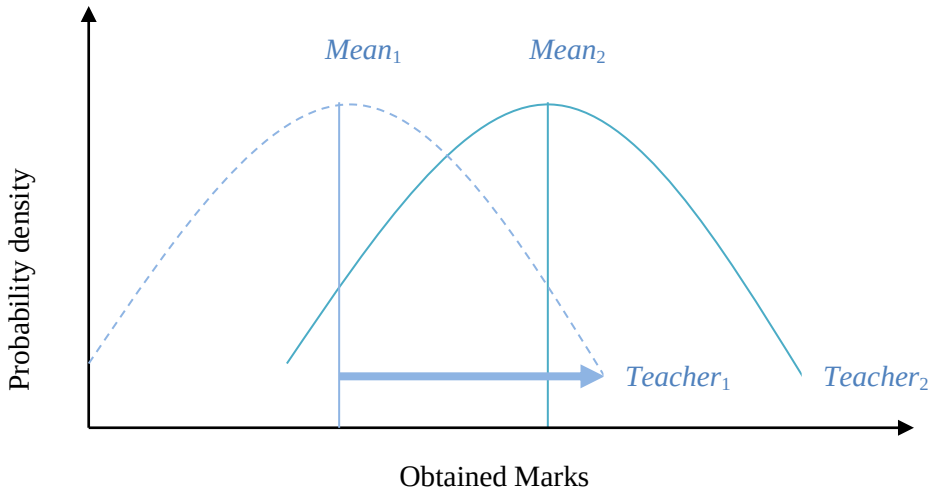


Figure 3.3: Model to show TLBO philosophy

3.4. The proposed TLBO for the RCPSP

As explained in section 3.1, the TLBO is primarily conceptualized for mechanical design optimization problems having design parameters continuous in nature. However, the RCPSP being a discrete optimization problem, some modifications need to be done in solution representations and other parameters before TLBO is applied on it. The framework of the proposed TLBO is shown in Figure 3.4. It can be seen that in addition to the conventional teacher and learner phase, the proposed TLBO also encompasses two additional phases namely self-study and examination phase.

The details of the various features of TLBO proposed in this work have been summarized under following steps:

Step 1: Initialization

Like other population based algorithms, the TLBO also starts with a group of solutions which is known as a class of 'learners (or students)' and treated as initial population. A group of learners is generated randomly using the parameterized regret-based biased random sampling (RBRS) method discussed later.

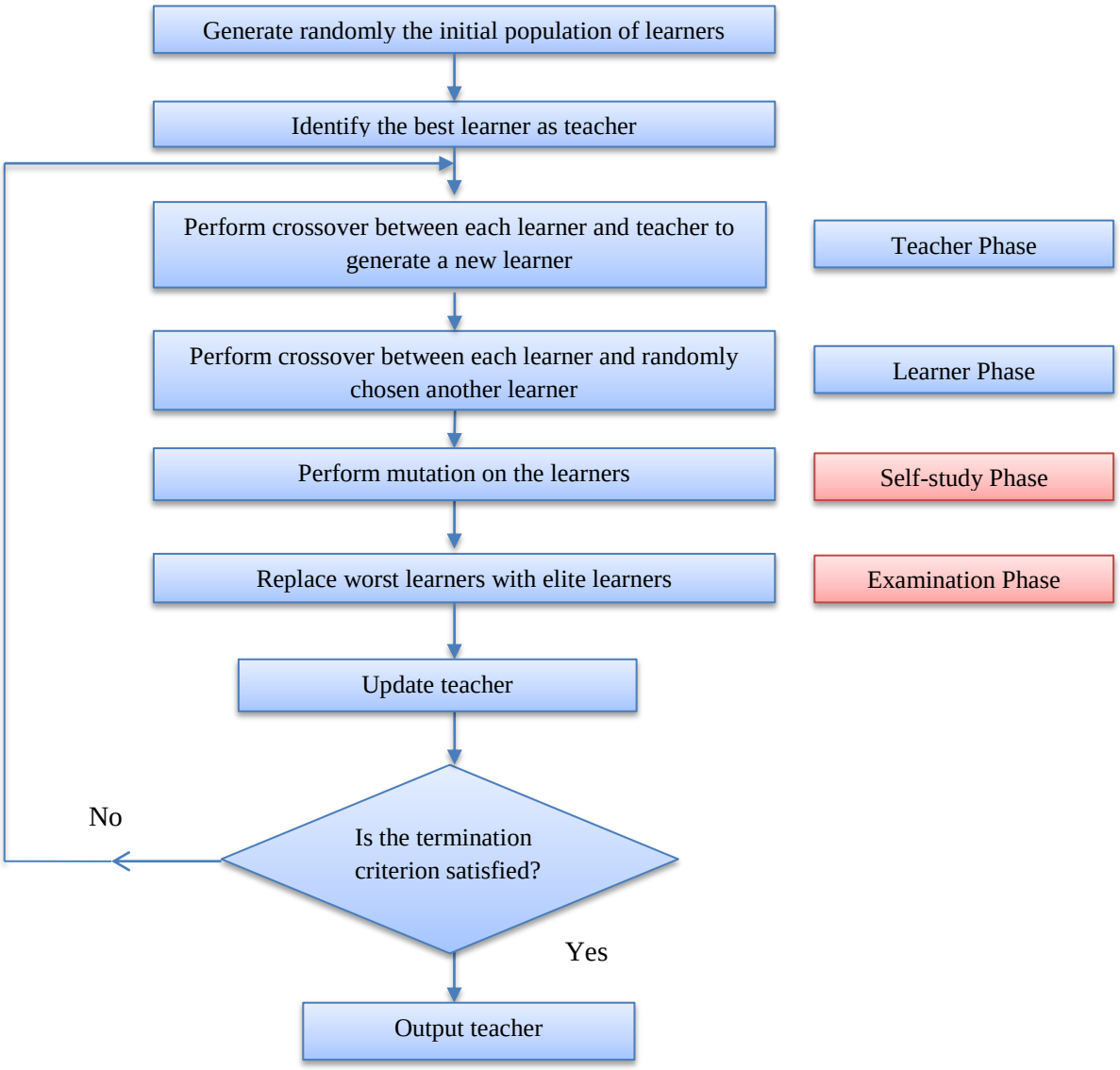


Figure 3.4: Framework of the proposed TLBO

Step 2: Identification of teacher

The fitness function (makespan in this case) is calculated for all these learners using serial schedule generation scheme (SGS) proposed by Kolisch, 1996. Since a teacher is considered as the most knowledgeable person in any class, the best learner having minimum makespan is designated as teacher.

Step 3: Application of teacher phase

In this phase the teacher tries to transfer his knowledge to each learner of the class. To realize this phase, crossover mechanism analogous to GA is employed which is explained in section 3.4.3 with an illustrative example.

Step 4: Application of learner phase

During this phase, learners are supposed to learn interactively through informal communications, mutual discussions, presentations etc. More precisely, each learner is subjected to have crossover with other randomly chosen learner. The new learner is retained in the class if its fitness is improved else previous learner is kept as usual.

Step 5: Application of self-study phase

This phase is appended in conventional TLBO to increase its exploration capabilities. Similar to the mutation concept of GA, the position of chromosomes (priorities of activities in this case) have been varied with some probability. As a result of this phase, possibility of algorithm being trapped in local minima is substantially reduced.

Step 6: Application of examination phase

This is another additional feature which is incorporated in conventional TLBO (Zheng and Wang, 2015) to increase its exploitation capabilities. To realize this phase, few best learners in a population replace the corresponding number of worst learners. The concept is akin to elitism in GA which guarantees that good features of a population are not lost rather transferred into subsequent populations. After the stopping criterion is met, the best learner in final iteration is reported as the solution of the problem.

Figure 3.5 shows a pseudo-code developed to incorporate the above-mentioned steps of the TLBO proposed in this work.

```
Input: Initialize class_size (number of learners in the class) , num_iter (Number of iterations),
                prob_ss ( probability of self-study) and  num_elite ( elite size i.e. number of students to be failed)
Output: The teacher for iteration size= num_iter
1: Begin
2:      Generate the initial population using RBRS method
3:      Calculate the fitness i.e. the makespan of each learner using SGS method
4:      Designate the learner with minimum makespan as the teacher
5:          while (stopping condition is not met);
6:              for i=1: class_size
7:                  Perform 2-point crossover between each learner and student          % teacher phase
8:                  Evaluate the new learner
9:                  if  makespan new learner < makespan old learner
10:                     Replace the old learner with new learner
11:                 end
12:                     Perform 2-point crossover between a learner with another random learner    % Student phase
13:                 if  makespan learner-2 < makespan learner-1
14:                     Replace the learner-1 with learner-2
15:                 end
16:                 Perform mutation with prob_ss                                      % Self-study phase
17:                 Retain num_elite learners in the class                             % Examination Phase
18:                 Calculate makespan and update the teacher
19:         end while
20: end
```

Figure 3.5: Pseudo-code for the proposed TLBO

3.4.1. Solutions encoding and decoding

The encoding and decoding schemes are one of a vital decision for application of any metaheuristic. In fact, the quality of solutions obtained by an algorithm is largely affected by the way a solution is encoded. There are various types of representations used by researchers in literature for encoding of the RCPSP such as shift vector representation, random key representation, priority rule representation and activity list representation etc. (Alcaraz and Moroto, 2001)

Based on the very promising and effective results obtained from activity list (AL) representation based metaheuristics (Kolisch and Hartmann, 1999), each individual is represented (called

'learner' in TLBO) as precedence feasible list of the activities where no activity appears before any of its predecessors. More specifically, a 'learner' is any precedence feasible permutation of activities represented as $\lambda = (j_1, j_2, \ldots\ldots, j_{n+1}, j_{n+2})$. The index values in this list represent the corresponding priorities of activities for their execution. As mentioned earlier, the solution of the RCPSP is to determine the finish (or start) time of each activity so that both the precedence and resource constraints are satisfied and makespan is minimized. A serial schedule generation scheme (SGS) has been chosen to transform a given activity list λ into a schedule. The motivation behind this choice is that search space in SGS comprises of active schedules and essentially contains an optimal solution (Kolish, 1996). Figure 3.6 gives a pseudo code for the SGS procedure developed in this work.

Define N=total number of activities, j =1, 2,.., n +2
Determine E(g): a set of all precedence feasible activities which can be started at stage g
1: *for* g=1 to N
2: Calculate the eligible set E(g)
3: Select one j ∈ E (g)
4: Schedule j at the earliest precedence and resource feasible start time
5: *end for*

Figure 3.6: Pseudo code for the SGS

3.4.2. Initial population

A parameterized regret-based biased random sampling (RBRS) method as mentioned in Kolisch and Drexl (1996) is used to generate the initial population. Unlike priority rule based multi-pass heuristics methods which produce the same schedule when applied each time, RBRS is a sampling method which assigns a probability $\psi(i)$ to each activity in the decision set D for being selected at each stage of the SGS such that $\psi: i \in D \rightarrow [0,1]$. A regret value ($\tau_i$) for each activity i is computed by comparing the priority value $\upsilon(i)$ of activity i with the worst priority value $\upsilon(j)$ of the decision set activities as per the following equation:

$$\tau_i = \max \upsilon(j) - \upsilon(i), \qquad j \in D \tag{3.5}$$

The parameterized probability mapping $\psi(i)$ is then determined as follows:

$$\psi(i) = \frac{(\tau_i+\varepsilon)^{\alpha}}{\sum_{j\in D}(\tau_j+\varepsilon)^{\alpha}} \quad (3.6)$$

The value of parameters ε and α has been fixed as 1 in the proposed algorithm as these values are known to exhibit good results in literature (Kolisch and Drexl, 1996). On similar lines, latest finish time (LFT) rule is employed to determine the regret values in the decision set *D* as this has been reported as one of the best priority rule in the literature. To understand the procedure discussed above an illustrative example is presented in what follows next.

- *An illustrative example for implementation of RBRS method along with LFT rule*

Problem Statement: Let at any stage the decision set contains three activities i.e. *D*= {1, 2, 3} and one of these activity has to be chosen as per RBRS method.

Given parameters: Let the latest finish times of the activities LFT_1=13; LFT_2=16; LFT_3=11 and values of ε=α=1

Procedure:

Step 1: Calculate the worst priority value among all activities in *D*;

υ_{worst}= max ($\upsilon_1, \upsilon_2, \upsilon_3$) = max (13, 16, 11) =16

Step 2: Calculate the regret value (τ_i) for each activity *i* using equation (3.5);

τ_1= 16-13 = 3;

τ_2= 16-16 = 0;

τ_1= 16-11 = 5;

Step 3: Calculate the parameterized probability ψ (*i*) for each activity i using equation (3.6);

$\psi(1) = ((3+1)^1) / \sum((3+1)^1 + (0+1)^1 + (5+1)^1) = 0.363$

Similarly, ψ (2) = 0.091 and ψ (3) = 0.545

Step 4: Calculate the parameterized probability mapping;

ψ (i) = (0.363, 0.091, 0.545)

Step 5: Calculate the cumulative parameterized probability mapping;

Cum (ψ (i)) = (0.363, 0.454, 0.999)

The activity selections can now be made by generating any random number (*rand*) and mapping it with Cum (ψ (i)). For example if *rand*= 0.71, activity 3 has to be selected and so on.

3.4.3. The teacher and learner phase

This section explains the mechanism of applying two conventional phases of TLBO for the RCPSP. Firstly, the fitness function (makespan) for all the individuals (learner) of the initial population is determined using SGS and set the best learner corresponding to minimum makespan as the teacher. As mentioned earlier, the teacher tries to transfer his or her knowledge to all the learners in the class (population). To realize this process the crossover technique as used in GA is employed. The two popular crossover versions namely 1-point and 2-point crossovers (Hartmann, 1998) have been considered in this work. This is primarily because these mechanisms preserve the precedence feasibility of the ALs. The following example shows the two-point crossover method for a teacher to transfer his knowledge into the learner:

Let λ_1 represent a learner and λ_2 be the best learner or a teacher in a given population. Two random integers u_1 and u_2 have been generated in the range [1, n]. Using Equations (3.7) to (3.9) and the values u_1 and u_2, a new activity list λ_{new} is determined. Figure 3.7 shows an example of implementation mechanism of 2-point crossover or teacher phase for u_1 = 4 and u_2= 7 in an AL having 10 activities in all. The learner so obtained after teaching is selected if it gives better makespan else previous learner is retained.

In the learner phase, as explained earlier, a learner tries to improve his or her knowledge by mutual interactions or discussions with other randomly chosen learner. To realize this phase, once again the crossover between the two randomly chosen learners is employed. The new

learner (i.e. new AL) is accepted in the new population if it gives a better makespan; else previous learner is retained in the population.

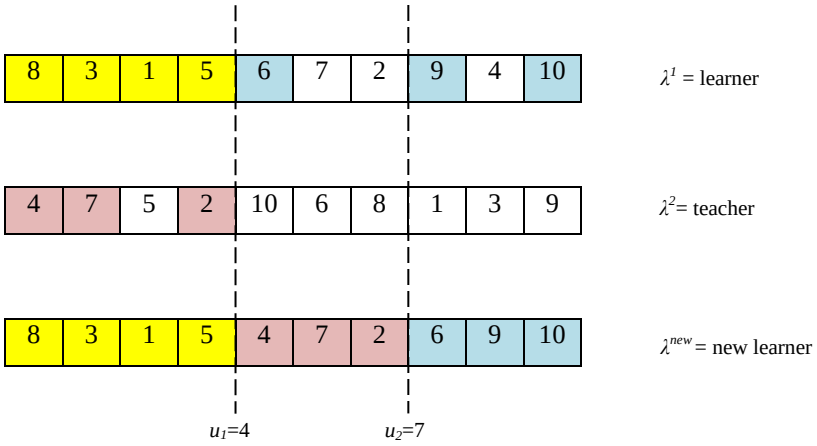


Figure 3.7: Mechanism of 2-point crossover in teacher phase

$$\lambda^{new}_j := \lambda^1_j , \quad 1 \le j \le u_1 \tag{3.7}$$

$$\lambda^{new}_j := \lambda^2_k , \quad k=\min \{k \mid \lambda^2_k \notin \{ \lambda^{new}_1, \ldots\ldots\ldots\ldots, \lambda^{new}_{u1} \}\}, \; u_1 + 1 \le j \le u_2 \tag{3.8}$$

$$\lambda^{new}_j := \lambda^1_k , \quad k=\min \{k \mid \lambda^1_k \notin \{ \lambda^{new}_1, \ldots\ldots\ldots\ldots, \lambda^{new}_{u2} \}\}, \; u_2 + 1 \le j \le n \tag{3.9}$$

3.4.4. The self-study phase

As already mentioned, the concept of self-study has also been incorporated in the proposed TLBO to increase its exploration capabilities. This phase is analogous to mutation in GA which is known to avoid premature convergence and thereby facilitating exploration search. It is a general phenomenon that students may improve their grades through self- study before examination. Two different mutation operators have been implemented to realize this phase in the developed algorithm. The first one is adapted from Boctor, 1996 in which each activity in the AL is shifted at some randomly chosen position with a probability *prob_ss*. The position chosen for the activity should be higher than any of its predecessors and lower than any of its successors to ensure that new AL is also precedence feasible.

The second mutation operator is adapted from Hartmann, 1998 in which pairwise exchange of activities is there. For a given AL represented as $\lambda = (j_1, \ldots. j_i, \ldots\ldots\ldots j_n)$, the activities j_i and j_{i+1} have been exchanged with a probability *prob_ss* provided the precedence feasibility is maintained.

3.4.5. The examination phase

This phase restricts the non-performing learners (individuals) to enter the next class (population). To keep the population size uniform some elite learners represented by *num_elite* replace such non-performing learners. The concept seems to be akin to elitism which has been widely used in genetic algorithms and other metaheuristics. Three different values of *num_elite* from literature have been considered during parameters tuning of the algorithm.

3.5. Computational experiences

The proposed algorithm has been coded in MATLAB R2008a (Version 7.6) and run in Windows 7 having 2.0 GHz processor and 2.00 GB RAM. To test the performance of the algorithm, well-known problem instances sets proposed by Kolisch & Specher (1997) and available in the Project Scheduling Problem Library (PSPLIB) at http://www.bwl.uni-kiel.de/Prod/psplib/ have been used in this work. More specifically, three problem instances sets namely J30, J60 and J120 which contains project instances with 30, 60 and 120 non-dummy activities respectively with varied levels of network complexity, resource factor, and resource strength have been tested in this work. There are 480 instances each for J30 and J60 activities whereas J120 set contains 600. The proposed algorithm is compared for average percent deviation (Avg._Dev) from optimal values for J30 set and critical-path based lower bound for set J60 and J120. The Avg._Dev has been calculated as:

$$\text{Avg.\_Dev} = [\sum_{i=1}^{N}(makespan_i - lb_i)/\, lb_i] \,/\, N \qquad (3.10)$$

where $makespan_i$ is the total project completion time of i^{th} instance as obtained by proposed TLBO, lb_i represents the critical-path based lower bound of the i^{th} instance and N is the total number of instances in a set. For J30 problem set lb_i has been replaced by optimal makespan values for all 480 instances as available in the literature.

3.5.1. Parameters setting

As explained in Section 3.4.2, the teacher and learner phase can be incorporated by 1-point or 2-point crossover mechanisms. Also, the self-study phase is realized by two mutation mechanisms as proposed by Boctor, 1996 and Hartmann, 1998. The two mutation mechanism will be represented as BM and HM respectively in further discussion. In order to determine the best combination of these factors, both the factors each having two levels have been crossed together, thus having four different combinations to be tested. To have fair representation of the entire problem set, 48 instances (10% of total 480 instances) are randomly chosen each from J30 and J60 set. On similar lines 60 instances (10% of total 600 instances) from J120 set are chosen. Each alternative combination was made to run five times for a maximum of 500 schedules to be generated as termination criterion with a uniform population size of 80. The average percentage deviation so obtained from optimal solutions for J30 and critical-path based lower bound for J60 and J120 is exhibited in Table 3.1.

Table 3.1: Test results for different crossover and mutation mechanisms

Crossover	Mutation	Avg. Percent Dev. for 500 schedules		
		J30	J60	J120
1-point	BM	0.74	13.61	42.54
1-point	HM	0.88	14.67	43.56
2-point	BM	0.68	13.62	42.53
2-point	HM	0.83	13.78	42.37

The results revealed that 2-point crossover and Boctor mutation have performed significantly better than other combinations especially in case of J30 and J60 problem sets. Looking the overall results obtained in above table, 2-point crossover and BM have been chosen in the proposed algorithm for the further testing and parameter tuning.

After selecting the best combination of crossover and mutation mechanisms, Taguchi method of design-of-experiment (DOE) is employed to tune the other parameters of the algorithm. There are three key parameters of interest: the number of learners in a class represented by *class_size* (commonly known as population size), the self-study probability (*prob_ss*) and the number of

non-performing students to be replaced by elite students (*num_elite*). Each of these factors is considered at three different levels as shown in Table 3.2. These levels are inspired from the values considered in literature for other population based metaheuristics namely GA and PSO.

Table 3.2: Parameters selected for the DOE

Parameters	Factor level		
	1	2	3
class_size	60	80	100
prob_ss	1%	5%	10%
num_elite	4	8	12

Looking at the aspect that there are three factors at three different levels, a L_9 (3^3) orthogonal array is chosen for this experimentation having eight degrees of freedom (DOF). The Taguchi approach helps to reasonably reduce the total 27 experiments for a full-factorial design into 9 numbers of treatments. To conduct the test, 600 number of schedules is set as the stopping criterion for each J30, J60 and J120 instance set and the average percent deviation (Avg._Dev) from the optimum values for J30 and critical path based lower bound for J60 and J120 is chosen as ARV(average response variable). The results of the DOE are shown in Table 3.3.

Table 3.3: Orthogonal table and the ARV values for DOE

Exp. number	Factors			ARV for 600 schedules		
	class_size	*prob_ss*	*num_elite*	J30	J60	J120
1	60	1	4	0.74	13.17	42.57
2	60	5	8	0.64	13.02	43.64
3	60	10	12	0.65	13.38	44.35
4	80	1	8	0.74	13.07	42.40
5	80	5	12	0.59	12.82	42.83
6	80	10	4	0.59	13.03	42.31
7	100	1	12	0.62	12.90	41.70
8	100	5	4	0.70	12.65	42.06
9	100	10	8	0.66	12.86	42.63

The ARV as obtained from Table 3.3 is utilized to determine the trend of each of the three key parameters for J30, J60 and J120 instance sets. Figure 3.8 to Figure 3.10 depict the best combinations of these parameters which have been tabulated in Table 3.4.

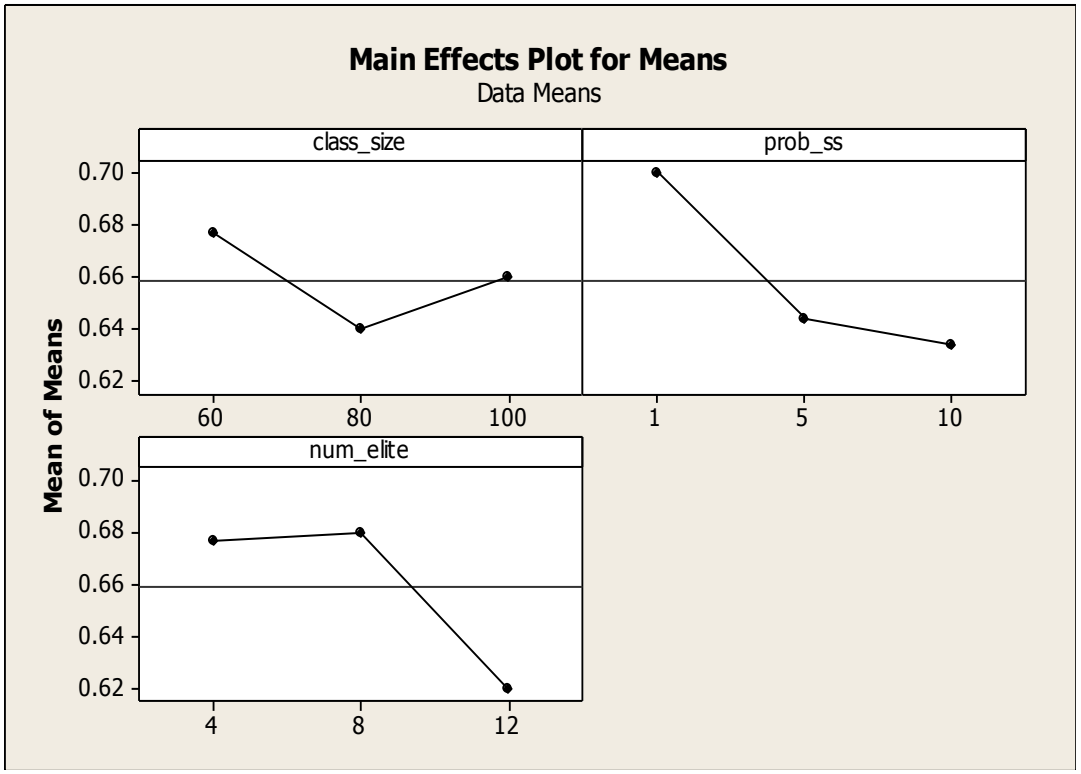


Figure 3.8: Trend of factor levels for J30

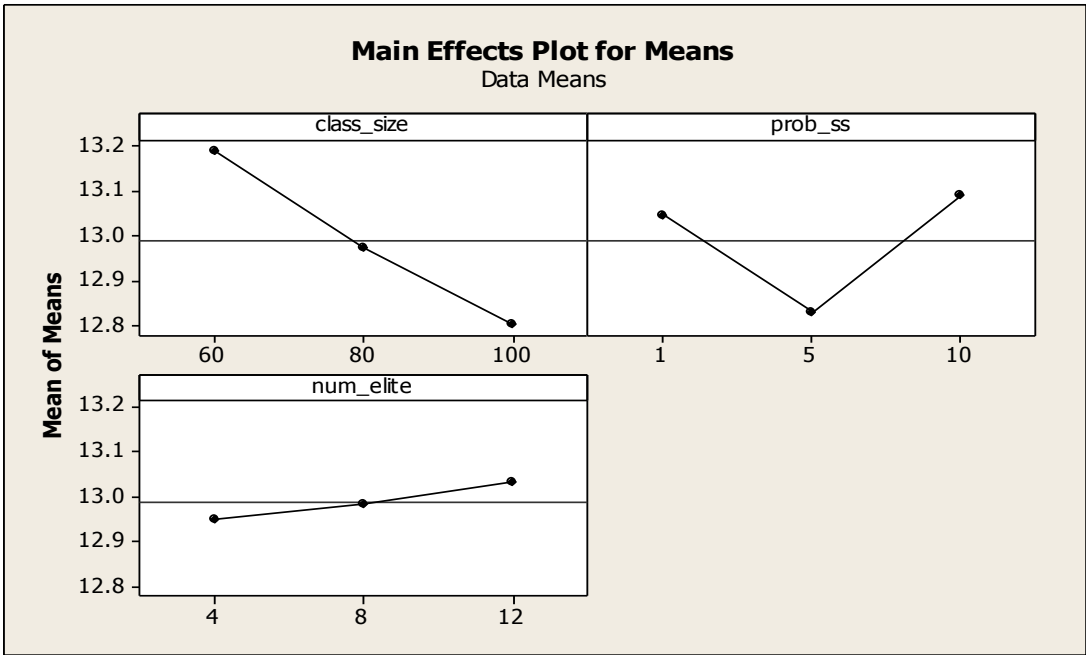


Figure 3.9: Trend of factor levels for J60

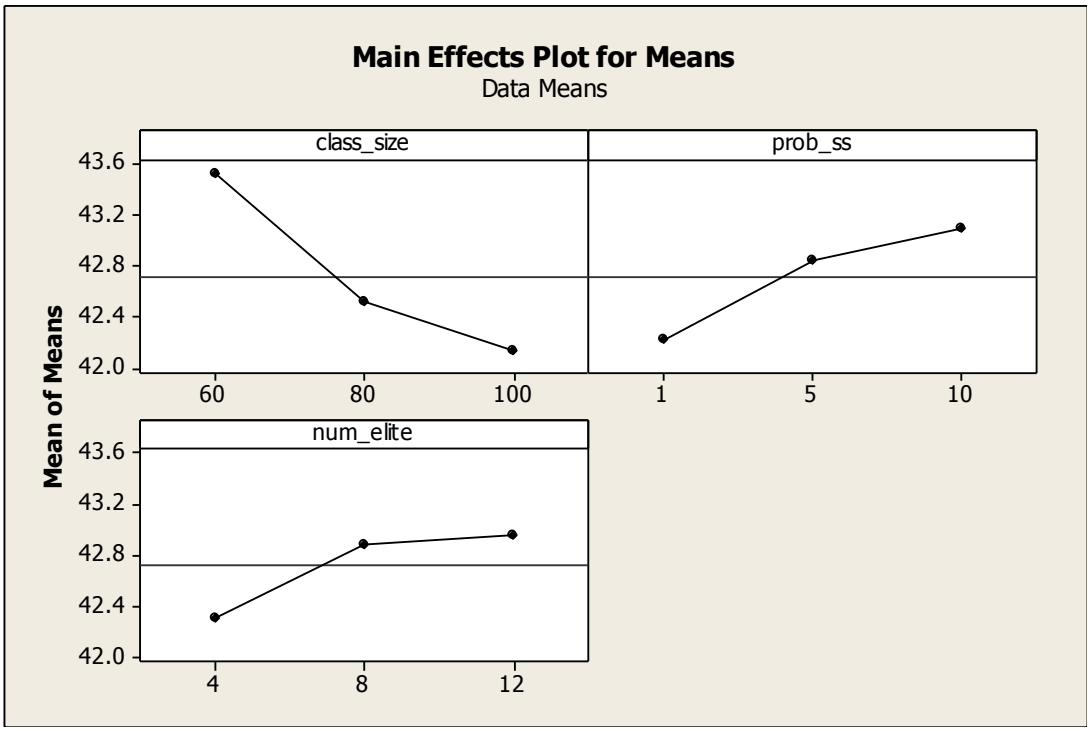


Figure 3.10: Trend of factor levels for J120

It is evident from Table 3.4 that the values of *class_size, prob_ss and num_elite* are not same for all instance sets. This may be attributed to the different network complexity and resource usage factors for the three different instance sets.

Table 3.4: The best combination of parameters

Problem Set	*class_size*	*prob_ss*	*num_elite*
J30	80	10	12
J60	100	5	4
J120	100	1	4

3.5.2. Comparison of proposed TLBO with other approaches

In order to test the effectiveness of the TLBO, the results have been compared with other existing approaches available in literature to solve the RCPSP. For each instance set of J30, J60 and J120, the algorithm was run for maximum 1000 and 5000 number of schedules as the

stopping criterion and other parameters namely *class-size*, *prob_ss* and *num_elite* have been selected from the DOE test as reported in Table 3.4. Table 3.5 presents the computational results of average percent deviations from the optimal makespan for instance set J30. Since the optimal values of J60and J120 are not known for all problems, average percent deviations from critical path based lower bounds is selected for the comparison purpose. The detailed results for J60 and J120 instance set have been presented in Table 3.6 and Table 3.7 respectively.

Table 3.5: Average deviations from optimal makespan for J30 instance set

References	**Algorithm**	**Maximum no. of schedules**	
		1000	**5000**
Mendes et al. (2009)	GAPS	0.06	0.02
Kochetov and Stolyar (2003)	GA, TS, path reli.	0.10	0.04
Agarwal et al. (2011)	Neurogenetic-FBI	0.13	0.02
Chen et al.(2010)	ACOSS	0.14	0.06
Tseng and Chen(2006)	ANGEL	0.22	0.09
Alcaraz et al. (2004)	GA	0.25	0.06
Valls et al.(2008)	Hybrid GA	0.27	0.06
Fang and Wang(2012)	SFLA	0.36	0.21
Hartmann (2002)	Self-adapting GA	0.38	0.22
Nonobe and Ibaraki (2002)	Tabu Search	0.46	0.16
This work	**TLBO**	**0.52**	**0.25**
Hartmann (1998)	Activity list GA	0.54	0.25
Schirmer (2000)	Adaptive sampling	0.65	0.44
Kolisch and Drexl (1996)	Adaptive sampling	0.74	0.52
Kolish(1996)	serial sampling (LFT)	0.83	0.53
Hartmann(1998)	Random Key GA	1.03	0.56
Hartmann(1998)	priority rule GA	1.38	1.12
Kolish(1996)	parallel sampling (LFT)	1.40	1.29
Leon and Balakrishnan (1995)	problem space GA	2.08	1.59

It can be seen from Table 3.5 that for J30 instance set, the developed algorithm is found to be 11^{th} best among all other approaches with average percent deviation as 0.52% for 1000 schedules and 0.25% for 5000 schedules. For the J60 and J120 instance set the average percent deviation from critical path based lower bound is found to be 13.19% and 39.90% respectively when 1000

schedules are computed and 12.72% and 38.73% when 5000 schedules have been computed. (see Table 3.6 and Table 3.7).

However, similar to other approaches, the performance of the algorithm diminishes with the increase in combination explosion in RCPSP for an increased number of activities and network complexity

Table 3.6. Average deviations from critical path lower bound for J60 instance set

References	Algorithm	Maximum no. of schedules	
		1000	5000
Fang and Wang(2012)	SFLA	11.44	10.87
Agarwal et al. (2011)	Neurogenetic-FBI	11.51	11.29
Kochetov and Stolyar (2003)	GA, TS, path reli.	11.71	11.17
Mendes et al. (2009)	GAPS	11.72	11.04
Chen et al.(2010)	ACOSS	11.75	10.98
Alcaraz et al. (2004)	GA	11.89	11.19
Tseng and Chen(2006)	ANGEL	11.94	11.27
Valls et al.(2008)	Hybrid GA	12.21	11.27
Hartmann (2002)	Self-adapting GA	12.21	11.70
Schirmer (2000)	Adaptive sampling	12.94	12.58
Nonobe and Ibaraki (2002)	Tabu Search	12.97	12.18
This work	**TLBO**	**13.19**	**12.72**
Hartmann (1998)	Activity list GA	13.30	12.74
Hartmann(1998)	priority rule GA	13.30	12.74
Kolisch and Drexl (1996)	Adaptive sampling	13.51	13.06
Kolish(1996)	parallel sampling (LFT)	13.59	13.23
Kolish(1996)	serial sampling (LFT)	13.96	13.53
Leon and Balakrishnan (1995)	problem space GA	14.33	13.49
Hartmann(1998)	Random Key GA	14.68	13.32

It can be concluded from the results exhibited in Table 3.5 to Table 3.7 that the proposed TLBO algorithm is competitive to the other 18 approaches and metaheuristics chosen for the comparison. For a limited number of schedules and small problem instances, the algorithm very quickly converges to optimum and near-optimal solutions.

3.6. Summary

In this chapter, a relatively new population based metaheuristic called teaching-learning based optimization algorithm for the RCPSP is presented and tested. For encoding the individuals, a precedence feasible activity list is employed whereas SGS is used as decoding scheme.

Table 3.7. Average deviations from critical path lower bound for J120 instance set

References	**Algorithm**	**Maximum no. of schedules**	
		1000	**5000**
Valls et al.(2008)	Hybrid GA	34.07	32.54
Agarwal et al. (2011)	Neurogenetic-FBI	34.65	34.15
Kochetov and Stolyar (2003)	GA, TS, path reli.	34.74	33.36
Fang and Wang(2012)	SFLA	34.83	33.20
Chen et al.(2010)	ACOSS	35.19	32.48
Mendes et al. (2009)	GAPS	35.87	33.03
Tseng and Chen(2006)	ANGEL	36.39	34.49
Alcaraz et al. (2004)	GA	36.53	33.91
Hartmann (2002)	Self-adapting GA	37.19	35.39
Hartmann (1998)	Activity list GA	39.37	36.74
Kolish(1996)	parallel sampling (LFT)	39.60	38.75
Schirmer (2000)	Adaptive sampling	39.85	38.70
This work	**TLBO**	**39.90**	**38.73**
Hartmann(1998)	priority rule GA	39.93	38.49
Nonobe and Ibaraki (2002)	Tabu Search	40.86	37.88
Kolisch and Drexl (1996)	Adaptive sampling	41.37	40.45
Kolish(1996)	serial sampling (LFT)	42.84	41.84
Leon and Balakrishnan (1995)	problem space GA	42.91	40.69
Hartmann(1998)	Random Key GA	45.82	42.25

To enhance the exploitation and exploration capabilities of the original algorithm, in addition to teacher and learner phase, the concepts of self-study and examination phase, inspired by other studies, have been also employed in this work. An orthogonal-array based Taguchi design was used to determine the best set of parameters for each instance set. The comprehensive test results on problem instance sets taken from literature showed that the TLBO approach is reasonably

effective and competitive to other well-known solution techniques and metaheuristics available to solve the RCPSP.

In the next chapter the TLBO developed here will be extended for the multi-skill resource-constrained project scheduling problem (MSRCPSP) which is another research objective of this thesis. It is obvious that some fundamental modifications in encoding and decoding schemes have to be made to incorporate the flexible nature of resources. The details of this modified TLBO for the MSRCPSP have been exhibited in the coming chapter.

Chapter 4

A Teaching-Learning-Based Optimization Algorithm for the MSRCPSP

Chapter 4

A Teaching Learning Based Optimization Algorithm for the MSRCPSP

4.1 Introduction

In this chapter, one of the recent and practical extensions of the RCPSP termed as the multi-skill resource constrained project scheduling problem (MSRCPSP) is considered for investigation. Unlike the RCPSP considered in last chapter, the resources are *multi-skilled* i.e. they have been assumed to possess more than one skill. To solve this complex problem, the teaching-learning-based optimization (TLBO) algorithm developed in the chapter 3 is extended by incorporating modified encoding and decoding schemes. More specifically, an activity list based encoding scheme has been modified to include the multi-skilled resource assignment information. For comparing the performance of this metaheuristic, a genetic algorithm (GA) is also developed to solve this problem. The computational experiments have been performed on the test instances generated for the purpose with varying characteristics of network complexity and resource strengths. The results obtained by the TLBO are quite promising in terms of average percentage deviation from the critical path based lower bound.

The remaining of this chapter is structured as follows: The problem nature and mathematical model along with an illustrative example is provided in next Section. Section 4.3 discusses the implementation issues and framework of the proposed TLBO. The procedure of instance generation and computational experiences has been presented in Section 4.4. Finally, Section 4.5 presents the key findings and conclusions drawn from the study**.**

4.2 Problem description and mathematical formulation

The problem considered here is Multi-Skill Resource-Constrained Project Scheduling Problem (MSRCPSP) which is a realistic extension of the problem studied by Zheng et al. (2017). Unlike Zheng et al. (2017), this work considers a real life scenario wherein resource requirement for any activity is not restricted to unity and can be more than one. Also, it is assumed that activities may require more than one type of skill for their execution. Thus, the number of persons required corresponding to each skill may be more than one. The resources are staff members mastering

one or more skills. An activity-on-node (AON) acyclic network G= (A, E) represents a single project where A= {1, 2,……., i, j,….N+2} denotes the activity set and E provides the precedence relations between them. The beginning and end activity of the project are dummy activities i.e they consume no resource and time for their execution. A set P of renewable resources comprising of staff members is considered wherein each member, as mentioned earlier, possesses one or more skills. It is also assumed that each resource is available for the entire project horizon but can contribute only one skill at a time to an activity. Finally, the activity times have been considered deterministic and positive integers.

4.2.1 Mathematical formulation for the MSRCPSP

To understand the nature of the problem under study, a mathematical model is presented in this section comprising of important notations, objective function and constraints along with their definitions.

Notations:

Parameters	**Definition**
N	number of non-dummy activities in the project
A , $i \in \{1, \ldots\ldots, N+2\}$	set of activities
p_i	processing time of activity A_i
K	total number of skills available
P	total number of available staff members
S , $k \in \{1, \ldots, K\}$	set of skills
P, $m \in \{1, \ldots\ldots, P\}$	set of staff members
$S_{m,k}$	equal to 1 if staff member P_m possesses skill S_k, 0 otherwise
$b_{i,k}$	number of staff members with skill S_k required by activity i
t_i	start time of activity i
T	project horizon

Decision variables:

$$x_{i,m,t} = \begin{cases} 1; & \text{if staff member } m \text{ starts an activity } i \text{ at time } t, \\ 0 & \text{otherwise} \end{cases}$$

$$y_{i,m,k} = \begin{cases} 1; & \text{if staff member } m \text{ starts an activity } i \text{ with skill } k, \\ 0 & \text{otherwise} \end{cases}$$

$$Z_{i,t} = \begin{cases} 1; & \text{if activity } i \text{ is started at time } t, \\ 0 & \text{otherwise} \end{cases}$$

Mathematical Model

Minimize t_{N+2} (4.1)

Sub. to :

$t_i = \sum_{t\in[0,T]}(Z_{i,t}\,.\,t)$ $\forall\, i \in A$ (4.2)

$t_i + p_i \leq t_j$ $\forall\, (i,j) \in E$ (4.3)

$\sum_{t\in[0,T]} x_{i,m,t} \leq 1$ $\forall\, i \in A,\ \forall m \in P$ (4.4)

$\sum_{i\in A}\sum_{d\in[t-p_i+1]} x_{i,m,d} \leq 1$ $\forall\, t \in [0,T]$ (4.5)

$x_{i,m,t} \leq z_{i,t}$ $\forall\, i \in A, \forall\, m \in P, \forall\, t \in [0,T]$ (4.6)

$x_{i,m,t} + 1 \leq z_{i,t} + \sum_{k\in S} y_{i,m,k}$ $\forall\, i \in A, \forall\, m \in P, \forall\, t \in [0,T]$ (4.7)

$y_{i,m,k} \leq S_{m,k}$ $\forall\, i \in A, \forall\, m \in P, \forall\, k \in S$ (4.8)

$\sum_{m\in P} y_{i,m,k} = b_{i,k}$ $\forall\, i \in A,\ \forall k \in S$ (4.9)

$\sum_{t\in[0,T]} x_{i,m,t} = \sum_{k\in S} y_{i,m,k}$ $\forall\, i \in A,\ \forall m \in P$ (4.10)

$x_{i,m,t} \in \{0,1\}$ $\forall\, i \in A,\ \forall m \in P\,, \forall\, t \in [0,T]$ (4.11)

$y_{i,m,k} \in \{0,1\}$ $\forall\, i \in A,\ \forall m \in P$ (4.12)

$$Z_{i,t} \in \{0,1\} \qquad \forall\, i \in A,\ \forall\, t \in [0,T] \qquad (4.13)$$

The objective function (4.1) minimizes the start time of (dummy) end activity and hence the makespan. Constraint (4.2) represents the start time of activity *i*. Constraint (4.3) ensures the precedence relations between the activities. It is obvious that each staff member can start an activity at most once in whole planning horizon which is handled by constraint (4.4). Constraint (4.5) avoids the possibility of a staff member to work simultaneously on more than one activity. Constraint (4.6) and (4.7) collectively guarantees the synchronization of start times for an activity initiated by different staff members. Constraint (4.8) ensures that a staff member delivers only the skill that he/she masters. The total skill requirement of an activity should be met and this is ensured by constraint (4.9). Constraint (4.10) ensures that a staff member cannot use more than one skill at a time when assigned to an activity. Constraints (4.11) to (4.13) are domain constraints which define the decision variables to be binary.

4.2.2 An illustrative example

This subsection discusses a small hypothetical example to understand the problem under study. Let a project comprises of four non-dummy activities linked by precedence relations as shown in Figure 4.1. The number of resources with particular skill required by each activity is presented in 'Activity-Skill Matrix' as shown in Table 4.1. For example, activity 2 requires one person with skill S_2 and one person with skill S_3 for its execution, activity 4 requires two persons with skill S_1 and one person with S_2 and so on. The activities 1 and 6 being dummy require no resource (and hence skills) for their execution.

The skills attained by four staff members are given by a 'Staff-Skill Matrix' shown in Table 4.2. The value '1' indicates that a staff member masters the particular skill while '0' signifies the absence of skill. To elaborate, staff member 1 possesses two types of skills i.e. S_2 and S_3, staff member 2 possesses skill type S_1 and S_3 and so on.

It is important to note that skill requirements of an activity can be accomplished by any of the staff member(s) (from a pool of total six members) who possesses these skills. Also, all the persons if assigned to execute an activity should be available at the start of the activity simultaneously.

The goal is to minimize the makespan respecting the precedence relations and resource requirements of the activities. A feasible solution determines the start/finish times of each activity along with assignment of subset of staff members that fulfill all the skill needs of the activities. Figure 4.2 presents one such feasible solution which is also optimal in this case.

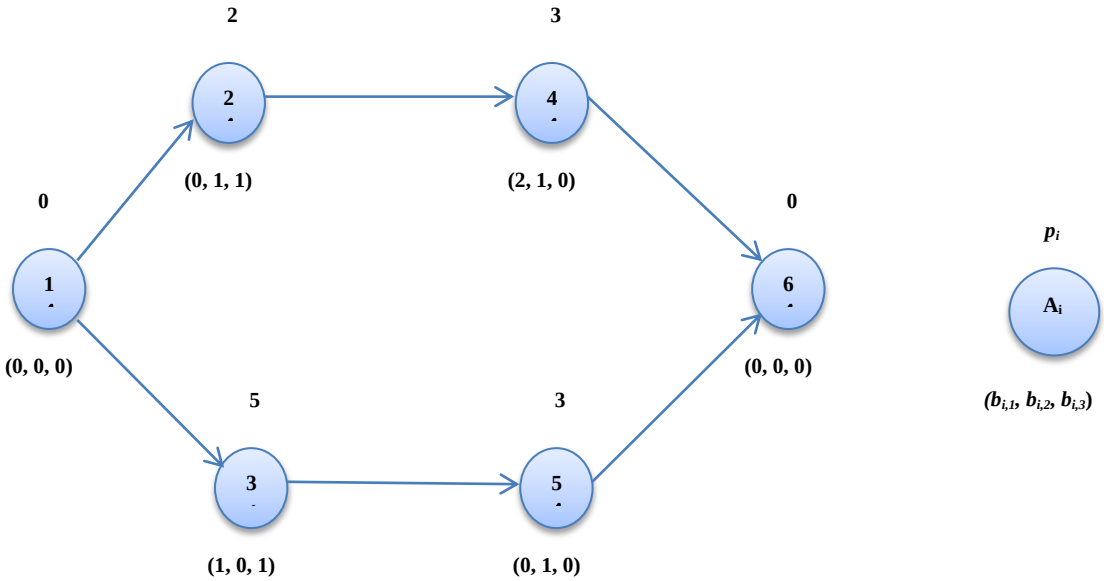


Figure 4.1: Precedence graph of the illustrative project

Table 4.1: Activity-Skill Matrix

Activity	No. of staff members required ($b_{i,k}$)		
	S_1	S_2	S_3
1	0	0	0
2	0	1	1
3	1	0	1
4	2	1	0
5	0	1	0
6	0	0	0

Table 4.2: Staff-Skill Matrix

Staff	Skills Attained		
	S_1	S_2	S_3
1	0	1	1
2	1	0	1
3	1	0	0
4	1	0	1

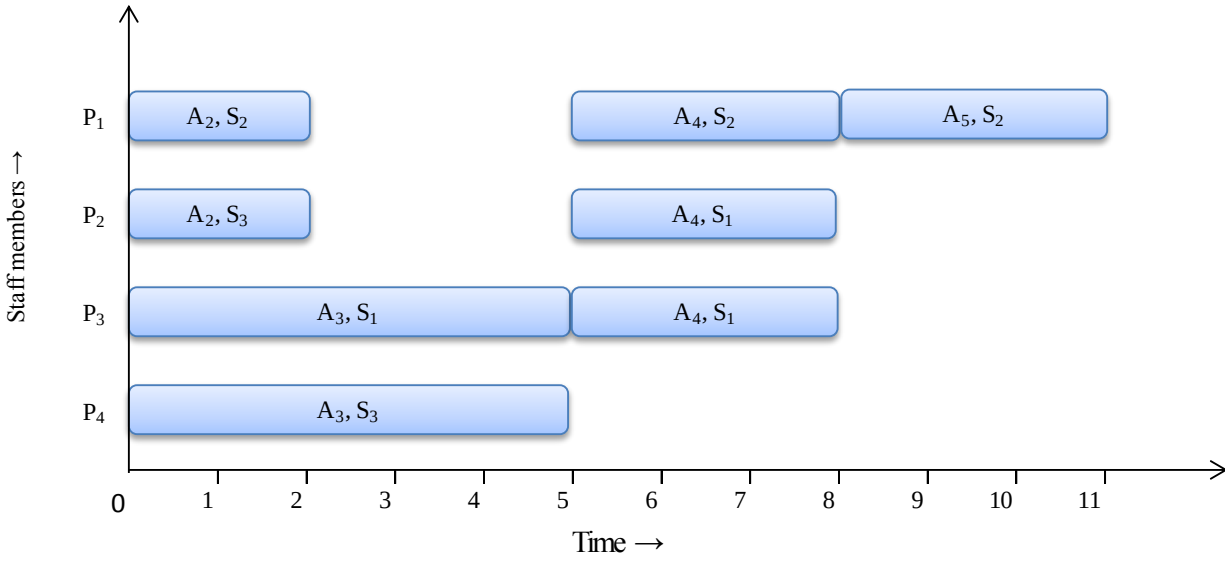


Figure 4.2: A feasible solution of the illustrative example

4.3 Proposed algorithms for the MSRCPSP

The problem under study being an extension of the RCPSP is also NP-hard and thus involves large search spaces. Metaheuristics are known to be natural candidates for tackling such complex problems. However, as explained in previous chapter, before applying a metaheuristic on any problem, some basic issues like encoding and decoding schemes, mechanisms of solution modifications etc. need to be taken care of. The following section presents in detail the

configuration and application methodology of the two algorithms namely TLBO and GA proposed in this work to solve the MSRCPSP.

4.3.1 Teaching-learning-based optimization algorithm for the MSRCPSP

In chapter 3, the philosophy and working mechanism of the TLBO has been explained in detail. The algorithm has been successfully applied for the standard RCPSP and inspired by the competitive results obtained thereof, it is extended for the MSRCPSP. However, due to the multi-skill nature of resources, the encoding and decoding mechanism will differ considerably as compared to the RCPSP. The problem size for the MSRCPSP is exceptionally large and complex and the concepts of self-study and examination are retained to avoid the algorithm been trapped in local optima. There are primarily two modifications that have been incorporated in the previous TLBO algorithm designed for the RCPSP:

1. The encoding scheme is appended with additional information to incorporate the feasible resource assignments of the activities. (please see section 4.3.1.1 for details).
2. The serial schedule generation scheme (SGS) available in literature for the RCPSP is suitably modified to ensure a feasible schedule considering the multi-skill nature of the resources (please see section 4.3.1.2 for details).

The framework or mechanism of proposed TLBO for the MSRCPSP is depicted in Figure 4.3. This is derived by incorporating the two additional features of multi-skilled resources mentioned above. The pseudo code for the solving the above problem is also modified accordingly and same is shown in Figure 4.4.

4.3.1.1 Encoding scheme

As mentioned in chapter 3, this is an important step that significantly affects the performance of any metaheuristic algorithm. Due to multi-skilled resources being involved, one needs to incorporate additional information of the feasible staff assignment corresponding to each activity in the representation. The encoding scheme of the RCPSP is, therefore, modified and a solution is encoded into two parts (Gürbüza, 2010). The first part determines the priority values of the activities while second part defines the feasible staff assignment to each activity.

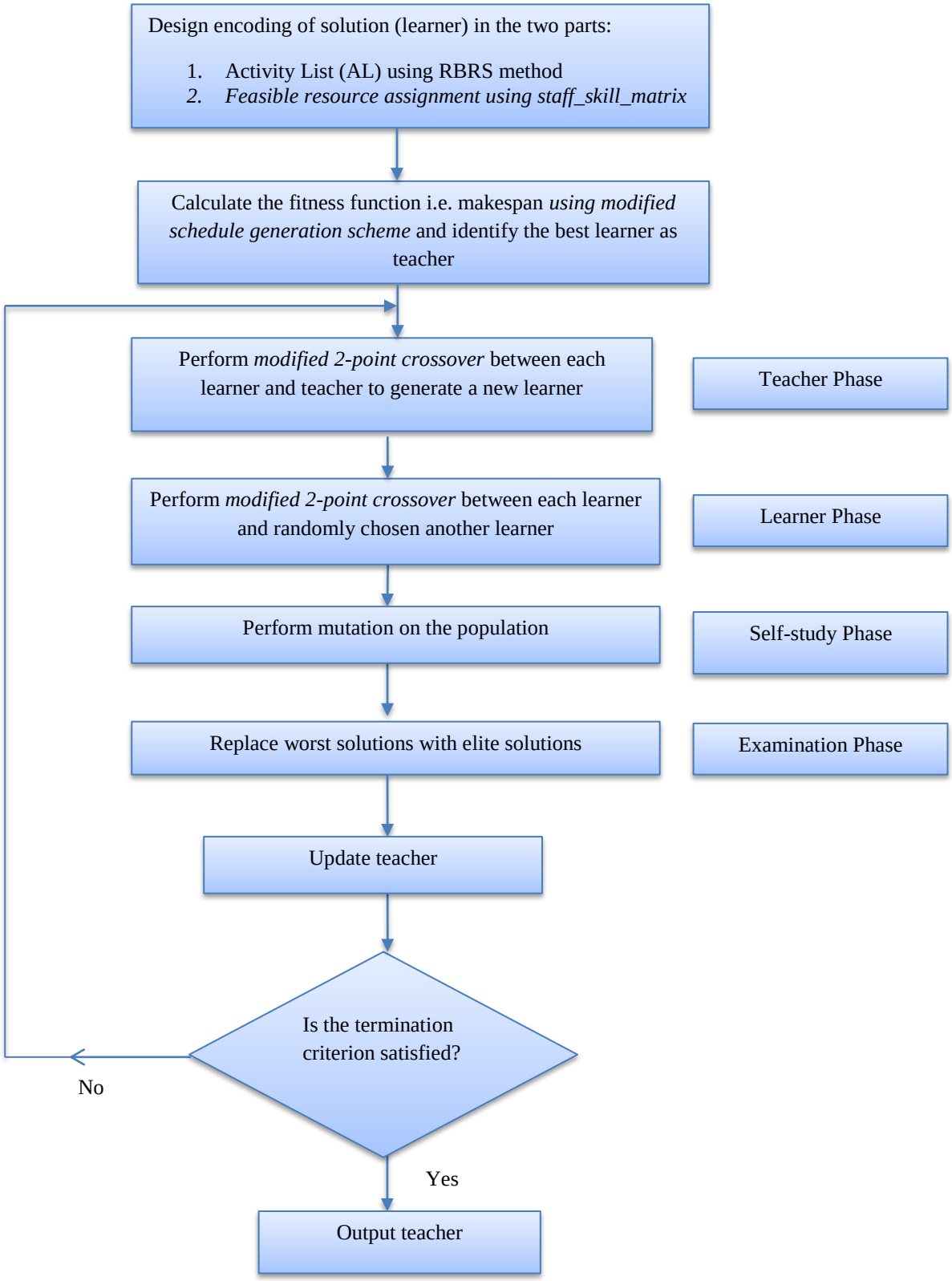


Figure 4.3: Flowchart of the proposed TLBO for the MSRCPSP

```
Input: Initialize Class_size (number of learners in the class) , Num_iter (Number of iterations),
                SS_prob ( probability of self-study) and  Elite_per ( elite size i.e. percentage of students to be failed)
               staff_skill_matrix (information of staffs mastering various skills)

Output: The teacher for iteration size= Num_iter
Begin
    for 1: Class_size
        Generate the upper part (activity list) of encoding scheme using RBRS method
        Assign feasible set of staff members to each activity using staff_skill_matrix to generate a learner
    end  for
    Calculate the fitness i.e. the makespan of each learner using modified SGS method
     Designate the learner with minimum makespan as the teacher
        while (stopping condition is not met);
            for i=1: Class_size
                Perform 2-point crossover between each learner and student                    % Teacher phase
                Evaluate the new learner
                 if  makespan new learner < makespan old learner
                    Replace the old learner with the new learner
                end
                    Perform 2-point crossover between a learner with another random learner    % Student phase
                if  makespan learner-2 < makespan learner-1
                    Replace the learner-1 with learner-2
                end
                Perform mutation with SS_prob                                                 % Self-study phase
                Retain Num_elite learners in the class                                        % Examination Phase
                Calculate makespan and update the teacher
        end while
end
```

Figure 4.4: Pseudo-code for the proposed TLBO for the MSRCPSP

To encode the first part, the precedence feasible activity list (AL) representation from Kolisch and Hartmann (1999) is reused in the encoding procedure. More specifically, a precedence feasible permutation of activities represented as $\lambda = (j_1, j_2, \ldots\ldots, j_{n+1}, j_{n+2})$ is created in which no activity appears before any of its predecessors. The corresponding priorities of activities for their execution in this list are represented by their index values.

To encode the second part of solution or individual, the set of staff members corresponding to each skill is constructed and randomly assigned a feasible set of staff members as per the requirement of each activity. These assignments are depicted in vertical columns corresponding to the activities in AL constructed in first part of encoding scheme (Gürbüza, 2010). A representation of an individual for the illustrative example discussed in section 4.2.2 is shown in Figure 4.5.

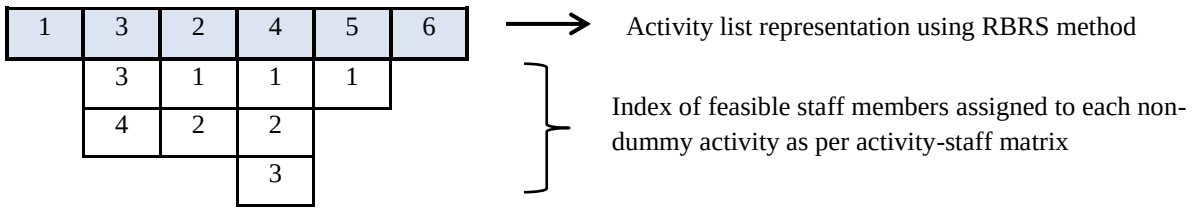


Figure 4.5: Encoding of solution for the MSRCPSP

4.3.1.2 Decoding scheme

To obtain a feasible schedule from above representation, the classical serial schedule generation scheme (SGS) by Kolisch (1996) is modified as used for the RCPSP in chapter 3. The motivation behind this choice is that the search space in SGS consists of active schedules and essentially contains an optimal solution. Unlike RCPSP, when this method is applied to the MSRCPSP verifying the resource constraints is not trivial. Instead, there are a number of feasible ways to assign resources to an activity. The SGS, is therefore, modified in the light of multi-skill nature of the resources. A pseudo code for this modified scheduling scheme is presented in Figure 4.6.

4.3.1.3 Initial population

As evident from section 4.3.1.1 that an individual (a teacher or a learner) comprises of two parts namely a precedence feasible activity list and resource assignment columns. To generate the first part i.e. precedence feasible activity list, a parameterized regret-based biased random sampling (RBRS) method (Kolisch and Drexl, 1996) is employed. It is worthwhile to mention that sampling methods offers advantage in the sense that they are probabilistic in nature which produces different schedules each time when applied and thus ensuring sufficient variability in

the initial population. For details of generation of first part of encoding scheme one is referred to section 3.4.1 of chapter 3.

Let, *A= Set of total non-dummy activities in the project*

Pred (j) = Set of all predecessor activities of activity j
EST (j) = Earliest start time of activity j (as per forward pass in CPM technique)
EFT (j) = Earliest finish time of activity j (as per backward pass in CPM technique)
p_j *=processing time of activity j*
pv_j*,= Priority value of activity j*
S_i *= Start time of activity i*
AL= Activity list as per encoding scheme

1: **Input:** *A, Pred (j), EST(j), EFT(j),* p_j*,* pv_j*, AL*
2: **Output:** *makespan*
3: **begin**
4: $AL \leftarrow A \setminus \{1, N+2\}; t \leftarrow 0; S_1 \leftarrow 0 ; S_j \leftarrow \infty , j \in AL;$
5: **while** $AL \neq \phi$ **do**
6: **find** j^*: $pv_{j^*} = \max \{ pv_{j^*} : j \in AL \wedge Pred(j) \cap AL = \phi \};$
7: **if** $Pred(j^*) = \phi$ **then**
8: $S_{j^*} \leftarrow 0;$
9: **else**
10: Compute SA_{j^*} // set of activities scheduled before j^*
11: **if** $S_k <= EST(j^*) < (S_k + p_k)$ *or* $S_k < EFT(j^*) <= (S_k + p_k)$, $k \in SA_{j^*}$ & there is common staff member in j^* and SA_{j^*}
12: $S_{j^*} \leftarrow \max\{S_k + p_k\}; k \in SA_{j^*}$
13: **else**
14: $S_{j^*} \leftarrow EST(j);$
15: **end if**
16: $AL \leftarrow AL \setminus j^*;$
17: **end if**
18: $makespan \leftarrow max \{S_j + p_j : j \in A\};$
19: **end while**
20: **end begin**

Figure 4.6: Pseudo code for the modified SGS

In order to generate an encoded individual for the MSRCPSP case, there is a need to assign the feasible set of staff members in corresponding vertical columns against each activity. By 'feasible' it is meant that all staff members assigned to an activity must be unique to fulfill the constraint that a staff member can only fulfill one skill requirement of an activity at a time (although he/she may master more than one skill). For this purpose, a set comprising of resources mastering the skills required by an activity is generated and resources are selected from this set using a random device respecting the feasibility constraint. In this way, a total of *Class_size* individuals are generated.

4.3.1.4 Teacher and learner phase

As mentioned earlier, in teacher phase, the teacher who is also the individual with best fitness transfers his/her knowledge into the other individuals (learners) of the population (class). In order to realize this phase, a teacher is subjected to crossover with learner in the class. Similar to the RCPSP, both the 1-point and 2-point crossovers mechanisms as proposed by Hartmann, 1998 have been tested. However, in this case the solution is encoded in two parts comprising vertical columns of resource assignments along with horizontal activity list (AL). To ensure that the new individual also confirms to a feasible resource assignment, the resource assignment columns are transferred along with the activities during the crossover. For the general details of 2-point crossover recall the section 3.4.2 of chapter 3.

It can be seen in Figure 4.7, a learner before crossover is represented as λ_1 whereas λ_2 represents a teacher who has to transfer his knowledge using crossover into the learner. A pair of two distinct random numbers, (u_1, u_2) is generated in the range [1, n], where n denotes the total number of activities in the activity list. It is important to note that to ensure the resource feasibility, the corresponding staffs members assigned to activities are carried over in the new learner. The learner so obtained after teaching is selected if it gives lower makespan; else previous learner is retained.

After teacher phase is over, the learner phase is applied in which each learner is subjected to cross-over with another randomly chosen learner from the population. This is based on the practical analogy that a learner improves his/her knowledge by mutual interactions and

discussions with other learners of class. Again, this new learner is accepted in population if it provides a better makespan else previous learner is retained in the population.

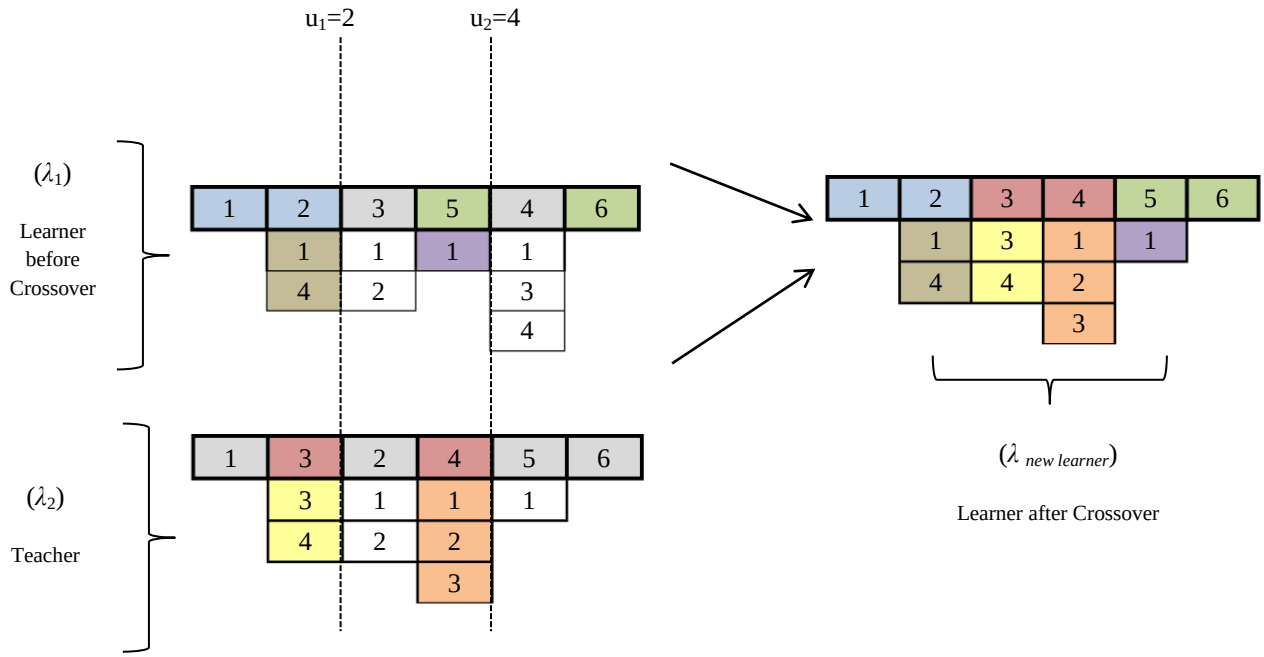


Figure 4.7: An illustration of 2-point crossover mechanism for the MSRCPSP

4.3.1.5 Self-study and examination Phase

Similar to the TLBO developed for the RCPSP, for the multi-skilled problem environment also, the concepts of the self-study and examination phase have been incorporated which are known to enhance the exploration and exploitation capabilities of the TLBO. To keep the discussion concise, the already mentioned details regarding implementation mechanisms of these two features have been omitted here.

4.3.2 Proposed Genetic Algorithm (GA) for the MSRCPSP

As stated earlier, this work also aims to develop another popular metaheuristic namely genetic algorithm (GA) for the MSRCPSP primarily for the comparison purpose. GA is based on natural selection and biological evolution process and has been an indispensable choice for solving

many hard optimization problems since its advent. The basic scheme, operators and implementation framework of the proposed GA on the MSRCPSP are discussed in next section.

4.3.2.1 Initial population and parent selection

Let an even integer *POP* denotes the total number of individuals in a population. Initial population is created by the regret-based biased random sampling method as mentioned in Section 4.3.1.3. While creating individuals, same encoding and decoding procedures used for the TLBO have been adapted for obvious reasons of fair comparison. The individuals in the *POP* have been evaluated for their fitness value (i.e. makespan) using the modified SGS developed earlier. In order to choose parents for crossover, 2-tournament selection method is employed (Hartmann, 1998) wherein two individuals I_1 and I_2 are randomly chosen from *POP* and if $f(I_1) < f(I_1)$, individual I_1 is chosen else this process is repeated till *POP* individuals are selected. A pseudo code for the proposed GA is presented in Figure 4.8.

```
// Initialize generation 0
g:=0;
//Create an initial population having POP individuals using RBRS method
// Evaluate POP;
Compute fitness (i) for each i ∈POP;
while g< GEN
    do
    {   //Create generation g+1;
        Select pairs of individuals (parents) using 2-tournament method;
        Perform crossover and mutation to produce CHI;
        Evaluate the fitness (i) for each i ∈CHI;
        Add the CHI to POP to get 2*POP individuals;
        Select POP individuals using ranking method
        Keep elite solutions for the next generation
        g:=g+1;
    }
    return the fittest individual as solution
```

Figure 4.8: Pseudo code for the proposed GA

4.3.2.2 Details of genetic operators

The two basic genetic operators conventionally used in GA are crossover and mutation. The 2-point crossover mechanism as applied in the TLBO is re-employed for similarity purpose in the two algorithms; however, in this case it is implemented between two members of a parent group producing two offspring. More specifically, for a given parent group having a mother $M = \{J_1^M, \dots \dots . J_N^M\}$ and a father $F = \{J_1^F, \dots \dots . J_N^F\}$, two offspring, a daughter $D = \{J_1^D, \dots \dots . J_N^D\}$ and a son $S = \{J_1^S, \dots \dots . J_N^S\}$ are produced. A pseudo code for producing a daughter and son using 2-point crossover is exhibited in Figure 4.9.

Following the crossover, mutation as proposed by Boctor, 1996 is implemented to introduce the genetic diversity in the population. Similar to the TLBO, elitism was also introduced to improve the algorithm performance. Section 4.4.2.2 explains the method to obtain the best combination of crossover, mutation and other parameter values for the proposed GA.

Select two random integers u_1 and u_2 such that, $1 \leq u_1 < u_2 \leq N$
/ u_1 and u_2 are the random crossover-points */*

/ Generation of the daughter */*
for k=1 to u_1 do
    $J_k^D = J_k^M$;
for k=u_1+1 to u_2 do
    i=lowest index/ $1 \leq i \leq N$ *and* $J_i^F \notin \{J_1^D, \dots \dots, J_{k-1}^D\}$;
    $J_k^D = J_i^F$;
for k=u_2+1 to N do
    i=lowest index/ $u_1 < i \leq N$ *and* $J_i^M \notin \{J_1^D, \dots \dots, J_{k-1}^D\}$;
    $J_k^D = J_i^M$;
end

/ Generation of the son */*
for k=1 to u_1 do
    $J_k^S = J_k^F$;
for k=u_1+1 to u_2 do
    i=lowest index/ $1 \leq i \leq N$ *and* $J_i^M \notin \{J_1^S, \dots \dots, J_{k-1}^S\}$;
    $J_k^S = J_i^M$;
for k=u_2+1 to N do
    i=lowest index/ $u_1 < i \leq N$ *and* $J_i^F \notin \{J_1^S, \dots \dots, J_{k-1}^S\}$;
    $J_k^S = J_i^F$;
end

Figure 4.9: Pseudo-code for 2-point crossover in the proposed GA

4.4. Computational experiences

In this section the computational experiments to assess the behaviour of the proposed TLBO and GA on test instances are presented. Both algorithms have been coded in MATLAB 7 environment and executed on a laptop computer with Core i3, and Windows 8.1 using 4 GB of RAM. In the next sections the methodology of generating the test instances, parameter setting and comparative results have been discussed in detail.

4.4.1 Test instances for the MSRCPSP

Unlike the RCPSP (PSPLIB, http://www.bwl.uni-kiel.de/Prod/psplib/), there does not exist any standard benchmark instances for the MSRCPSP. Although in the work of Myszkowski et al. 2018, 36 benchmark instances (available as iMOPSE dataset) have been developed for the bi-objective MSRCPSP but they cannot be used here. This is because in this work it is assumed that resource requirements of activities are more than one. In addition, no cost aspects have been considered in this work.

In the work of Almeida et al. (2015), a methodology is available for generating the MSRCPSP instances (as used in this work) with variable characteristics. This methodology will be used to develop different MSRCPSP instances for testing the two algorithms. As mentioned thereof, the three major characteristics that mainly affect the complexity of a MSRCPSP instance are:

i. *Network Complexity* (*NC*)*:* It is a measure of average number of non-redundant arcs or average number of successors of each activity in the precedence graph. For the project instance shown in Figure 4.1, total number of (non-redundant) arcs is 6 and there are 6 activities in the project. As per definition, the network complexity (*NC*) is calculated as *NC*= 6/6=1.0. It is obvious that as the number of arcs in a network increases for a given number of activities, *NC* also increases.

ii. *Skill Factor* (*SF*)*:* It is simply the ratio of types of skills required by a particular activity to the total skills number of types available in a given project. For example, w.r.t. Table 4.1, 3 type of skills are available in project namely S_1, S_2 and S_3. It is evident that activity 2 requires only two types of skills (S_2 and S_3) for its execution which gives its *SF* as

2/3=0.67. On similar lines, SF of activity 3, 4 and 5 can be calculated as 0.67, 0.67 and 0.33 respectively.

iii. *Modified Resource Strength* (*MRS*)*:* It is a ratio between available resources (staff members) to the total number of resource units required to execute all the activities.

Let,
- N = Number of non-dummy activities;
- P = Total number of available staff members;
- |S| = Number of skill types;
- n = Average number of staff members required for an activity (assuming 2 in this case)

Then, as per definition, *MRS* can be calculated as: $MRS = \frac{P}{N*|S|*n}$. Alternatively, if one pre-defines the *MRS*, the number of staff members required for a particular configuration can then be determined as $P = N * |S| * n$.

The methodology proposed by Almeida et al. (2015) is coded in MATLAB 7 and 216 different instances for the MSRCPSP have been generated using different combinations of the above three characteristics. The details of the developed instances with salient features have been as mentioned below:

- Total number of non-dummy activities in the instances, $N = 30$.
- Processing time of activities, p is randomly derived as $D \sim U(1, 10)$.
- A total of four different skill types have been considered i.e. $|S|=4$.
- Network Complexity (*NC*) is varied for three possible values similar to PSPLIB instances (Kolisch and Sprecher, 1997), i.e. $NC \in \{1.5, 1.8, 2.1\}$.
- Skill Factor is varied to have four possible values i.e. $SF \in \{0.5, 0.75, 1, \text{variable}\}$. For instance, $SF=0.75$ means that each activity requires three out of total four available skills for its execution. By "variable" it is meant that for each activity number of skills required is varied randomly in the set {2, 3, 4}.

- Modified Resource factor (*MRS*) is varied from 0.0667 to 0.0944. This is because too low MRS value tends to generate instances extremely easy to solve while higher value may induce the resource infeasibility.
- Each activity is assumed to require {1, 2, 3} staff members corresponding to each skill.
- Each staff member is assumed to master 1, 2 or 3 skills out of four skills.
- The total number of staff members assigned to each instance, *P* can be determined by fixing its *SF* and *MRS* values as depicted in Table 4.3.

Table 4.3: Number of staff members for given values of *SF* and *MRS*

SF=0.5		*SF*=0.75		*SF*=1.0		*SF*=var.	
MRS	*P*	*MRS*	*P*	*MRS*	*P*	*MRS*	*P*
0.0667	8	0.0667	12	0.0667	16	0.0667	12
0.0750	9	0.0778	14	0.0750	18	0.0778	14
0.0917	11	0.0944	17	0.0917	22	0.0944	17

Table 4.4 presents the summary of the characteristics of the test instances generated at a glance. For each combination of *SF*, *NC* and *MRS*, 6 instances have been generated thus a total of (4*3*3) *6=216 instances have been generated for testing the behaviour of the developed algorithms.

Table 4.4: Summary of characteristics of the test instances

Factor	Value
Number of activities (*N*)	N=30
Activity duration (p_i)	$p_i \sim U(1,10)$
Network Complexity (*NC*)	$NC \in \{1.5, 1.8, 2.1\}$
Skill Factor (*SF*)	$SF \in \{0.5, 0.75, 1, \text{variable}\}$
Modified Resource Strength (*MRS*)	$0.0667 \leq MRS \leq 0.0944$
Total no. of available skills ($\|S\|$)	$\|S\|=4$
Number of staff members required by each skill, $b_{i,k}$	$b_{i,k} \in \{1,2,3\}$

4.4.2 Parameter setting

Parameter setting is a crucial step in any metaheuristic to determine the optimum combination of factors that influence its performance. It is first performed for the TLBO and subsequently done for the GA. It is important to note that conventional TLBO has only two parameters to tune i.e. number of iterations and population size. However, in the modified TLBO as proposed in this work, three parameters need to be tuned, the details as mentioned below.

4.4.2.1 Parameter setting for the TLBO

In the proposed TLBO three factors at three different levels have been selected for fine tuning as depicted in Table 4.5. These levels have been selected from the literature (Rao et al., 2012) and inspired by their competitive performance when applied to different benchmark functions. In another work of Rao and Patel (2012) where elitist TLBO was proposed, population sizes of 25, 50, 75 and 100 were used with elite size as 0, 4, 8 and 12. Inspired by these values, three population sizes have been selected which are reasonably close to the values chosen in the literature i.e. 20, 40 and 60. These values are equidistant for ensuring fair comparison during parameter tuning. It is interesting to note that elite size proposed in the TLBO is employed as a percentage of total population hence all three population sizes have been chosen as even integers to avoid the fractional values in the elite size. Probability of self-study is basically a mutation concept and its levels are inspired from Alcaraz and Moroto (2001).

Table 4.5: Factors and corresponding levels for the TLBO

Factor	Symbol	Level	Values		
Number of learners (pop size)	*Class_size*	3	20	40	60
Probability of self-study	*SS_prob*	3	0.01	0.05	0.10
Percentage of elite learners	*Elite_per*	3	0.05	0.10	0.15

Two crossover mechanisms from literature namely 1-point and 2-point have been utilized to realize the teacher and learner phase. Moreover, the self-study phase is also incorporated using two mutation mechanisms one proposed by Boctor (1996) and other by Hartmann (1998) and

referred in this work as BM and HM respectively. To determine the best combination of these two factors (each having two levels), all these are crossed together, thus having four different combinations to be tested. The *Class_size* is fixed at 20, *SS_prob* at 0.01 and *Elite_per* at 0.05 during this test and 50 instances have been selected randomly from the 216 instances developed for the MSRCPSP. As optimal solutions of the instances are not known, deviation from critical path based lower bound has been taken into consideration which is given as, $DEV = (Z^H - Z^{CP})/Z^{CP}$ where Z^H is the solution provided by the developed algorithm and Z^{CP} is the critical path duration. The test results for crossover and mutation combinations are shown in Table 4.6.

It can be seen that Boctor mutation and 2-point crossover have performed relatively better than other combinations. These have been therefore selected for further tuning of factors. A full factorial method needs 3^3=27 different tests to be performed. Again, in order to decrease the number of tests, Taguchi's design of experiments (DOE) approach is employed. The proposed work employs traditional method of handling the responses of multiple trials by using average or mean (Fang and Wang, 2012). However, it is surmised that S/N ratio should be used if consistency is desired over mean especially in situations where responses are more susceptible to be affected by variation within the data. A L_9 (3^3) orthogonal array is used for the test having eight degrees of freedom (DOF) as mentioned in Table 4.7. A total of 1000 schedules have been generated as stopping criterion and average deviation from critical path based lower bound for the 50 randomly selected instances is calculated as ARV (average response variable).

Table 4.6: Test results for different combinations of crossover and mutation

Crossover	Mutation	AVG % DEV.
1-point	BM	0.7125
1-point	HM	0.7236
2-point	BM	0.6895
2-point	HM	0.7105

Table 4.7: Orthogonal table and the ARV values for DOE test for TLBO

Exp. number	Factors			ARV
	Class_size	*SS_prob*	*Elite_per*	
1	20	0.01	0.05	0.6922
2	20	0.05	0.10	0.6316
3	20	0.10	0.15	0.5986
4	40	0.01	0.10	0.5813
5	40	0.05	0.15	0.6227
6	40	0.10	0.05	0.5930
7	60	0.01	0.15	0.6309
8	60	0.05	0.05	0.6096
9	60	0.10	0.10	0.5967

The results of Taguchi test can be seen from Table 4.7. The main effects plots for each factor are shown in Figure 4.10. It is evident that optimum levels obtained for the parameters are: *Class_size* = 40, *SS_prob* = 0.10 and *Elite_per* = 0.10.

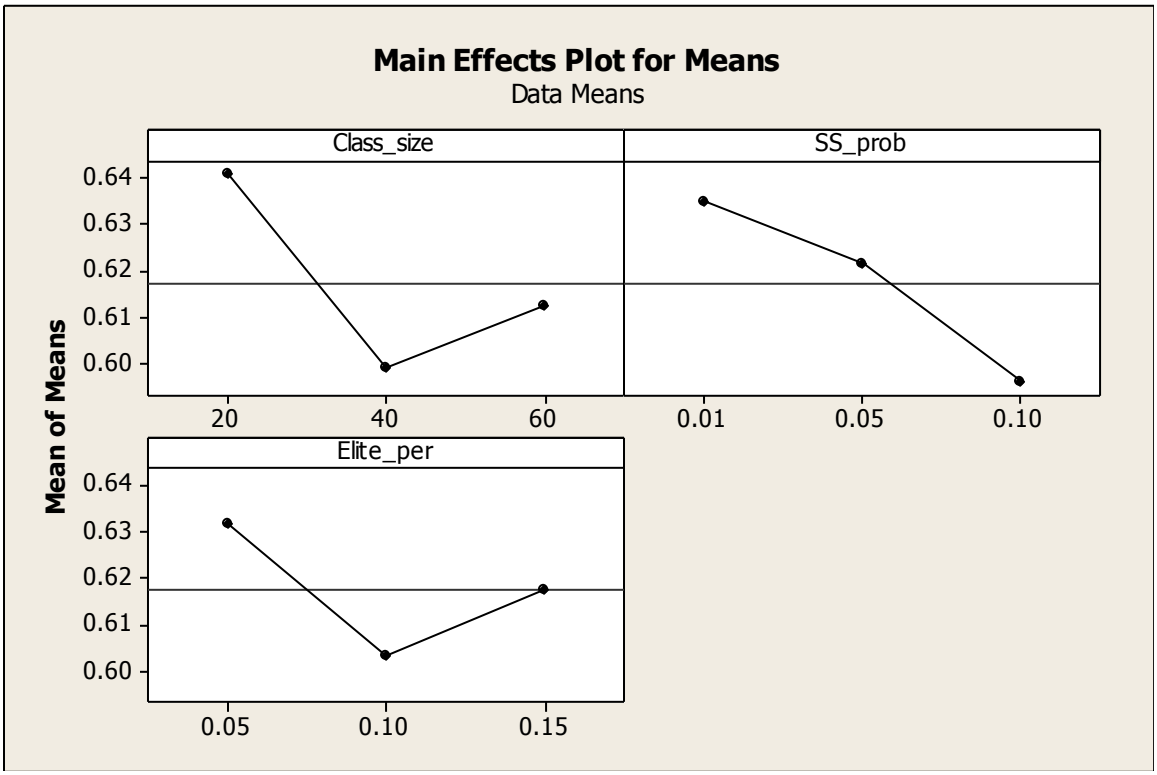


Figure 4.10: Main effects plot for each level of factors of the TLBO

4.4.2.2 Parameter setting for the GA

For the GA discussed in Section 3.2., there are four different parameters of interest namely population size (*Pop_size*), crossover probability (*Cross_prob*), mutation probability (*Mut_prob*) and elite size (expressed as percentage of population size and denoted by *Elite_per*). Three levels have been chosen for each of these parameters as shown in Table 4.8 which are inspired by typical values used in literature for the RCPSP (Alcaraz and Maroto, 2001). The corresponding values of ARV as obtained for L_9 (3^4) orthogonal array have been exhibited in Table 4.9. The optimum levels as obtained after the test are *Pop_size* =40, *Cross_prob* = 0.80 , *Mut_prob* = 0.10 , *Elite_per* = 0.10 (Figure 4.10).

4.4.3 Comparative results

On the basis of optimum values of parameters obtained by DOE tests, the behaviour of both the algorithms i.e. TLBO and GA is tested for 216 instances (36*6) induced by different values of *NC*, *SF* and *MRS*. A total of 5000 schedules were generated for both the algorithms as stopping criterion. As the optimum solutions of these problems are not known, the percentage deviation from critical path based lower bound is used for comparison purpose which is as given below:

$$\% \text{ DEV} = (Z^H - Z^{CP})/Z^{CP} * 100$$

where Z^H is the heuristic solution provided by the algorithm and Z^{CP} is the critical path duration

Table 4.8: Factors and corresponding levels for the GA

Factor	Symbol	Level	Values		
Population size	*Pop_size*	3	20	40	60
Probability of crossover	*Cross_prob*	3	0.70	0.80	0.90
Probability of mutation	*Mut_prob*	3	0.01	0.05	0.10
Percentage of elite individuals	*Elite_per*	3	0.05	0.10	0.15

Table 4.9: Orthogonal table and the ARV values for DOE test for GA

Exp. number	Factors				ARV
	Class_size	*Cross_prob*	*Mut_prob*	*Elite_per*	
1	20	0.7	0.01	0.05	0.8430
2	20	0.8	0.05	0.1	0.7615
3	20	0.9	0.1	0.15	0.7845
4	40	0.7	0.05	0.15	0.7650
5	40	0.8	0.1	0.05	0.7612
6	40	0.9	0.01	0.1	0.7751
7	60	0.7	0.1	0.1	0.7590
8	60	0.8	0.01	0.15	0.8104
9	60	0.9	0.05	0.05	0.7846

The detailed results have been shown in Table 4.10. Each row represents the combination of parameters of the test instance and average percentage deviation obtained by running both the algorithms for the 6 instances for this combination.

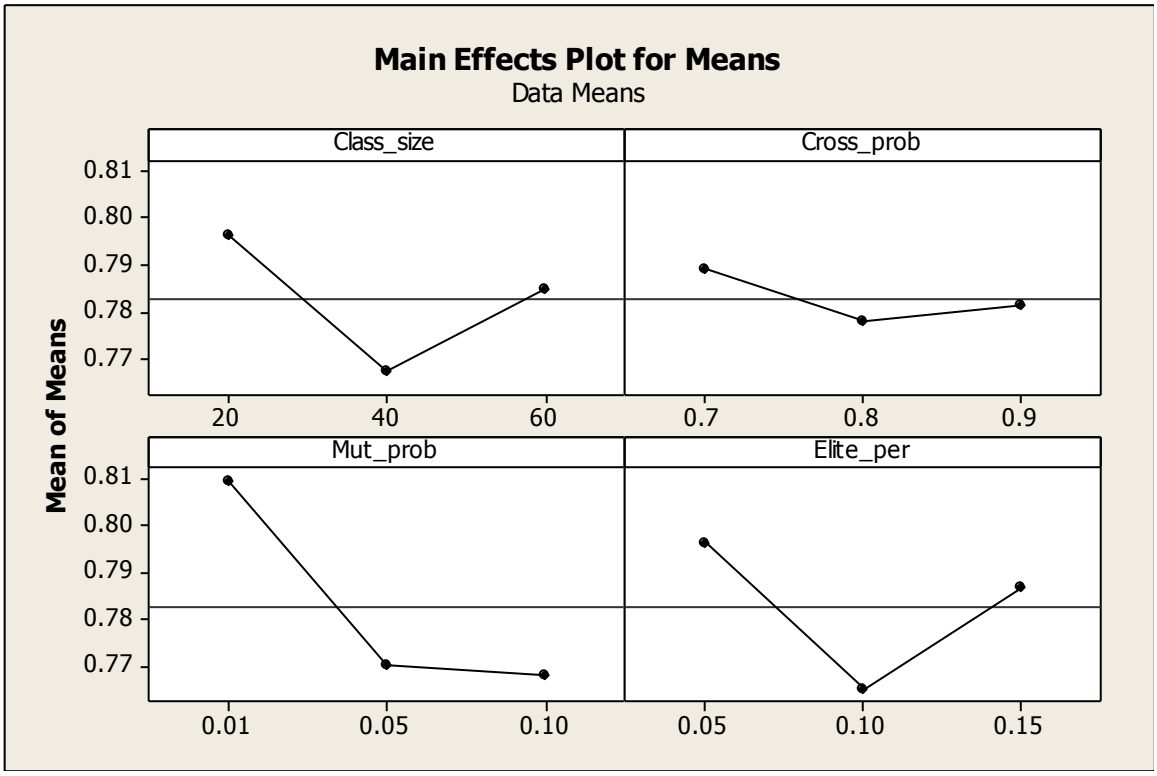


Figure 4.11: Main effects plot for each level of factors of the GA

Table 4.10: Comparison of TLBO and GA for the test instances

SF	*NC*	*MRS*	*P*	TLBO	GA
				(% DEV)	(% DEV)
0.5	1.5	0.0667	8	62.79	83.72
		0.0750	9	58.14	79.07
		0.0917	11	51.16	72.09
	1.8	0.0667	8	57.58	77.27
		0.0750	9	54.55	60.61
		0.0917	11	39.39	50.00
	2.1	0.0667	8	56.72	64.18
		0.0750	9	46.27	61.19
		0.0917	11	34.33	46.27
0.75	1.5	0.0667	12	74.42	86.05
		0.0778	14	61.63	81.40
		0.0944	17	58.14	74.42
	1.8	0.0667	12	66.67	80.30
		0.0778	14	59.09	63.64
		0.0944	17	57.58	56.06
	2.1	0.0667	12	58.21	67.16
		0.0778	14	49.25	53.73
		0.0944	17	37.31	52.24
1	1.5	0.0667	16	83.72	109.30
		0.0750	18	69.77	93.02
		0.0917	22	67.44	86.05
	1.8	0.0667	16	74.24	92.42
		0.0750	18	71.21	72.73
		0.0917	22	66.67	60.61
	2.1	0.0667	16	61.19	73.13
		0.0750	18	52.24	64.18
		0.0917	22	49.25	58.21
var.	1.5	0.0667	12	76.74	90.70
		0.0778	14	65.12	81.40
		0.0944	17	51.16	76.74
	1.8	0.0667	12	68.18	78.79
		0.0778	14	65.15	71.21
		0.0944	17	60.61	62.12
	2.1	0.0667	12	56.72	68.66
		0.0778	14	50.75	59.70
		0.0944	17	38.81	49.25
Avg.				**58.67**	**71.05**

Table 4.11: Summary of results

Parameters	Values	Algorithms	
		TLBO (% DEV)	**GA (% DEV)**
SF	0.5	51.21	66.04
	0.75	58.03	68.17
	1.0	66.19	78.85
	var.	59.25	70.95
NC	1.5	65.02	84.50
	1.8	61.75	68.82
	2.1	49.26	59.82
MRS	0.0667	66.43	80.97
	0.0750	58.70	71.18
	0.0778	58.49	68.51
	0.0917	51.37	62.21
	0.0944	50.60	61.81
	Avg.	**58.67**	**71.05**

As evident from the results, the average percentage deviation obtained by the proposed TLBO is 58.67% while it is 71.05% for the GA thus showing 12.38% reduction on average makespan. A relative comparison shows that TLBO results are 21.10 % better as compared to the GA.

The effects of individual parameters i.e. skill factor (*SF*), network complexity (*NC*) and modified resource strength (*MRS*) have also been investigated on the results obtained. A summary of the behaviour of these parameters on quality of results is presented in Table 11. Out of the total 216 instances, 72 instances correspond to each *NC* while there are 54 instances related to each *SF*. It can be seen that there occur five different values of *MRS* in the overall instances. A *MRS* of 0.0667 is there in 72 instances while the other four *MRS* values 0.0750, 0.0778, 0.0917 and 0.0944 occur in 36 instances each.

From Table 4.11, it can be seen that average deviation increases with increase in skill factor (*SF*) which can be attributed to the fact that serial scheduling scheme is applied for resource assignments and due to the increased proportion of resource requirements by activities, number of possible resource combinations increases accordingly. The behaviour of the results with respect to the instance characteristics is pictorially represented through Figures 4.12 to Figure 4.14.

It can be observed that with the increase in network complexity (*NC*), average deviation decreases similar to the inferences drawn by Almeida et al. (2016). In other words, instances with increased value of *NC* are easier to solve because as the number of precedence relations increases, there becomes less number of activities available that can be processed simultaneously, i.e. the degree of parallelization decreases.

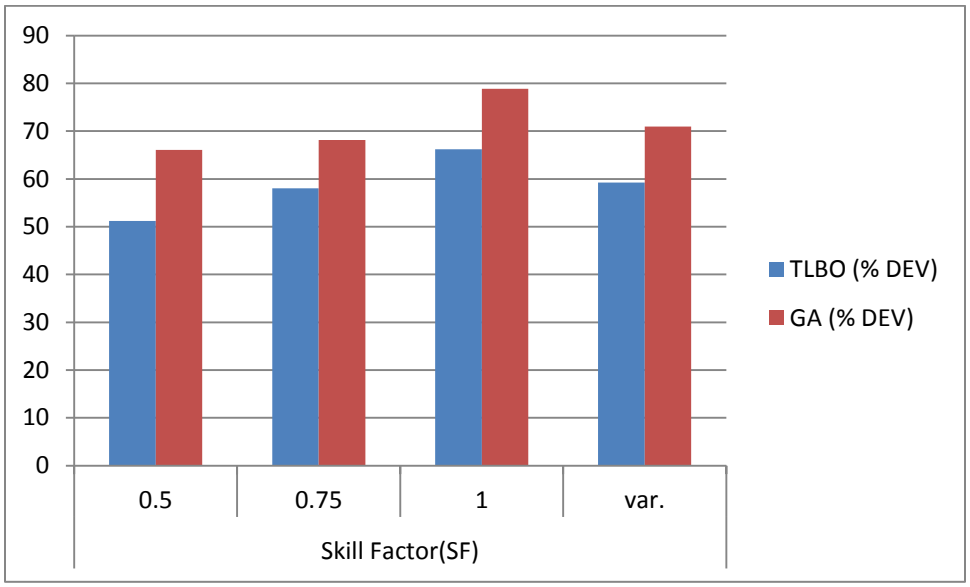


Figure 4.12: Comparison of the TLBO and GA results for different skill factors

Similarly with an increased value of modified resource strength (*MRS*), one can observe relative improvement in results. This is obvious because with the availability of increased number of staff members mastering a particular skill, the resource needs of an activity are satisfied relatively earlier than with reduced value of *MRS*.

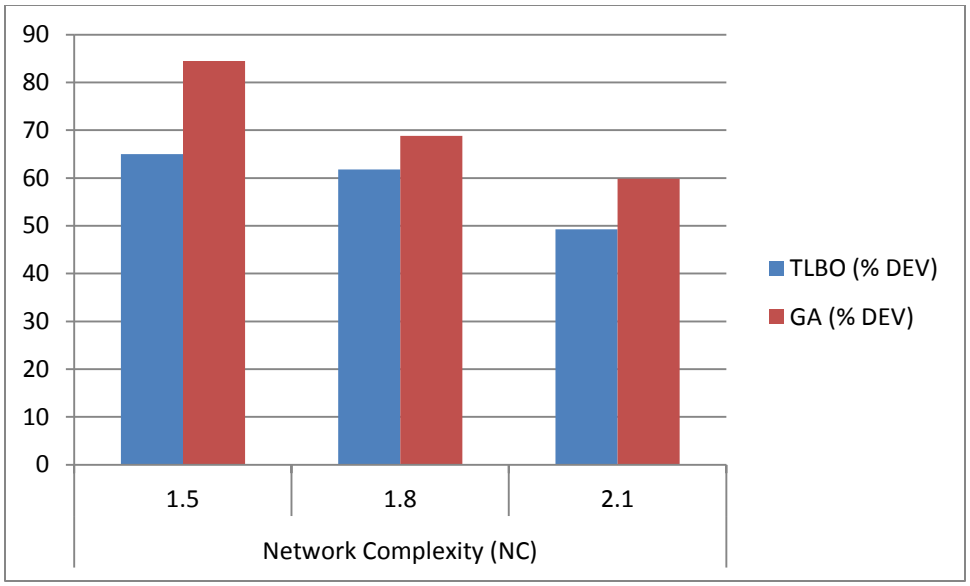


Figure 4.13: Comparison of the TLBO and GA results for different network complexity

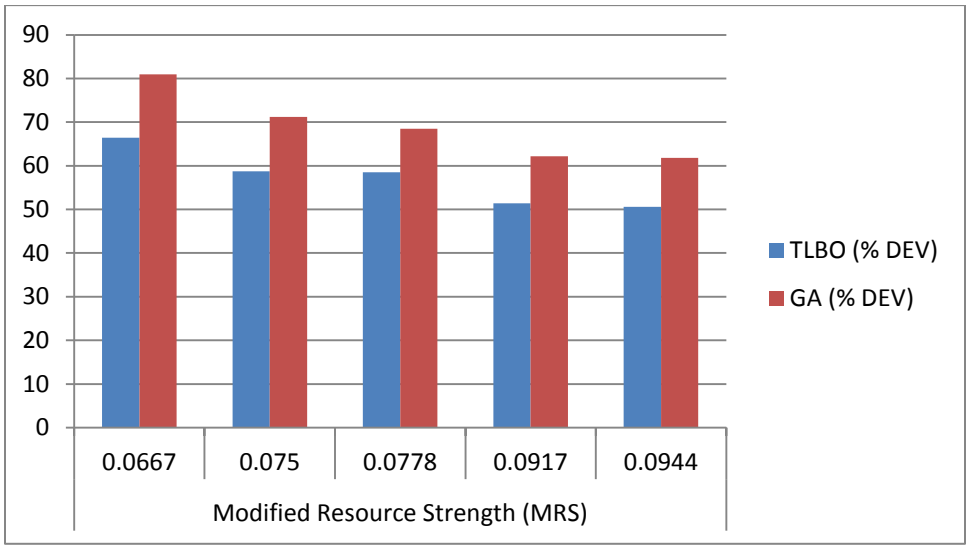


Figure 4.14: Comparison of the TLBO and GA results for different modified resource strength

It can be seen from Table 4.11 that percentage deviations (% DEV) obtained by TLBO for different combinations of *SF*, *NC* and *MRS* are comparatively better than those of GA. Thus proposed TLBO can be regarded as a competitive metaheuristic to solve the MSRCPSP.

4.5 Summary

This chapter investigates the problem of multi-skill resource-constrained project scheduling problem (MSRCPSP) in which resources possess more than a single skill. It is assumed that each activity requires one or more resource persons with varying skill for its execution. A mixed-integer linear programming model has been formulated for this NP-hard problem. Two metaheuristics have been developed to solve this problem namely TLBO and GA. In addition to the conventional teacher and learner phase in TLBO, the concept of self-study and examination phase have also been employed which are known to enhance the exploration and exploitation capabilities of the algorithm according to other studies. Taguchi's orthogonal array method is employed for parameter tuning of both these algorithms. To test the behaviour of proposed algorithms, 36 different combinations of three parameters viz. skill factor, network complexity and modified resource strength have been designed and 6 instances are generated for each combination i.e. 216 (36*6) instances in all have been tested. The results obtained after computational study show that the TLBO has performed significantly better than GA in terms of average percentage deviation from critical path based lower bound for each combination of these parameters. In addition, TLBO also offers an additional advantage of less parameter to tune as compared to GA.

In the next chapter, the metaheuristic developed here will be extended for the multi-objective version of the MSRCPSP which is again a scarcely treated work in literature. The objective of makespan minimization will be handled along with minimization of time elapsed with less-skilled resource assignments. Many product and service organizations involving a large pool of multi-skilled resources are facing problems of uneven allocation of skills to tasks rendering low productivity indices, dissatisfaction among workers and reduced employee morale. Under this motivation, effective metaheuristic algorithms for a multi-objective MSRCPSP will be developed in the next chapter.

Chapter 5

A Multi Objective-TLBO for the Multi Objective -MSRCPSP

Chapter 5

A Multi-Objective TLBO for the Multi-Objective MSRCPSP

5.1 Introduction

In this chapter, the TLBO algorithm developed for the single-objective MSRCPSP is extended for its multi-objective version. One can recall from the comprehensive literature review in chapter 2 that although resource limitation aspect has been sufficiently addressed by researchers in project scheduling, the notion of skilled resources still presents a potential area of research. Moreover, in most of the current studies, only one scheduling objective (mostly the makespan) is considered but in many practical situations decision-makers are generally concerned about several objectives simultaneously which give rise to what is known as multi-objective multi-skill resource-constrained project scheduling problem (MO-MSRCPSP).

There are only few researchers who have focused on other objectives in addition to the regular objective of minimizing the project's makespan for a MSRCPSP. These objectives are largely related to minimizing the cost of allocating the workers in a project or maximizing the overall quality of a given project (Maghsoudlou et al. 2016, Wang et al., 2018). Furthermore, in most of these multi-objective MSRCPSP research works, it has been assumed that a staff member is able to exhibit different skills with the same proficiency or expertise. However, this is not true in real life. Usually in organizations, a person possessing various skills may be expert in one (or two) skill(s) but may only be moderately trained for performing other skills. It is seen that a multi-skilled person in an organization by experience is likely to achieve expertise in the skill domains where he/she is associated for a prolonged period of time. In addition, by his very inherent training he/she is also competent to execute activities requiring other skills. These (other) skills may not be significantly different from his basic skill type but share some common features. For example, a coder in a software development companies may be highly expert to code in JAVA platform but may have moderate level of proficiency in Python, Elixir, TypeScript or other programming languages. The proficiency or expertise of a skill signifies the degree of sophistication, ease or superiority by which a staff member can execute a particular skill.

In this environment, it is sometimes judicious to assign a slightly less proficient person to an activity if an 'expert' or high-skill person is busy in executing other activity. This assignment may help the project manager to meet the project deadlines with only a minor loss in project quality which is often acceptable to the stakeholders. Moreover, it is possible that resources having different levels of proficiencies of same skill can be simultaneously assigned to a particular activity if resource need for the activity is more than one. This may provide an opportunity to an individual to work in team with an expert or high-skilled person thus increasing his/her proficiency level. Nevertheless, the assignments of persons with less-skilled levels should always be kept as low as possible to achieve satisfactory quality targets.

Under this practical motivation, this chapter specifically aims to develop efficient solution techniques for the multi-objective multi-skill resource-constrained project scheduling problem (MO-MSRCPSP) taking variable skill proficiencies into account. In addition to the regular objective of minimizing the makespan, second objective aims at minimizing the total time elapsed in resource assignments with less-skilled persons.

5.2 Multi-objective MSRCPSP

5.2.1 Problem description

Unlike some researchers who focused on discrete type of hierarchical skill levels (for example 1, 2 3 etc.), proficiency levels considered in this work have been realized on a continuous scale similar to the performance ratings of workers under a time-study. More precisely, following levels of proficiencies have been considered for staff members' skills with corresponding values shown in Table 5.1:

Table 5.1: Levels of proficiencies

Level of proficiency	Value
Expert	1.0
Highly-skilled	0.9
Moderately-skilled	0.8
Less-skilled	0.7

In the above table it is important to note that no person can have skill proficiency above 100%, thus proficiency level of an 'expert' staff member is 1. The persons below the value 0.7 are excluded from this consideration as they are highly under-skilled which may be detrimental to the overall project quality. One may argue that allocating less-skilled person to an activity (as compared to an expert) may also deteriorate the quality of a project. The statement is true in strict theoretical sense. However, during a period when an expert person is already busy in executing any pre-allocated activity, it is realistic and practical to allocate a slightly less-skilled (available) person to a current schedulable activity so that a project is completed on time. In addition, with this so called 'less proficient' assignments, the project can still have reasonable or satisfactory level of quality and performance often acceptable to the stakeholders. In simple terms, it can be a natural choice for a project manager to complete a project as early as possible with acceptable quality levels under a given pool of slightly less-proficient resources.

5.2.2 Mathematical model

To accommodate the above philosophy, a bi-objective mathematical model is formulated for the MSRCPSP. The model is similar to the one proposed for the single-objective MSRCPSP with a slight modification in the objective function. For the purpose of completeness and coherence, it is reproduced here with all notations and constraints but with a bi-objective function as shown below:

Notations:

Parameters	**Definition**
N	number of non-dummy activities in the project
A , $i \in \{1, \ldots\ldots, N+2\}$	set of activities
p_i	processing time of activity A_i
K	total number of skills available
P	total number of available staff members
S , $k \in \{1, \ldots, K\}$	set of skills

$P, m \in \{1, \ldots\ldots, P\}$	set of staff members
$S_{m,k}$	greater than 0 if staff member P_m possesses skill S_k, 0 otherwise
$b_{i,k}$	number of staff members with skill S_k required by activity *i*
t_i	start time of activity *i*
T	project horizon

Decision variables:

$$x_{i,m,t} = \begin{cases} 1; & \text{if staff member } m \text{ starts an activity } i \text{ at time } t, \\ 0 & \text{otherwise} \end{cases}$$

$$y_{i,m,k} = \begin{cases} 1; & \text{if staff member } m \text{ starts an activity } i \text{ with skill } k, \\ 0 & \text{otherwise} \end{cases}$$

$$Z_{i,t} = \begin{cases} 1; & \text{if activity } i \text{ is started at time } t, \\ 0 & \text{otherwise} \end{cases}$$

Mathematical Model for MO-MSRCPSP

Min. $Z_1 = t_{N+2}$ (5.1)

Min. $Z_2 = \sum_{i \in A} \sum_{k \in S} \sum_{m \in P} (y_{i,m,k} \cdot p_i (1 - S_{m,k}))$ (5.2)

Sub. to :

$t_i = \sum_{t \in [0,T]} (Z_{i,t} \cdot t)$ $\forall\ i\ \in A$ (5.3)

$t_i + p_i \leq t_j$ $\forall\ (i,j)\ \in E$ (5.4)

$$\sum_{t\in[0,T]} x_{i,m,t} \leq 1 \qquad \forall\, i \in A,\ \forall m \in P \qquad (5.5)$$

$$\sum_{i\in A}\sum_{d\in[t-p_i+1]} x_{i,m,d} \leq 1 \qquad \forall\, t \in [0,T] \qquad (5.6)$$

$$x_{i,m,t} \leq z_{i,t} \qquad \forall\, i \in A, \forall\, m \in P, \forall\, t \in [0,T] \qquad (5.7)$$

$$x_{i,m,t} + 1 \leq z_{i,t} + \sum_{k\in S} y_{i,m,k} \qquad \forall\, i \in A, \forall\, m \in P, \forall\, t \in [0,T] \qquad (5.8)$$

$$y_{i,m,k} \leq S_{m,k} \qquad \forall\, i \in A, \forall\, m \in P, \forall\, k \in S \qquad (5.9)$$

$$\sum_{m\in P} y_{i,m,k} = b_{i,k} \qquad \forall\, i \in A,\ \forall k \in S \qquad (5.10)$$

$$\sum_{t\in[0,T]} x_{i,m,t} = \sum_{k\in S} y_{i,m,k} \qquad \forall\, i \in A,\ \forall m \in P \qquad (5.11)$$

$$x_{i,m,t} \in \{0,1\} \qquad \forall\, i \in A,\ \forall m \in P\,, \forall\, t \in [0,T] \qquad (5.12)$$

$$y_{i,m,k} \in \{0,1\} \qquad \forall\, i \in A,\ \forall m \in P \qquad (5.13)$$

$$Z_{i,t} \in \{0,1\} \qquad \forall\, i \in A,\ \forall\, t \in [0,T] \qquad (5.14)$$

For the detailed explanation and definition of constraints, one is referred to the section 4.2.1 of chapter 4. It is evident from this revised model that two objectives have been considered for investigation as mentioned below:

1. To minimize the project *makespan* (Z_1)
2. To minimize the total time elapsed with less-skilled resource assignments defined as *Skill Divergence Span* (*SDS*) (Z_2).

It is interesting to note that second objective does not merely consider the minimization of less-skilled resource assignments, rather a 'Skill Divergence Span (*SDS*)' , in terms of total time elapsed with less-skilled assignments, is considered for minimization. The *SDS* is basically the product of processing time of an activity and the corresponding amount of penalty attracted due to the low skill attained by staff member in performing that activity. Mathematically,

$$\text{Skill Divergence Span } (SDS) = y_{i,m,k}.\, p_i.\, (1 - S_{m,k}) \qquad (5.15)$$

where $y_{i,m,k}$ is a binary decision variable whose value is 1 if staff member *m* starts an activity *i* with skill *k* and 0, otherwise.

The advantage of considering *SDS* is that instead of minimizing only the absolute loss due to less-skilled assignments one can minimize the relative time (span) elapsed in less-skilled

assignments which a more reasonable way of capturing performance disparities arising due to assignments of less-skilled resources in a project.

It is easier to understand that the two objectives mentioned above are conflicting to each other. If second objective alone is considered, resource assignments having skill proficiency other than '1' are not preferred for assignments which may increase project duration.

5.2.3 An illustrative example

To understand the nature of the MO-MSRCPSP under investigation, the project instance presented in section 4.2 of chapter 4 is considered again.

Table 5.2: A project instance for the MO-MSRCPSP

Activities	S-1	S-2	S-3	Activity times (p_i)	Number of Successors	Successor Activities	
1	0	0	0	0	2	2	3
2	0	1	1	2	1	4	-
3	1	0	1	5	1	5	-
4	2	1	0	3	1	6	-
5	0	1	0	3	1	6	-
6	0	0	0	0	0	-	-

It comprises of four non-dummy activities linked by precedence relations as shown in Table 5.2. Activity number 1 and 6 are dummy activities i.e. they do not consume any time or resource for their execution. For other activities, there is at least one resource is required for their execution. The information given in above table has been converted into a pictorial representation or network diagram as shown in Figure 5.1.

Unlike the previous model of the single-objective MSRCPSP, a modification is made in this problem in the sense that the skills attained by the staff members have been assumed to have different proficiencies or expertise level (Table 5.3). The value '1' indicates that a staff member masters that particular skill at an expert level while value less than 1 signifies low proficiency to exhibit the skill relative to the expert level. For example, w.r.t. Table 5.3, one can see that staff member 1 is expert in exhibiting skill S_2 while has only 70 % proficiency in skill S_3 and so on.

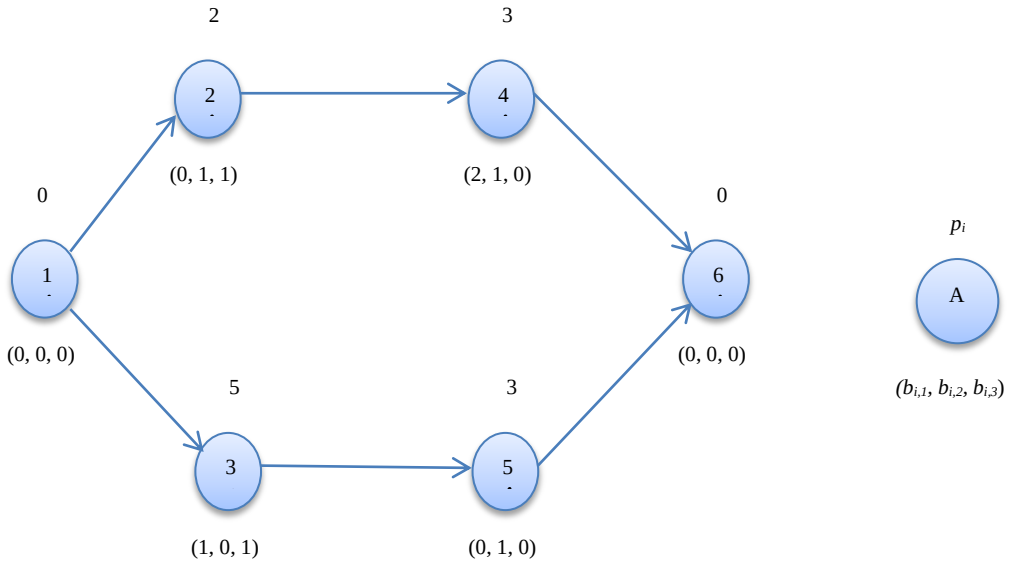


Figure 5.1: Precedence graph of the illustrative project

Table 5.3: Staff-Skill Proficiency Matrix

Staff	Skills attained		
	S_1	S_2	S_3
1	0	1	0.7
2	0.9	0	1
3	1	0	0
4	1	0	0.8

One can recall from chapter 4 that an individual for the MSRCPSP can be conveniently encoded into two parts as shown in Figure 5.2. The first part is in the form of a top horizontal row that determines the relative priorities of the activities which is more popularly known as activity list (AL). The second part comprises of vertical columns corresponding to each activity such that the number of elements in each column is equal to the total number of resources required for the particular activity. In fact, the digits in the columns represent the corresponding index of staff members assigned to perform the said activity. It is easier to understand that no digits in a

particular column are identical due to the constraint (5.11) of mathematical model which ensures that a staff member cannot use more than one skill at a time when assigned to an activity.

1	3	2	4	5	6
	3	1	1	1	
	4	2	2		
			3		

Figure 5.2: A solution (encoded individual) of illustrative project

The procedure of computing the values of two objective functions for this particular instance is illustrated now. In order to calculate the makespan, the already developed modified serial schedule generation scheme (refer chapter 4) is employed. Using this procedure the value of Z_1 i.e. makespan is calculated as 11 time units as shown in Figure 5.3.

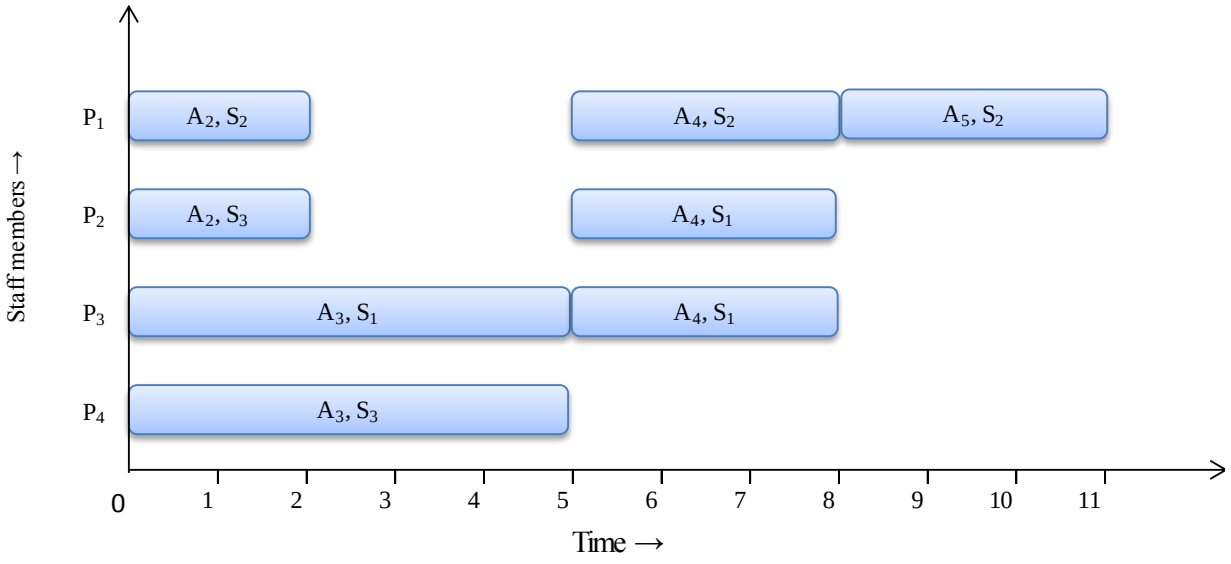


Figure 5.3: A feasible solution of the illustrative example

To calculate the value of second objective function i.e. total time elapsed in less-skilled resource assignments, the *SDS* is computed corresponding to each resource using equation (5.15) as mentioned below:

Skill Divergence Span (*SDS*) = $y_{i,m,k}.p_i.(1 - S_{m,k})$

For this particular problem, i=1,2,3,4,5,6; k =1,2,3; m=1,2,3,4)

From the solution of the instance shown in Figure 5.3, it is evident that following values of skill divergence span (*SDS*) can be conveniently computed for each of the six activities of the project (One should note that calculations are shown only for those non-dummy activities where $y_{i,m,k} = 1$)

SDS_1 = 0 (dummy start activity)

SDS_2 = 1*2* (1-1) + 1*2* (1-1) =0

SDS_3 = 1*5* (1-1) + 1*5* (1-0.8) =1

SDS_4 = 1*3* (1-1) + 1*3* (1-0.9) + 1*3* (1-1) =0.3

SDS_5 = 1*3* (1-1) =0

SDS_6 =0 (dummy end activity)

To calculate the value of second objective function i.e. Z_2, summation of *SDS* values is performed for all activities,

Thus, $Z_2 = \sum_i^{N+2} SDS$ = 0+0+ 1+ 0.3+0+ 0=1.3 time units.

The value of Z_2 quantifies the total amount of time during which resources with skill proficiency less than 1 are engaged in performing project activities. A fraction value is justified by the fact that proficiencies of staff members have been assumed on a continuous scale rather than discrete or hierarchical levels as found in literature. The Z_2 has to be minimized in the algorithm along with makespan.

5.3 Proposed algorithms for solving the MO-MSRCPSP

There are several approaches available in literature to tackle the multi-objective optimization problems (Kalyanmoy D., 2001). These include weighted-sum or scalarization method, ε-constraints method, goal programming, multi-level programming and multi-objective optimization using metaheuristic approaches. As mentioned in chapter 2, this work employs the weighted-sum or scalarization method as it is simple and intuitive. More specifically, the two objectives have been combined by assigning suitable weights and the MO-MSRCPSP is conveniently converted into a single-objective problem by using the TLBO and GA developed in chapter 4. To elaborate further, a weighted-sum or scalarization method for an *n*-objective problem assigns weight w_i to the i^{th} objective function $f_i(x)$ and minimizes a positively weighted sum of all the objectives. Mathematically,

$$Z_3=\text{Min.} \sum_{i=1}^{n} w_i . f_i(x) \tag{5.16}$$

$$\sum_{i=1}^{n} w_i = 1 \tag{5.17}$$

$$w_i > 0, i = 1, \ldots\ldots, n \tag{5.18}$$

Equation (5.16) presents a unique objective function denoted by Z_3. Mathematically, it can be proved that minimization of this new single-objective function can be an efficient solution for the multi-objective problem initially defined by equations (5.1) and (5.2) in the mathematical model of MO-MSRCPSP. More specifically, if the *w* weight vector is greater than zero, the solution obtained by the minimizer Z_3 is a strict Pareto optimum.

There the two basic approaches to handle a multi-objective optimization problem namely priori and posteriori approach. In the former, a decision maker has to assign suitable weights to the objective functions according to his conceptual skill and preference. Actually, one cannot be certain to say that which weights are the most appropriate to obtain a satisfactory solution. Also, the consistent change in the solution cannot be correlated by the changing weights. Decision makers have to try different combinations of weight vectors to touch different portions of the Pareto curve which obviously create a considerable computational burden. In spite of these shortcomings, the priori approach is chosen to solve the MO-MSRCPSP in this work as it is

simple and intuitive. In comparison to minimization of *SDC*, a higher consideration is given to minimization of makespan and hence the weights of the two objective functions have been specified as w_1=0.7 and w_2=0.3. Nevertheless, these weights have been chosen arbitrarily and a decision maker can suitably alter their values depending upon his/her preference of the objective function to be minimized. In the next section the details of the metaheuristic approaches employed to solve the problem under hand have been presented.

5.3.1 A multi-objective TLBO for the MO-MSRCPSP

Looking to the promising results of the TLBO algorithm developed for the single-objective MSRCPSP, the same set of parameters have been applied for the multi-objective MSRCPSP. In chapter 4 the details of the solution representation schemes and implementation methodology of the TLBO can be referred. The following table summarizes the important parameters used for the TLBO developed for the MO-MSRCPSP:

Table 5.4: Summary of the MO-TLBO algorithm

Architecture of the TLBO developed for the MO-MSRCPSP	
Encoding scheme	Activity List (AL) with vertical columns having index of staff members as resource assignment
Decoding scheme	Modified serial schedule generation scheme
Initial population	RBRS sampling method with LFT priority rule
Teacher and Learner phase	Using 2-point crossover mechanism
Self-study phase	Using Boctor's mutation (BM)
Examination phase	Elitism
Test Instances	Generated by using methodology proposed by Almeida et al. (2015) with Staff-Skill Matrix replaced by Staff-Skill Proficiency Matrix.
Parametric details of the TLBO for the MO-MSRCPSP	
Size of initial population (*Class_size*)	40
Probability of self-study (*SS_prob*)	10%
Percentage of elite learners (*Elite_per*)	10 %
Number of instance tested	216
Number of schedules generated per instance	5000

5.3.2 A multi-objective GA for the MO-MSRCPSP

As mentioned earlier there is no reported algorithm in literature for the multi-objective MSRCPSP by considering mixed level of proficiencies of skills having continuous nature. Looking to this aspect, a MO-GA is also developed as an alternative metaheuristic primarily for comparing the results with the proposed TLBO. The basic scheme, operators and implementation framework of the proposed GA on the MO-MSRCPSP are same as designed for the single-objective MSRCPSP. Table 5.5 presents a brief summary of the same.

Table 5.5: Summary of the proposed MO-GA

Architecture of the GA developed for the MO-MSRCPSP	
Encoding scheme	Activity List (AL) with vertical columns having index of staff members as resource assignment
Decoding scheme	Modified serial schedule generation scheme
Initial population	RBRS sampling method with LFT priority rule
Crossover mechanism	2-point crossover mechanism
Mutation mechanism	Boctor's mutation (BM)
Elitism	Ranking based
Selection mechanism	2-tournament method for parents selection
Test Instances	Generated by using methodology proposed by Almeida et al. (2015) with Staff-Skill Matrix replaced by Staff-Skill Proficiency Matrix.
Parametric details of the GA for the MO-MSRCPSP	
Size of initial population (*Class_size*)	40
Crossover probability (*Cross_prob*)	0.80
Mutation Probability (*Mut_prob*)	0.10
Percentage of elite learners (*Elite_per*)	0.10
Number of instance tested	216
Number of schedules generated per instance	5000

5.4 Computational results

To test the behaviour of the two algorithms the algorithms have been coded in MATLAB 7 environment with Core i3 processor having Windows 8.1 and 4GB RAM. The staff-skill matrix

of the 216 different test instances generated for the single-objective MSRCPSP have been modified to incorporate the different skill proficiencies among the staff members. The new matrix is designated as staff-skill-proficiency matrix as shown in Table 5.3. It is ensured that each staff member is 'expert' in at least one skill type. For other skills attained by him, the proficiency level is varied using a random number r such that $r \in \{0.7, 0.8, 0.9\}$.

On the basis of values of parameters mentioned in Tables 5.4 and Table 5.5, the behaviour of both the algorithms have been tested for 216 different instances induced by different values of network complexity (*NC)*, skill factor (*SF*) and modified resource strength (*MRS*) mentioned in chapter 4. A total of 5000 schedules have been generated for both the algorithms as stopping criterion. It is important to note that fitness (objective) function in these problem set is modified by combining two objective functions by selecting a weight vector as $w = (0.7, 0.3)$. As the optimum solutions of these problems are not known, the percentage deviation from critical path based lower bound is calculated which is given as:

$$\% \text{ DEV} = (Z^H - Z^{CP})/Z^{CP} * 100$$

where Z^H is the heuristic solution provided by the algorithm and Z^{CP} is the critical path duration.

The detailed computational results are shown in Table 5.6 whereas summary of these results is exhibited in Table 5.7. In addition, the variation of the different characteristics of instances with respect to avg. % deviation is also shown graphically from Figure 5.5 to Figure 5.7. On the basis of these results, following useful observations can be derived:

- The average % deviation from ctitical path based lower bound obtained is comparatively lower for the MO-TLBO as compared to the MO- GA. It is 61.78% for the proposed MO-TLBO while for MO-GA its value is 74.25%.
- Similar to the results of single-objective MSRCPSP, average % deviation increases with increase in skill factor (SF) (see Figure 5.4) which can be attributed to the fact that due to the increased proportion of resource requirements by activities, number of possible resource combinations increases accordingly.

Table 5.6: Comparison of MO-TLBO and MO-GA

SF	NC	MRS	*P*	MO-TLBO (AVG. % DEV)	MO-GA (AVG. % DEV)
0.5	1.5	0.0667	8	66.28%	86.05%
		0.0750	9	60.47%	81.40%
		0.0917	11	53.49%	74.42%
	1.8	0.0667	8	59.09%	78.79%
		0.0750	9	56.06%	62.88%
		0.0917	11	40.91%	51.52%
	2.1	0.0667	8	58.21%	65.67%
		0.0750	9	47.76%	62.69%
		0.0917	11	35.82%	47.76%
0.75	1.5	0.0667	12	79.07%	90.70%
		0.0778	14	65.12%	86.05%
		0.0944	17	62.79%	80.23%
	1.8	0.0667	12	69.70%	84.85%
		0.0778	14	62.12%	66.67%
		0.0944	17	60.61%	59.09%
	2.1	0.0667	12	61.19%	70.15%
		0.0778	14	52.24%	56.72%
		0.0944	17	40.30%	55.22%
1	1.5	0.0667	16	90.70%	116.28%
		0.0750	18	76.74%	100.00%
		0.0917	22	74.42%	93.02%
	1.8	0.0667	16	78.79%	96.97%
		0.0750	18	75.76%	77.27%
		0.0917	22	71.21%	65.15%
	2.1	0.0667	16	65.67%	77.61%
		0.0750	18	56.72%	68.66%
		0.0917	22	53.73%	62.69%
var.	1.5	0.0667	12	79.07%	93.02%
		0.0778	14	67.44%	83.72%
		0.0944	17	53.49%	79.07%
	1.8	0.0667	12	69.70%	80.30%
		0.0778	14	66.67%	72.73%
		0.0944	17	62.12%	63.64%
	2.1	0.0667	12	58.21%	70.15%
		0.0778	14	52.24%	61.19%
		0.0944	17	40.30%	50.75%
Avg.				**61.78%**	**74.25%**

Table 5.7: Summary of results for MO-TLBO and MO-GA

Parameters	Values	Algorithms	
		TLBO (% DEV)	GA (% DEV)
SF	0.5	53.12	67.91
	0.75	61.46	72.19
	1.0	71.53	84.18
	var.	61.03	72.73
NC	1.5	69.09	88.66
	1.8	64.39	71.66
	2.1	51.87	62.44
MRS	0.0667	69.64	84.21
	0.0750	65.25	75.48
	0.0778	60.97	71.18
	0.9170	54.93	65.76
	0.9440	53.27	64.67

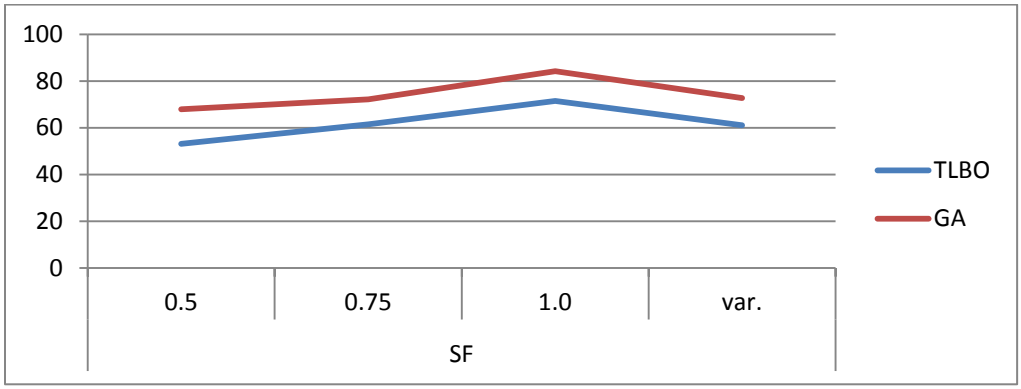


Figure 5.4: Avg. % deviation for different skill factor

- With increase in network complexity (NC), the number of precedence relations also increases. This means less number of activities are available that can be processed

simultaneously, i.e. the degree of parallelization decreases. This in turn results in low values of % deviation as observed from Figure 5.5.

- It can be seen that with the availability of increased number of staff members mastering a particular skill, the resource needs of an activity are satisfied relatively earlier than with reduced value of *MRS* which justifies the low values of % deviation with higher MRS values (Figure 5.6).

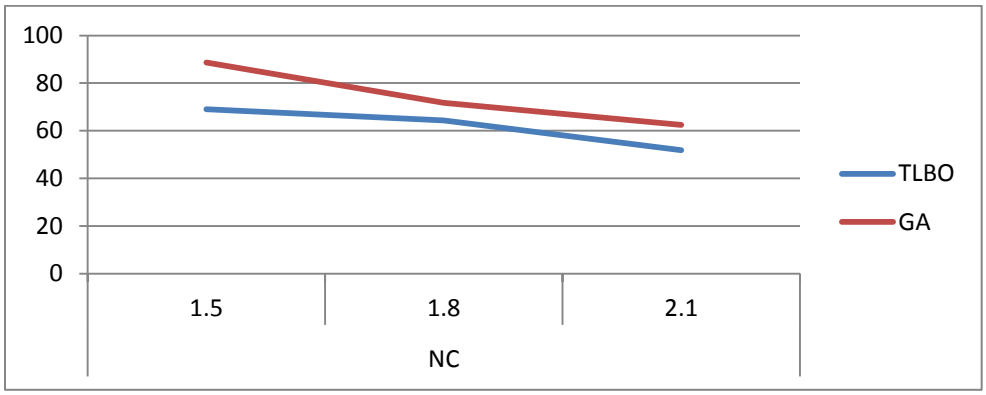


Figure 5.5: Avg. % deviation for different network complexity

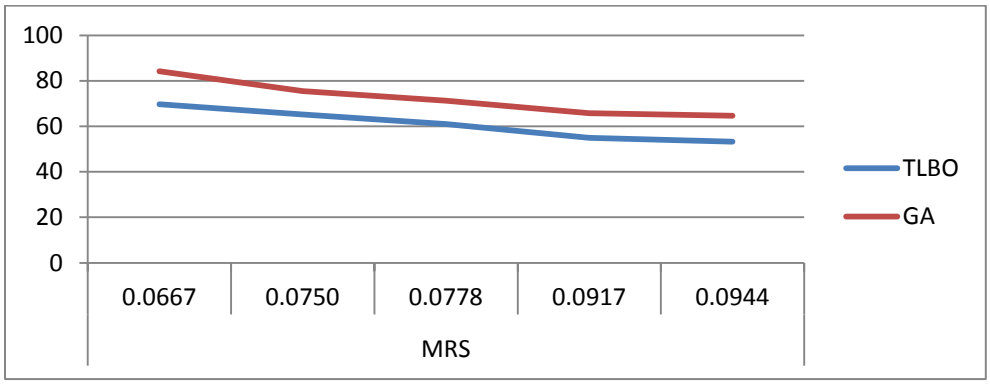


Figure 5.6: Avg. % deviation for different modified resource strength

5.5 Summary

In most of the real world optimization and search problems, it is inevitable to have multiple objectives which are mostly conflicting to each other. In this chapter, a scarcely treated work in

literature about multi-objective multi-skilled resource-constrained project scheduling problem (MO-MSRCPSP) by considering mixed skill proficiencies is investigated. A multi-objective mathematical formulation is presented for this problem which aims to minimize two time estimates; the project makespan and the total time elapsed with less-skilled resource assignments which is defined as total skill divergence span (*SDS*). To solve this complex problem, a priori approach based on weighted-sum or scalarization method is used. The weights given to makespan and *SDS* function are 0.7 and 0.3 respectively which can be arbitrarily modified by decision maker.

To solve this complex problem, two metaheuristics have been proposed namely MO-TLBO and MO-GA. The test instances developed for the MSRCPSP have been modified with mixed proficiency levels of the staff members. The comprehensive test results reveals that the MO-TLBO has performed significantly better than the MO-GA and can be an effective metaheuristic for solving such real life problems.

Chapter 6

Conclusions and future research directions

Chapter 6

Conclusions and future research directions

In this work the resource-constrained project scheduling problem (RCPSP) involving flexible resources is studied. In these problems the resources are 'multi-skilled' i.e. each resource has the functionality of various renewable resources. The problem in literature has been studied under the name multi-skill resource-constrained project scheduling problem (MSRCPSP) in literature.

In chapter 2, a comprehensive literature review is conducted on the MSRCPSP with a brief overview of the RCPSP. As an outcome, it is revealed that although resource limitation aspect has been sufficiently addressed by researchers in project scheduling, the notion of skilled resources still presents a potential area of research. It is found that in most of the research works, a resource or staff member is assumed to possess different skills with same proficiency levels. However, this is not true in real life. Usually in organizations, a staff member possessing various skills may be expert in one (or more) skill(s) but may not be able to exhibit the same level of expertise in all skill types. Moreover, in most of the studies it was found that only one scheduling objective (mostly the makespan) is considered. However, in many practical situations decision-makers need to take care about several objectives simultaneously which give rise to what is known as multi-objective multi-skill resource-constrained project scheduling problem (MO-MSRCPSP). The MO-MSRCPSP is investigated in the later stage of this work.

Although a number of metaheuristic approaches exist in literature but it is interesting to note that no single approach can be guaranteed to give better results for all types of scheduling problems. During the literature review, it has been found that one of the recent metaheuristics for optimization problems is teaching-learning- based algorithm (TLBO) which was introduced by Rao et al. (2011). The TLBO has been reported to have high convergence rate and it also inherits a merit of few algorithm specific parameters to tune (Rao et al., 2011). Inspired by the performance of TLBO on continuous non-linear problems, researchers have also applied it on discrete optimization problems. However, to the best of the knowledge there is no reported work in literature having application of TLBO on standard RCPSP and its multi-skill version with

finite resource requirements and consideration of mixed skill proficiencies. Under this motivation, in this thesis a TLBO algorithm with some modifications is developed as an alternative metaheuristic approach for general class of the RCPSP as well as for the MSRCPSP and its multi-objective case.

Chapter 3 presents a modified TLBO algorithm for the RCPSP as a preliminary approach before its application on the MSRCPSP. For encoding the individuals, a precedence feasible activity list is employed whereas serial generation scheme (SGS) is used as a decoding scheme. To enhance the exploitation and exploration capabilities of the original algorithm, in addition to teacher and learner phase, the concepts of self-study and examination phase have also been utilized in this work. The comprehensive test results on problem instance sets taken from literature show that the developed algorithm is reasonably effective and competitive to other well-known solution approaches for the RCPSP.

Using the basic architecture of the TLBO algorithm for the RCPSP, chapter 4 discusses its extension it for the MSRCPSP. For this purpose, the activity list (AL) is specifically modified by a revised encoding scheme to incorporate the multi-skilled nature of the resources. In addition, a modified schedule generation scheme is also proposed as a decoding procedure to obtain a feasible schedule from a given solution. For the comparison purpose, A GA is also conceptualized as an alternative metaheuristic for solving this problem. The computational study on 216 test instances generated with different characteristics established that the proposed TLBO is comparatively better than the GA.

The research is extended in chapter 5 by investigating the multi-objective MSRCPSP involving staff members having mixed proficiency levels of skill types. To the best of the knowledge, the problem has not been previously addressed in literature in this fashion. A bi-objective mathematical model is designed for minimizing the project makespan and total time elapsed with less-skilled resource assignments defined as *Skill Divergence Span* (*SDS*). For solving this complex problem, a weighted-sum or scalarization method integrated with a MO-TLBO is employed. The results obtained are quite encouraging in contrast to the MO-GA which is also developed in this work for the comparison purpose.

6.1 Major research contributions

The results of the proposed TLBO for the RCPSP and its multi-skill extension i.e. MSRCPSP have already been highlighted in chapters 3, 4 and 5. These are quite promising and encouraging in terms of providing quick near optimal solutions of these problems. In what follows next, some of the major research contributions have been highlighted as fruitful outcomes of this study:

1. A modified version of the teaching-learning-based optimization (TLBO) algorithm has been used to solve the RCPSP with additional phases of self-study and examination to enhance the exploration and exploitation capabilities of the conventional TLBO algorithm.
2. The TLBO algorithm developed in this work is simple to apply and offers less number of algorithm-specific parameters to tune. In particular, it gives competitive results for the RCPSP when compared to several other metaheuristic proposed in the literature to solve this problem.
3. As evident, a new representation scheme of individual (solution) has been used in this work from literature which contains feasible resource assignments for a given MSRCPSP. To handle this new representation, the study brings out a modified schedule generation scheme as a decoding procedure which is capable of avoiding resource-conflicts in multi-skill environment.
4. The TLBO developed for the MSRCPSP can be conveniently applied to a product or service organization wherein human resources are involved in executing project activities. Some typical beneficiaries may include software development companies, consultancy firms, R & D based organizations, maintenance firms, big construction houses etc. which incorporate multi-skilled staff members to accomplish different client orders simultaneously. The developed model can suitably handle resource allocation problems faced in real-life large-sized projects usually administered in these organizations. The research findings can assist the practitioners and managers to recruit and schedule the staff members of their organizations in a much economic way.
5. The research also contributes by developing MATLAB codes for the generation of test instances of a MSRCPSP with varying levels of network complexity (*NC*), skill factor

(*SF*) and modified resource strength (*MRS*). These can be used by other researchers to generate standard or benchmark MSRCPSP instances for testing their algorithms.

6. As another research contribution, the study provides an effective metaheuristic for the multi-objective MSRCPSP involving different proficiency levels of skills administered by the staff members. Besides minimization of project makespan, a novel concept of minimization of *skill divergence span* (*SDS*) is also introduced aims to minimize the total time elapsed with less-skilled resource assignments.

6.2 Limitations of the research

In spite of above-mentioned contributions, the outcome of this research is limited by its scope and applicability. These have been summarized below:

1. The activity times in this work have been assumed to be deterministic and constant which is an unrealistic assumption. In a project scheduling environment where resources are human beings and multi-skilled in nature it is very obvious to assume that a highly-skilled worker can finish an activity earlier as compared to a moderately-skilled or an under-skilled worker. This aspect is not considered in the mathematical models proposed in this study.
2. The resources which are staff members have been assumed to be available throughout the project span without any unavailability periods. This is, however, not true in practical scenario. In many organizations due to the flexible work profiles, human resources are not available uninterruptedly all along the time horizon.
3. The multi-objective approach adopted in this research is based on weighted-sum or scalarization method which has some inherent weaknesses. This approach is not able to find effective Pareto-optimal solutions in problems having non-convex Pareto-optimal region. Moreover, it is highly subjective for a decision maker to decide appropriate weight vectors for a given multi-objective optimization problem. In fact, the final trade-off solution obtained by this method is highly sensitive to the relative preference vector utilized to form a (single) composite objective function. In contrast to this approach,

'ideal' approach for multi-objective optimization is usually favored by researchers as it is more methodical, more practical and less subjective.

6.3 Future directions for the research

This section brings out some interesting areas that may steer new research in the domain of project scheduling with flexible resources. These are motivated from the limitations or restricted scope that is observed in this study. These have been mentioned below:

1. The activity characteristics in a project related to time-switch constraints, preemptibility, time-varying tasks, crashing etc. may be integrated with multi-skilled nature of resources to handle real-life problems faced by many product and service organizations.
2. It will be interesting to consider the time and cost aspects of skilled resources simultaneously. In other words, effective trade-offs can be designed for the project managers to relate the extra costs incurred in employing highly skilled resources with the benefit of relative reduction in the project makespan.
3. Although this work focuses on metaheuristic approaches, a lot of research avenues do exist for developing effective exact approaches to solve the MSRCPSP. In addition, one may think of designing tighter upper and lower bounds for these problems to augment the existing exact procedures available in literature.
4. Furthermore, it will be interesting to perform sensitivity analysis of the solutions obtained in project scheduling environment with flexible resources. This will help the practitioners to devise more robust and reactive schedules along with scope of repairing the current schedule which may be of utmost importance in today's uncertain environment.
5. As another future direction for research, the bi-objective model proposed in this work for the MSRCPSP can be extended to incorporate more objectives inspired from real-life scenario. Non-dominated sorting metaheuristic approaches can be designed to tackle such complex problems.

6. The multi-skill RCPSP environment can be investigated in a multi-project scenario and related complexities of the problem can be tackled by designing more powerful and efficient metaheuristics.

References

References

1. Abdolshah, M. (2014). A review of resource-constrained project scheduling problems (RCPSP) approaches and solutions. *International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies*, *5*, 253-286.
2. Agarwal, A., Colak, S., & Erenguc, S. (2011). A neurogenetic approach for the resource-constrained project scheduling problem. *Computers & Operations Research*, *38*(1), 44-50.
3. Alcaraz, J., & Maroto, C. (2001). A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, *102*(1-4), 83-109.
4. Alcaraz, J., Maroto, C., & Ruiz, R. (2004). Improving the performance of genetic algorithms for the RCPS problem. In *Proceedings of the ninth international workshop on project management and scheduling* (Vol. 40, p. 43).
5. Alvarez-Valdes, R., Crespo, E., & Tamarit, J. M. (2002). Design and implementation of a course scheduling system using Tabu Search. *European Journal of Operational Research*, *137*(3), 512-523.
6. Almeida, B. F., Correia, I., & Saldanha-da-Gama, F. (2015). An instance generator for the multi-skill resource-constrained project scheduling problem.
7. Almeida, B. F., Correia, I., & Saldanha-da-Gama, F. (2016). Priority-based heuristics for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications*, *57*, 91-103.
8. Almeida, B. F., Correia, I., & Saldanha-da-Gama, F. (2018). A biased random-key genetic algorithm for the project scheduling problem with flexible resources. *Top*, *26* (2), 283-308.
9. Atkinson, R., Crawford, L., & Ward, S. (2006). Fundamental uncertainties in projects and the scope of project management. *International Journal of Project Management*, *24*(8), 687-698.
10. Beach, R., Muhlemann, A.P., Price, D.H.R., Paterson, A., Sharp, J.A., (2000). A review of manufacturing flexibility. *European Journal of Operational Research*, *122* (1), 41–57.
11. Bianco, L., Dell'Olmo, P., & Speranza, M. G. (1998). Heuristics for multimode scheduling problems with dedicated resources. *European Journal of Operational Research*, *107*(2), 260-271.

12. Bellenguez, O., & Néron, E. (2004, August). Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills. In *International Conference on the Practice and Theory of Automated Timetabling* (pp. 229-243). Springer, Berlin, Heidelberg.
13. Bellenguez-Morineau, O., & Néron, E. (2007). A branch-and-bound method for solving multi-skill project scheduling problem. *RAIRO-Operations Research*, *41*(2), 155-170.
14. Błażewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., & Węglarz, J. (2001). Scheduling under Resource Constraints. In *Scheduling Computer and Manufacturing Processes* (pp. 317-365). Springer, Berlin, Heidelberg.
15. Blazewicz, J., Lenstra, J. K., & Kan, A. R. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, *5*(1), 11-24.
16. Boctor, F. F. (1990). Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research*, *49*(1), 3-13.
17. Boctor, F. F. (1996). Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, *34*(8), 2335-2351.
18. Bordoloi, S.K., Cooper, W.W., Matsuo, H., 1999. Flexibility, adaptability, and efficiency in manufacturing systems. *Production and Operations Management*, *8*(2), 133–150.
19. Böttcher, J., Drexl, A., Kolisch, R., & Salewski, F. (1999). Project scheduling under partially renewable resource constraints. Management Science, *45*(4), 543-559.
20. Bouleimen, K. L. E. I. N., & Lecocq, H. O. U. S. N. I. (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.
21. Brucker, P., Knust, S., Schoo, A., & Thiele, O. (1998). A branch and bound algorithm for the resource-constrained project scheduling problem1. *European Journal of Operational Research*, *107*(2), 272-288.
22. Bukata, L., Šůcha, P., & Hanzálek, Z. (2015). Solving the resource constrained project scheduling problem using the parallel tabu search designed for the CUDA platform. *Journal of Parallel and Distributed Computing*, *77*, 58-68.
23. Cai, X., & Li, K. N. (2000). A genetic algorithm for scheduling staff of mixed skills under multi-criteria. *European Journal of Operational Research*, *125*(2), 359-369.

24. Chen, W., Shi, Y. J., Teng, H. F., Lan, X. P., & Hu, L. C. (2010). An efficient hybrid algorithm for resource-constrained project scheduling. *Information Sciences*, *180*(6), 1031-1039.
25. Cho, J. H., & Kim, Y. D. (1997). A simulated annealing algorithm for resource constrained project scheduling problems. *Journal of the Operational Research Society*, *48*(7), 736-744.
26. Cordeau, J. F., Laporte, G., Pasin, F., & Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, *13*(4), 393-409.
27. Correia, I., Lourenço, L. L., & Saldanha-da-Gama, F. (2012). Project scheduling with flexible resources: formulation and inequalities. *OR spectrum*, *34*(3), 635-663.
28. Dauzère-Pérès, S., Roux, W., & Lasserre, J. B. (1998). Multi-resource shop scheduling with resource flexibility. *European Journal of Operational Research*, *107*(2), 289-305.
29. De Boer, R. (1998). Resource-constrained multi-project management (Doctoral dissertation, PhD thesis, University of Twente, The Netherlands).
30. De Groote, X. (1994). The flexibility of production processes: a general framework. *Management Science*, *40*(7), 933-945.
31. De Reyck, B., & Herroelen, W. (1999). The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, *119*(2), 538-556.
32. Demeulemeester, E., & Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, *38*(12), 1803-1818.
33. Duin, C. W., & Van der Sluis, E. (2006). On the complexity of adjacent resource scheduling. *Journal of Scheduling*, *9*(1), 49-62.
34. Ernst, A. T., Jiang, H., Krishnamoorthy, M., Owens, B., & Sier, D. (2004). An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, *127*(1-4), 21-144.
35. Ernst, A. T., Jiang, H., Krishnamoorthy, M., & Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, *153*(1), 3-27.

36. Eshraghi, A. (2016).A new approach for solving resource constrained project scheduling problems using differential evolution algorithm. *International Journal of Industrial Engineering Computations*, *7*(2), 205-216.
37. Fang, C., & Wang, L. (2012).An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem. *Computers & Operations Research*, *39*(5), 890-901.
38. Firat, M., & Hurkens, C. A. J. (2012).An improved MIP-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, *15*(3), 363-380.
39. Gerwin, D., 1987. An agenda for research on the flexibility of manufacturing processes. *International Journal of Operations and Production Management*, *7* (1), 39-49.
40. Giran, O., Temur, R., & Bekdaş, G. (2017). Resource constrained project scheduling by harmony search algorithm. *KSCE Journal of Civil Engineering*, *21*(2), 479-487.
41. Golden, W., Powell, P., 2000. Towards a definition of flexibility: in search of the Holy Grail? *Omega*, *28*(4), 373–384.
42. Goudswaard, A., & De Nanteuil, M. (2000). Flexibility and working conditions: a european bibliographical review. *Dublin: European Foundation for the Improvement of Living and Working Conditions*.
43. Gürbüza, E. (2010). *Genetic algorithm for bi-objective multi-skill project scheduling problem with hierarchical levels of skills* (Doctoral dissertation, Thesis on industrial engineering department. Middle East Technical University, Turkey).
44. Gutjahr, W. J., Katzensteiner, S., Reiter, P., Stummer, C., & Denk, M. (2008). Competence-driven project portfolio selection, scheduling and staff assignment. *Central European Journal of Operations Research*, *16*(3), 281-306.
45. Josefowska, J., Mika, M., Rozycki, R., Waligora, G., & Weglarz, J. (2001). Simulated annealing for multi-mode resource-constrained project scheduling problem. *Annals of Operational Research*, *102*, 137-155.
46. Habibi, F., Barzinpour, F., & Sadjadi, S. (2018). Resource-constrained project scheduling problem: review of past and recent developments. *Journal of Project Management*, *3*(2), 55-88.
47. Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics* (*NRL*), *45*(7), 733-750.

48. Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics* (*NRL*), 49(5), 433-448.
49. 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.
50. Hegazy, T., Shabeeb, A. K., Elbeltagi, E., & Cheema, T. (2000). Algorithm for scheduling with multiskilled constrained resources. *Journal of Construction Engineering and Management*, *126*(6), 414-421.
51. Heimerl, C., & Kolisch, R. (2010). Scheduling and staffing multiple projects with a multi-skilled workforce. *OR spectrum*, 32(2), 343-368.
52. Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, *165*(2), 289-306.
53. Javanmard, S., Afshar-Nadjafi, B., & Niaki, S. T. A. (2017). Preemptive multi-skilled resource investment project scheduling problem: Mathematical modelling and solution approaches. *Computers & Chemical Engineering*, *96*, 55-68.
54. Kadrou, Y., & Najid, N. M. (2006, October). A new heuristic to solve RCPSP with multiple execution modes and Multi-Skilled Labor. In *The Proceedings of the Multi-conference on* "Computational Engineering in Systems Applications" (Vol. 2, pp. 1302-1309). IEEE.
55. Kalyanmoy, D. (2001). Multi objective optimization using evolutionary algorithms (pp. 124-124). John Wiley and Sons.
56. Kazemipoor, H., Tavakkoli-Moghaddam, R., & Shahnazari-Shahrezaei, P. (2002). Solving a mixed-integer linear programming model for a multi-skilled project scheduling problem by simulated annealing. *Management Science Letters*, *2*(2), 681-688.
57. Kelley Jr, J. E., & Walker, M. R. (1959, December). Critical-path planning and scheduling. In *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference* (pp. 160-173). ACM.
58. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671-680.
59. Klein, R. (2000). Project scheduling with time-varying resource constraints. *International Journal of Production Research*, *38*(16), 3937-3952.

60. Kochetov, Y., & Stolyar, A. (2003). Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In *Proceedings of the 3rd international workshop of computer science and information technologies* (*Vol. 132*).
61. Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, *90*(2), 320-333.
62. Kolisch, R., & Drexl, A. (1996). Adaptive search for solving hard project scheduling problems. *Naval Research Logistics* (*NRL*), 43(1), 23-40.
63. Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In *Project scheduling* (pp. 147-178). Springer, Boston, MA.
64. Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, *174*(1), 23-37.
65. 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.
66. Koste, L.L., & Malhotra, M.K. (1999). A theoretical framework for analyzing the dimensions of manufacturing flexibility. *Journal of Operations Management, 18*(1), 75–93.
67. Lacomme, P., Larabi, M., & Tchernev, N. (2013). Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Economics*, *143*(1), 24-34.
68. Leon, V. J., & Balakrishnan, R. (1995). Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. *Operations-Research-Spektrum*, *17*(2-3), 173-182.
69. Li, K. Y., & Willis, R. J. (1992). An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research*, *56*(3), 370-379.
70. Li, H., & Womer, K. (2009). Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm. *Journal of Scheduling*, *12*(3), 281.
71. Liu, S. S., & Wang, C. J. (2012). Optimizing linear project scheduling with multi-skilled crews. *Automation in Construction*, *24*, 16-23.

72. Maghsoudlou, H., Afshar-Nadjafi, B., & Niaki, S. T. A. (2016). A multi-objective invasive weeds optimization algorithm for solving multi-skill multi-mode resource constrained project scheduling problem. *Computers & Chemical Engineering*, *88*, 157-169.
73. Malcolm, D. G., Roseboom, J. H., Clark, C. E., & Fazar, W. (1959). Application of a technique for research and development program evaluation. *Operations Research*, *7*(5), 646-669.
74. Mellentien, C., Schwindt, C., & Trautmann, N. (2004). Scheduling the factory pick-up of new cars. *OR Spectrum*, *26*(4), 579-601.
75. Mendes, J. J. D. M., Gonçalves, J. F., & Resende, M. G. (2009). A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, *36*(1), 92-109.
76. Myszkowski, P. B., Laszczyk, M., Nikulin, I., & Skowroński, M. (2018). iMOPSE: a library for bicriteria optimization in Multi-Skill Resource-Constrained Project Scheduling Problem. *Soft Computing*, 1-14.
77. Myszkowski, P. B., Olech, Ł. P., Laszczyk, M., & Skowroński, M. E. (2018). Hybrid Differential Evolution and Greedy Algorithm (DEGR) for solving Multi-Skill Resource-Constrained Project Scheduling Problem. *Applied Soft Computing*, *62*, 1-14.
78. Myszkowski, P. B., Skowroński, M. E., Olech, Ł. P., & Oślizło, K. (2015). Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem. *Soft Computing*, *19*(12), 3599-3619.
79. Naber, A., & Kolisch, R. (2014). MIP models for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, *239*(2), 335-348.
80. Néron, E. (2002, April). Lower bounds for the multi-skill project scheduling problem. In *Proceeding of the Eighth International Workshop on Project Management and Scheduling* (pp. 274-277).
81. Neumann, K., Schwindt, C., & Trautmann, N. (2002). Advanced production scheduling for batch plants in process industries. *OR spectrum*, *24*(3), 251-279.
82. Neumann, K., Schwindt, C., & Zimmermann, J. (2006). Resource-constrained project scheduling with time windows. In *Perspectives in modern project scheduling* (pp. 375-407). Springer, Boston, MA.

83. Nonobe, K., & Ibaraki, T. (2002). Formulation and tabu search algorithm for the resource constrained project scheduling problem. In *Essays and surveys in metaheuristics* (pp. 557-588). Springer, Boston, MA.

84. Olsson, N. O. (2006). Management of flexibility in projects. *International Journal of Project Management*, *24*(1), 66-74.

85. Pan, N. H., Hsaio, P. W., & Chen, K. Y. (2008). A study of project scheduling optimization using Tabu Search algorithm. *Engineering Applications of Artificial Intelligence*, *21*(7), 1101-1112.

86. Patterson, J. H., Brian Talbot, F., Slowinski, R., & Weglarz, J. (1990). Computational experience with a backtracking algorithm for solving a general class of precedence and resource-constrained scheduling problems. *European Journal of Operational Research*, 49, 68-79.

87. Patterson, J. H., & Roth, G. W. (1976). Scheduling a project under multiple resource constraints: a zero-one programming approach. *AIIE Transactions*, *8*(4), 449-455.

88. Ranjbar, M., & Kianfar, F. (2010). Resource-constrained project scheduling problem with flexible work profiles: a genetic algorithm approach. *Scientia Iranica, Transaction E, Industrial Engineering*, *17*(1), 25.

89. Rao, R., & Patel, V. (2012). An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems. *International Journal of Industrial Engineering Computations*, *3*(4), 535-560.

90. Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, *43*(3), 303-315.

91. Santos, M. A., & Tereso, A. P. (2010). On the multi-mode, multi-skill resource constrained project scheduling problem (MRCPSP-MS). In *2nd International Conference on Engineering Optimization* (*EngOpt2010*).

92. Schirmer, A. (2000). Case-based reasoning and improved adaptive search for project scheduling. *Naval Research Logistics* (*NRL*), 47(3), 201-222.

93. Shewchuk, J. P., & Chang, T. C. (1995). Resource-constrained job scheduling with recyclable resources. *European Journal of Operational Research*, *81*(2), 364-375.

94. Talbot, F. B. (1982). Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science, 28*(10), 1197-1210.
95. Thomas, P. R., & Salhi, S. (1998). A tabu search approach for the resource constrained project scheduling problem. *Journal of Heuristics, 4*(2), 123-139.
96. Tiwari, V., Patterson, J. H., & Mabert, V. A. (2009). Scheduling projects with heterogeneous resources to meet time and quality objectives. *European Journal of Operational Research, 193*(3), 780-790.
97. Tormos, P., & Lova, A. (2001). A competitive heuristic solution technique for resource-constrained project scheduling. *Annals of Operations Research, 102*(1-4), 65-81.
98. Tseng, L. Y., & Chen, S. C. (2006). A hybrid metaheuristic for the resource-constrained project scheduling problem. *European Journal of Operational Research, 175*(2), 707-721.
99. 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.
100. Valls, V., Pérez, Á., & Quintanilla, S. (2009). Skilled workforce scheduling in service centres. *European Journal of Operational Research, 193*(3), 791-804.
101. Wang, L., & Zheng, X. L. (2018). A knowledge-guided multi-objective fruit fly optimization algorithm for the multi-skill resource constrained project scheduling problem. *Swarm and Evolutionary Computation, 38*, 54-63.
102. Węglarz, J., Józefowska, J., Mika, M., & Waligóra, G. (2011). Project scheduling with finite or infinite number of activity processing modes–A survey. *European Journal of Operational Research, 208*(3), 177-205.
103. Wiest, J. D. (1967). A heuristic model for scheduling large projects with limited resources. *Management Science*, *13*(6), B-359.
104. Wongwai, N., & Malaikrisanachalee, S. (2011). Augmented heuristic algorithm for multi-skilled resource scheduling. *Automation in Construction, 20*(4), 429-445.
105. 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.
106. Zhang, H., Li, X., Li, H., & Huang, F. (2005). Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction, 14*(3), 393-404.

107. Zhang, H., Li, H., & Tam, C. M. (2006). Particle swarm optimization for resource-constrained project scheduling. *International Journal of Project Management*, *24*(1), 83-92.
108. Zheng, H. Y., & Wang, L. (2015). An effective teaching–learning-based optimisation algorithm for RCPSP with ordinal interval numbers. *International Journal of Production Research*, *53*(6), 1777-1790.
109. Zheng, H. Y., Wang, L., & Zheng, X. L. (2017). Teaching–learning-based optimization algorithm for multi-skill resource constrained project scheduling problem. *Soft Computing*, *21*(6), 1537-1548.
110. Zou, F., Chen, D., & Xu, Q. (2019). A survey of teaching–learning-based optimization. *Neurocomputing*, 335, 366-383.

Appendices

Appendix-I: Problem Instance formats

In this appendix, the sample instances of the RCPSP and MSRCPSP used in computational experiments are provided.

(a) Sample .txt input file format for the RCPSP instance:

Row 1: No. of activities; No. of renewable resource types
Row 2: Maximum availability of each renewable resource
Row 3 onwards: Activity resource requirements (first 4 columns); No. of successors; Successor activities

```
32  4
12  13  4  12
0   0   0  0   0  3  2  3  4
8   4   0  0   0  3  6  11 15
4   10  0  0   0  3  7  8  13
6   0   0  0   3  3  5  9  10
3   3   0  0   0  1  20
8   0   0  0   8  1  30
5   4   0  0   0  1  27
9   0   1  0   0  3  12 19 27
2   6   0  0   0  1  14
7   0   0  0   1  2  16 25
9   0   5  0   0  2  20 26
2   0   7  0   0  1  14
6   4   0  0   0  2  17 18
3   0   8  0   0  1  17
9   3   0  0   0  1  25
10  0   0  0   5  2  21 22
6   0   0  0   8  1  22
5   0   0  0   7  2  20 22
3   0   1  0   0  2  24 29
7   0   10 0   0  2  23 25
2   0   0  0   6  1  28
7   2   0  0   0  1  23
2   3   0  0   0  1  24
3   0   9  0   0  1  30
3   4   0  0   0  1  30
7   0   0  4   0  1  31
8   0   0  0   7  1  28
3   0   8  0   0  1  31
7   0   7  0   0  1  32
2   0   7  0   0  1  32
2   0   0  2   0  1  32
0   0   0  0   0  0
```

(b) Sample .txt input file format for the MSRCPSP instance:

Number of Activities	S-1	S-2	S-3	S-4	Processing Time	Number of Successors	Successor Activities			Number of staff (P)	S-1	S-2	S-3	S-4
1	0	0	0	0	0	3	2	3	4	1	1	1	1	0
2	0	2	2	0	3	3	12	19	23	2	1	0	0	1
3	3	0	0	1	7	3	8	13	16	3	0	1	0	1
4	0	3	3	0	10	3	5	6	22	4	1	1	1	0
5	0	3	0	1	9	3	14	15	17	5	1	0	1	1
6	3	0	3	0	6	3	7	13	16	6	0	1	1	1
7	3	2	0	0	6	3	15	21	26	7	0	0	1	0
8	0	2	0	1	8	3	9	10	14	8	1	0	0	0
9	0	3	2	0	1	3	11	19	21	9	1	1	1	0
10	1	0	3	0	7	3	11	12	15	10	1	0	1	1
11	0	1	1	0	6	2	17	26	0	11	0	1	1	1
12	2	3	0	0	10	3	21	22	25	0	0	0	0	0
13	3	0	3	0	8	3	14	17	23	0	0	0	0	0
14	0	0	2	2	5	3	18	24	27	0	0	0	0	0
15	0	2	0	1	3	3	18	20	23	0	0	0	0	0
16	0	1	0	3	3	2	18	25	0	0	0	0	0	0
17	1	0	0	3	4	1	20	0	0	0	0	0	0	0
18	0	0	2	2	3	3	19	29	30	0	0	0	0	0
19	2	0	2	0	3	1	28	0	0	0	0	0	0	0
20	2	0	3	0	1	1	25	0	0	0	0	0	0	0
21	1	1	0	0	9	3	24	29	30	0	0	0	0	0
22	3	0	0	2	5	2	24	26	0	0	0	0	0	0
23	2	0	0	1	5	1	27	0	0	0	0	0	0	0
24	0	0	3	2	10	1	28	0	0	0	0	0	0	0
25	0	1	2	0	10	2	27	29	0	0	0	0	0	0
26	0	0	2	1	4	2	28	30	0	0	0	0	0	0
27	1	0	0	1	5	1	31	0	0	0	0	0	0	0
28	1	0	3	0	9	1	31	0	0	0	0	0	0	0
29	0	0	2	1	1	1	32	0	0	0	0	0	0	0
30	0	2	0	2	1	1	32	0	0	0	0	0	0	0
31	0	0	2	2	7	1	32	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(c) Sample .txt input file format for the MO-MSRCPSP instance (Staff with skill proficiency levels):

Number of Activities	S-1	S-2	S-3	S-4	Processing Time	Number of Successors	Successor Activities			Number of staff (P)	S-1	S-2	S-3	S-4
1	0	0	0	0	0	3	2	3	4	1	1	0.8	0.7	0
2	0	1	2	0	3	3	12	19	23	2	0.9	0	0	1
3	3	0	0	1	7	3	8	13	16	3	0	0.8	0	1
4	0	3	3	0	6	3	5	6	22	4	1	0.7	0.8	0
5	0	3	0	1	9	3	14	15	17	5	0.9	0	1	0.7
6	3	0	3	0	6	3	7	13	16	6	0	0.8	0.7	1
7	2	2	0	0	6	3	15	21	26	7	0	0	1	0
8	0	2	0	1	8	3	9	10	14	8	0.8	0	0	0
9	0	3	2	0	1	3	11	19	21	9	1	0.7	1	0
10	1	0	3	0	7	3	11	12	15	10	0.7	0	1	0.9
11	0	1	1	0	6	2	17	26	0	11	0	1	0.7	0.8
12	2	3	0	0	8	3	21	22	25	0	0	0	0	0
13	3	0	3	0	8	3	14	17	23	0	0	0	0	0
14	0	0	2	2	5	3	18	24	27	0	0	0	0	0
15	0	2	0	1	3	3	18	20	23	0	0	0	0	0
16	0	1	0	3	3	2	18	25	0	0	0	0	0	0
17	1	0	0	3	10	1	20	0	0	0	0	0	0	0
18	0	0	2	2	3	3	19	29	30	0	0	0	0	0
19	2	0	2	0	3	1	28	0	0	0	0	0	0	0
20	2	0	3	0	1	1	25	0	0	0	0	0	0	0
21	1	1	0	0	9	3	24	29	30	0	0	0	0	0
22	3	0	0	2	5	2	24	26	0	0	0	0	0	0
23	2	0	0	1	5	1	27	0	0	0	0	0	0	0
24	0	0	3	2	6	1	28	0	0	0	0	0	0	0
25	0	1	2	0	10	2	27	29	0	0	0	0	0	0
26	0	0	2	1	4	2	28	30	0	0	0	0	0	0
27	1	0	0	1	5	1	31	0	0	0	0	0	0	0
28	1	0	3	0	9	1	31	0	0	0	0	0	0	0
29	0	0	2	1	1	1	32	0	0	0	0	0	0	0
30	0	2	0	2	5	1	32	0	0	0	0	0	0	0
31	0	0	2	2	7	1	32	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Appendix II: MATLAB Codes

This appendix presents the MATLAB codes developed in this work for the RCPSP, MSRCPSP and MO-MSRCPSP.

%%% Codes for initialization and calling the RCPSP instances in required formats %%%

```
clc;
clear all;
Npop=80;
No_iterations=20;
A=textread('J30_Optimum Makespan.txt');
folder='D:\PhD\PhD MATLAB\J30_480';
filetype='*.txt';
f=fullfile(folder,filetype);
d=dir(f);
A1=(A(1:(numel(d)),3));
for k=1:numel(d)
fprintf('Problem Instance: J30_0%d\nOptimum Makespan from PSPLIB= %d\n',k,A1(k))
    MATRIX{k}=dlmread(fullfile(folder,d(k).name));
PROBLEM_MATRIX=MATRIX{1,k};
ACTmtx=[(1:32)',ones(32,1),PROBLEM_MATRIX(3:34,1),PROBLEM_MATRIX(3:34,2:5)];
LIMmtx=PROBLEM_MATRIX(2,1:4);
RELmtx=[(1:32)',ones(32,1),PROBLEM_MATRIX(3:34,6),PROBLEM_MATRIX(3:34,7:9)];
NumAct=size(RELmtx,1);
PREDmtx=zeros(NumAct,max(RELmtx(:,3))+1); % This step creates a skeleton of
precedence matrix with all zeros
LimLen=length(LIMmtx);
% LIMmtx=LIMmtx.*1000;
CeilDur=sum(ACTmtx(:,3))+1;
RESLIMmtx=zeros(LimLen,CeilDur);

for i=1:LimLen
    RESLIMmtx(i,1:CeilDur)=LIMmtx(1,i);
end

%%%Creating a precedence matrix %%%%%%

for i=4:size(RELmtx,2)
    for j=1:NumAct
        if RELmtx(j,i)~=0
            PREDmtx(RELmtx(j,i),1)=PREDmtx(RELmtx(j,i),1)+1;
            PREDmtx(RELmtx(j,i),PREDmtx(RELmtx(j,i),1)+1)=RELmtx(j,1);
        end
    end
end
```

%%% Codes for applying TLBO philosophy for the RCPSP %%%

```
function[Makespan_tlbo,Student]=tlbo(Npop,NumAct,Student,PREDmtx,ACTmtx,RELmtx,
RESLIMmtx)
```

```
Makespan1=zeros(Npop,1);
for i=1:Npop
    Student1=Student(i,:);
    [Makespan1(i,:)]=SSS(NumAct,Student1,PREDmtx,ACTmtx,RESLIMmtx);
end
% Makespan1;
[~,k0]=min(Makespan1); % k0 is the index of best student in the Class which will be
designated as Teacher

%%% Start of Teacher Phase %%%
Teacher=Student(k0,:);
Student1_tp=zeros(1,NumAct);
Student2_tp=zeros(1,NumAct);
StudentNew_tp=zeros(1,NumAct);
Makespan_tp=zeros(Npop,1);
for i=1:Npop
    Student1_tp(i,:)=Student(i,:);
    Student2_tp(i,:)=Teacher;
    StudentNew_tp(i,:)=Crossover(Student1_tp(i,:),Student2_tp(i,:),NumAct);
    Makespan_tp(i,:)=SSS(NumAct,StudentNew_tp(i,:),PREDmtx,ACTmtx,RESLIMmtx);
    if Makespan_tp(i,:)< Makespan1(i,:);
        Student1_tp(i,:)=StudentNew_tp(i,:);
    else
        Student1_tp(i,:)=Student(i,:);
        Makespan_tp(i,:)=Makespan1(i,:);
    end
end

%%% Start of Student Phase %%%
Student1_sp=zeros(1,NumAct);
Student2_sp=zeros(1,NumAct);
StudentNew_sp=zeros(1,NumAct);
Makespan_sp=zeros(Npop,1);
r=randperm(Npop);
for i=1:Npop
    Student1_sp(i,:)=Student1_tp(i,:);
    Student2_sp(i,:)=Student1_tp(r(i),:);
  if Makespan_tp(i,:)> Makespan_tp(r(i),:);
    StudentNew_sp(i,:)=Crossover(Student1_sp(i,:),Student2_sp(i,:),NumAct);
  else
    StudentNew_sp(i,:)=Crossover(Student2_sp(i,:),Student1_sp(i,:), NumAct);
  end
    Makespan_sp(i,:)=SSS(NumAct,StudentNew_sp(i,:),PREDmtx,ACTmtx,RESLIMmtx);
     if Makespan_sp(i,:)< Makespan_tp(i,:);
        Student1_sp(i,:)=StudentNew_sp(i,:);
    else
       Student1_sp(i,:)=Student1_sp(i,:);
       Makespan_sp(i,:)=Makespan_tp(i,:);
     end
end
Student= Student1_sp;
Makespan_tlbo= min(Makespan_sp);
```

%%% Start of self-study phase by applying mutation in the TLBO %%%

```
Makespan_mut=zeros(Npop,1);
for i=1:Npop
    Student(i,:)= mutation(NumAct,Student1_sp(i,:),RELmtx);
    Makespan_mut(i,:)=SSS(NumAct,Student(i,:),PREDmtx,ACTmtx,RESLIMmtx);
    if Makespan_mut(i,:)< Makespan_sp(i,:)
        Student(i,:)=Student(i,:);
        Makespan_mut(i,:)=Makespan_mut(i,:);
    else
        Student(i,:)=Student1_sp(i,:);
        Makespan_mut(i,:)=Makespan_sp(i,:);
    end
    Makespan_tlbo= min(Makespan_mut);
end
fprintf('TLBO DETAILED MATRIX AFTER ITERATION')
```

%%% Applying elitism i.e. replacing worst solutions by elite solutions %%%

```
e=4; % elite size
[~, index]=sort(Makespan_sp);
for i=1:Npop
    Student1_sp(i,:)=Student1_sp(index(i,1),:);
end
Student1_sp((Npop-e+1:Npop),:)=Student1_sp(1:e,:); % Replacing worst solutions with
elite solutions
```

%%% A function file for Serial Schedule Generation Scheme (SSS) for the RCPSP %%%

```
function[Makespan]=SSS(NumAct,Student,PREDmtx,ACTmtx,RESLIMmtx)
FDmtx=zeros(NumAct,1);
 while ismember(0,FDmtx)
     i=1;
     while i<=NumAct
        if FDmtx(Student(i))==0
            Npre=PREDmtx(Student(i),1);
            FDpre=zeros(Npre,1);

            if Npre~=0
                FDpre=zeros(Npre,1);
                for j=1:Npre
                    if FDmtx(PREDmtx(Student(i),1+j))~=0
                        FDpre(j)=FDmtx(PREDmtx(Student(i),1+j));
                    end
                end
            else
            % If no predecessors Studentlow activity to schedule
            FDpre=zeros(1,1);
            FDpre(1)=1;
            end

            if not(ismember(0, FDpre))
                EarST = max(FDpre);
```

```
                ACTdur = ACTmtx(Student(i),3);
                RESlim =(3 + size(RESLIMmtx,1));
                ACTres(1: size(RESLIMmtx,1)) = ACTmtx(Student(i),4:RESlim);
                ResFes = 0;
                while   ResFes == 0;
                        EarFI = EarST + ACTdur;
                         for k=1: size(RESLIMmtx,1)
                                 z(k,:)=RESLIMmtx(k, EarST+1:EarFI)>=ACTres(1,k);
                          end
                             if not(ismember(0,z))
                                    ResFes=1 ;  %Record Activity Finish Date in
                                       Matrix based on  %resource 'k' requirement.
                                    z=[];
                             else
                                    ResFes =0;
                                    EarST = EarST+1;
                             end
                end
                if  ResFes==1
                    for k=1: size(RESLIMmtx,1)
                         RESLIMmtx(k, EarFI-ACTdur+1:EarFI)=RESLIMmtx(k,EarFI-
                          ACTdur+1:EarFI)-ACTres(k);
                    end
                end
                  FDmtx(Student(i)) = EarFI;
                  i=0;
         end
        end
        i=i+1;
     end
 end
 FDmtx(:) = FDmtx(:)-1;
 Makespan= max(FDmtx);
SDmtx=zeros(NumAct,1);
for k=1:NumAct
SDmtx(k)=FDmtx(k)-ACTmtx(k,3);
end
Activity_Duration_StartTime_FinishTime=[(1:NumAct)' ACTmtx(:,3) SDmtx FDmtx];
Makespan= max(FDmtx);
```

%%% Codes for creating the 'mutation' function in TLBO and GA %%%

```
%% Hartmann (1998) mutation

function[Student]=mutation(NumAct,Student,RELmtx)
mut_prob=0.05;
for i=2:NumAct-2
    r=rand;
     if r< mut_prob
         PosA=i;
         PosB=i+1;
```

```
         ActA=Student(PosA);
         ActB=Student(PosB);
         if not(ismember(ActB,RELmtx(ActA,:)))
             Student(PosA)=ActB;
             Student(PosB)=ActA;
         end
     end
end
```

% Boctor mutation

```
function[Student]=mutation(NumAct,Student,RELmtx)
mut_prob=0.01;
for i=2:NumAct-2
    r=rand;
    if r< mut_prob
        PosA=i;
        PosB=randi([i+1,NumAct-1]);
        ActA=Student(PosA);
        ActB=Student(PosB);
        if not(ismember(ActB,RELmtx(ActA,:)))
            Student(PosA)=ActB;
            Student(PosB)=ActA;
        end
    end
end
```

%%% Codes for creating the 'crossover' function in TLBO and GA %%%

% Codes for 1-point Crossover

```
function[StudentNew]=Crossover(Student1,Student2,NumAct)
u=randi((NumAct-1));
a=Student1(1:u);
b=zeros(1,NumAct);
for i=1:NumAct
     if not(ismember(Student2(1,i),a));
         b(1,i)=Student2(1,i);
     end
end
b=b(logical(b));
StudentNew=[a,b];
```

%% Codes for 2-point Crossover%%

```
function[StudentNew]=Crossover(Student1,Student2,NumAct)
u=sort(randperm(NumAct-2,2));
a=Student1(1:u(1));
b=zeros(1,NumAct);
for i=1:NumAct
    if not(ismember(Student2(1,i),a));
        b(1,i)=Student2(1,i);
    end
end
```

```
b=b(logical(b));
b=b(1:u(2)-u(1));
b=[a,b];

if size(b,2)==NumAct;
    StudentNew=b;
else
    c=zeros(1,NumAct);
    for i=1:NumAct
        if not (ismember(Student1(1,i),b))
            c(1,i)=Student1(1,i);
        end
    end
 c=c(logical(c));
 StudentNew=[b,c];
end
```

%% Codes for calculating the activity times as per different priority rules %%

```
function[EST,EFT,LST,LFT] = ActTimes(ACTmtx,RELmtx,PREDmtx)
NumAct=size(RELmtx,1);
% PREDmtx=zeros(NumAct,max(RELmtx(:,3))+1);

for i=4:size(RELmtx,2)
    for j=1:NumAct
         if not(isnan(RELmtx(j,i)))
             PREDmtx(RELmtx(j,i),1)=PREDmtx(RELmtx(j,i),1)+1;
             PREDmtx(RELmtx(j,i),PREDmtx(RELmtx(j,i),1)+1)=RELmtx(j,1);
         end
     end
end
Npre=PREDmtx(1:NumAct,1);
Nsucc=RELmtx(1:NumAct,3);

% Codes to determine EST and EFT of the activities %
EST=zeros(NumAct,1);
EFT=zeros(NumAct,1);

for j=1:NumAct

    if Npre(j,1)==0
        EST(j,1)=0;
    else
        PredAct =[];
        L1=[];
        T1=[];
        PredAct=PREDmtx(j,2:Npre(j)+1);
        L1=length(PredAct);
        for k=1:L1
            T1(k,1)= EFT(PredAct(1,k),1);
        end
        EST(j,1)=max(T1);
    end
    EFT(j,1)=EST(j,1)+ ACTmtx(j,3);
```

```
end


% Codes to determine LFT and LST of the activities %
LFT=zeros(NumAct,1);
for m=NumAct:-1:1
    if  Nsucc(m)==0
        LFT(m)=EFT(NumAct,1);
    else
        SuccAct=[];
        L2=[];
        T2=[];
        SuccAct=RELmtx(m,4:Nsucc(m)+3);
        L2=length(SuccAct);
        for k=1:L2
            T2(k,1)=LFT(SuccAct(1,k),1)-ACTmtx(SuccAct(1,k),3);
        end
        LFT(m,1)=min(T2);
    end
    LST(m,1)=LFT(m,1)-ACTmtx(m,3);
end
```

% %% Codes to create Initial population as precedence feasible Activity List with Regret Based Biased Random Sampling (RBRS),Refer: Kolish & Hartmnn (1999) paper %%

```
function[Student]=Initial_Pop(NumAct,ACTmtx,SUCCmtx,PREDmtx)
[~,~,~,LFT] = ActTimes_Multi_Skill(ACTmtx,SUCCmtx,PREDmtx);
TIMES=[EST EFT LST LFT];
Student=zeros(1,NumAct);
        AL=[];       % Initialization of Activity List which is precedence feasible %
        DecSet=[];    % Decision Set of eligible activities at any stage %
        stage=1;      % Maximum stages is equal to number of activities i.e. NumAct %
while stage<= NumAct
    if stage==1
        AL=1;
        n2=[];
    else
        SuccAct=[];    % Successor activities at any stage %
        PredAct=[];    % Predecessor activities at any stage %
        NumSucc=[];    % Number of successors at any stage %
        PRIORITY_VALUE=[];
        BIASED_VALUE=[];
        BIASED_PROB=[];
        CUMM_PROB=[];
        ActSelect=[];
        n1=length(AL);
        NumSucc=SUCCmtx(AL(1,n1),2);
        SuccAct=SUCCmtx(AL(1,n1),3:(NumSucc+2));
        n2=[];
        for i=1:NumSucc
            PredAct=PREDmtx(SuccAct(1,i),3:size(PREDmtx,2));
            if  ismember(PredAct,[AL 0]);
                DecSet=[DecSet,SuccAct(1,i)];
```

```
            else
                DecSet=[DecSet];
            end
        end
           n2=length(DecSet);
        for j=1:n2
            PRIORITY_VALUE(1,j)=LFT(DecSet(1,j),1);
        end
        MAX=max(PRIORITY_VALUE);
        for j=1:n2
            BIASED_VALUE(1,j)=MAX-PRIORITY_VALUE(1,j)+1;  % taking alpha and eta as 1
                                                           from Hartmann(1999) paper
        end
        SUM=sum(BIASED_VALUE);
        for j=1:n2
            BIASED_PROB(1,j)=BIASED_VALUE(1,j)/SUM;
        end
        for j=1:n2
            if j==1
                CUMM_PROB(1,j)=BIASED_PROB(1,j);
            else
                CUMM_PROB(1,j)=BIASED_PROB(1,j)+ CUMM_PROB(1,j-1);
            end
        end
        r=rand;
        for j=1:n2
                    if r < CUMM_PROB(1,j)
                 ActSelect=[ActSelect DecSet(1,j)];
            end
        end
        n3=length(ActSelect);
        ActSelect=ActSelect(1,1);
        AL=[AL ActSelect];
        DecSet(n2-n3+1)=[ ];
        n2=[ ];
    end
    stage=stage+1;
end
Student(1,:)=AL;
end
```

% These codes have been developed to generate a MSRCPSP instance based on the methodology proposed in Almeida et al. (2015) %

```
clc
clear all
N=22;   % Enter the total number of activities including dummy source and sink
activities
NC=2.1; % Enter the desired network complexity

nStart=3;  % Number of activities having dummy source as predecessor
nFinish=3; % Number of activities having dummy sink as successor

MaxSucc=3; % Number of maximum successors allowable for any activity
```

```
MaxPred=3; % Number of maximum predecessors allowable for any activity

PRED_Matrix=zeros(N+1,N+2); % Initializing precedence matrix
SUCC_Matrix=zeros(N+1,N+2); % Initializing successor matrix

PRED_Matrix(1,3:N+2)=1:N; % Assigning index of activities in rows
PRED_Matrix(2:N+1,1)=1:N; % Assigning index of activities in columns

SUCC_Matrix(1,3:N+2)=1:N;  % Assigning index of activities in rows
SUCC_Matrix(2:N+1,1)=1:N;  % Assigning index of activities in columns

PRED_Matrix(3:nStart+2,3)=ones(nStart,1); % Updating PRED_Matrix by nstart and
nfinish
PRED_Matrix(N+1,N+2-nFinish:N+1)=ones(1,nFinish);

SUCC_Matrix(2,4:nStart+3)=ones(1,nStart); % Updating SUCC_Matrix by nstart and
nfinish
SUCC_Matrix(N-nFinish+1:N,N+2)=ones(nFinish,1);


SUCC_Matrix2=SUCC_Matrix(2:N+1,3:N+2);  % Selecting a sub-set matrix from main matrix
having logical 0 or 1 representing successors.
SUM_Succ=sum(SUCC_Matrix2,2);   % Calculating row-wise summation of elements of
matrix.
Num_Succ=[(1:N)' SUM_Succ];  % Calculating number of successors of each activity till
this stage.

PRED_Matrix2=PRED_Matrix(2:N+1,3:N+2); % Selecting a sub-set matrix from main matrix
having logical 0 or 1 representing predecessors.
SUM_Pred=sum(PRED_Matrix2,2);   % Calculating row-wise summation of elements of
matrix.
Num_Pred=[(1:N)' SUM_Pred]; % Calculating number of predecessors of each activity
till this stage.
```

% Now follows codes to randomly assign a predecessor to an activity with NIL predecessor based on ALGORITHM-1(Almeida et al., 2015)

```
nonredarcs=nStart+nFinish;
j=nStart+2;
LIMIT1=1;
while LIMIT1<100 && j<N
    while Num_Pred(j,2)==0
        if j>=(N-nFinish)
            i=randi([2,N-nFinish-1],1);
        else
            i=randi([2,j-1],1);
        end

        if Num_Succ(i,2)< MaxSucc

            SUCC_Matrix2(i,j)=1;
            SUM_Succ=sum(SUCC_Matrix2,2);
            Num_Succ=[(1:N)' SUM_Succ];

            PRED_Matrix2(j,i)=1;
            SUM_Pred=sum(PRED_Matrix2,2);
```

```
            Num_Pred=[(1:N)' SUM_Pred];
            nonredarcs=nonredarcs+1;
        end
    end
    j=j+1;
    LIMIT1=LIMIT1+1;
end

SUCC_MATRIX_ALL=zeros(N,3);
for i=1:N
    idx1 = find(SUCC_Matrix2(i,:));
    SUCC_MATRIX_ALL(i,1:length(idx1)) = idx1;
end
SUCC_MATRIX_ALL=[Num_Succ SUCC_MATRIX_ALL];

PRED_MATRIX_ALL=zeros(N,3);
for i=1:N
    idx2 = find(PRED_Matrix2(i,:));
    PRED_MATRIX_ALL(i,1:length(idx2)) = idx2;
end
PRED_MATRIX_ALL=[Num_Pred PRED_MATRIX_ALL]
nonredarcs ;
```

% % Now follows codes to randomly assign a successor to an activity with NIL successor based on ALGORITHM-2 (Almeida et al., 2015)

```
j=N-nFinish-1;  % Initialization of j
LIMIT2=1;
while LIMIT2<100 && j>1
    PRED_IMM=PRED_MATRIX_ALL(j,3:5); % PRED_IMM gives immediate predecessors of j
    PRED_ALL=[];
    PRED1_ALL=[];
    PRED2_ALL=[];
    PRED3_ALL=[];
    PRED4_ALL=[];
    for k1=1:length(PRED_IMM)
        if PRED_IMM(k1)>1
            PRED1_ALL(k1,:)=PRED_MATRIX_ALL(PRED_IMM(k1),3:5);
            for k2=1:length(PRED1_ALL)
                if PRED1_ALL(k2)>1
                    PRED2_ALL(k2,:)=PRED_MATRIX_ALL(PRED1_ALL(k2),3:5);
                    for k3=1:length(PRED2_ALL)
                        if PRED2_ALL(k3)>1
                            PRED3_ALL(k3,:)=PRED_MATRIX_ALL(PRED2_ALL(k3),3:5);
                            for k4=1:length(PRED3_ALL)
                                if PRED3_ALL(k4)>1

PRED4_ALL(k4,:)=PRED_MATRIX_ALL(PRED3_ALL(k4),3:5);
                                end
                            end
                        end
                    end
                end
```

```
            end
        end
    end
  PRED_ALL=[PRED_IMM ;PRED1_ALL;PRED2_ALL;PRED3_ALL;PRED4_ALL];
    Z=unique(PRED_ALL)';
    Z(Z==0)=[];
    PRED_ALL=Z; % Gives set of all immediate and distant predecessors for activity j
    SUCC_OF_PREDS=[];
    A=[];
    for i=1:length(PRED_ALL)
        A=SUCC_MATRIX_ALL(PRED_ALL(i),3:5);
        SUCC_OF_PREDS=[A SUCC_OF_PREDS];
    end
    SUCC_OF_PREDS;
    Z1=unique(SUCC_OF_PREDS);
    Z1(Z1==0)=[];
    SUCC_OF_PREDS=Z1;

    while Num_Succ(j,2)==0
        if j<=nStart+1
            u=randi([nStart+2,N-1],1);
        else
            u=randi([j+1,N-1],1);
        end

        if Num_Pred(u,2)< MaxPred && ~(ismember(u,SUCC_OF_PREDS))

            SUCC_Matrix2(j,u)=1;
            SUM_Succ=sum(SUCC_Matrix2,2);
            Num_Succ=[(1:N)' SUM_Succ];

            PRED_Matrix2(u,j)=1;
            SUM_Pred=sum(PRED_Matrix2,2);
            Num_Pred=[(1:N)' SUM_Pred];
            nonredarcs=nonredarcs+1;
        end
    end
    j=j-1;
    LIMIT2=LIMIT2+1;
end
SUCC_Matrix2;
SUCC_MATRIX_ALL=zeros(N,3);
for i=1:N
    idx3 = find(SUCC_Matrix2(i,:));
    SUCC_MATRIX_ALL(i,1:length(idx3)) = idx3;
end
SUCC_MATRIX_ALL=[Num_Succ SUCC_MATRIX_ALL];

PRED_Matrix2;
PRED_MATRIX_ALL=zeros(N,3);
for i=1:N
    idx4 = find(PRED_Matrix2(i,:));
    PRED_MATRIX_ALL(i,1:length(idx4)) = idx4;
end
PRED_MATRIX_ALL=[Num_Pred PRED_MATRIX_ALL];
```

%%% Now follows codes to achieve the graph with desired network complexity based on ALGORITHM-2 (Almeida et al., 2015) %%%

```
reqnumarcs=ceil(NC*N)
LIMIT3=1;
while   LIMIT3<100 && nonredarcs<= reqnumarcs
    nonredarcs;
    i=randi([2,N-nFinish-1],1);
    if Num_Succ(i,2)< MaxSucc
        if i<=nStart+1
            S1=nStart+2:N-1;
            S2=SUCC_MATRIX_ALL(i,3:5);
            S3=setdiff(S1,S2);
            j=S3(randi([1,length(S3)],1));
        else
            S1=i+1:N-1;
            S2=SUCC_MATRIX_ALL(i,3:5);
            S3=setdiff(S1,S2);
            j=S3(randi([1,length(S3)],1));
        end

        if Num_Pred(j,2)< MaxPred


% We now compute all predecessors of activity i (represented as PRED_ALL then Z2)

             PRED_IMM=[];

        PRED_IMM=PRED_MATRIX_ALL(i,3:5); % PRED_IMM gives immediate predecessors
                      of i chosen above
            PRED_ALL=[];
            PRED1_ALL=[];
            PRED2_ALL=[];
            PRED3_ALL=[];
            PRED4_ALL=[];
            PRED5_ALL=[];
            for k11=1:length(PRED_IMM)
                if PRED_IMM(k11)>1
                    PRED1_ALL(k11,:)=PRED_MATRIX_ALL(PRED_IMM(k11),3:5);
                    for k12=1:length(PRED1_ALL)
                        if PRED1_ALL(k12)>1
                            PRED2_ALL(k12,:)=PRED_MATRIX_ALL(PRED1_ALL(k12),3:5);
                            for k13=1:length(PRED2_ALL)
                                if PRED2_ALL(k13)>1

PRED3_ALL(k13,:)=PRED_MATRIX_ALL(PRED2_ALL(k13),3:5);
                                        for k14=1:length(PRED3_ALL)
                                            if PRED3_ALL(k14)>1

PRED4_ALL(k14,:)=PRED_MATRIX_ALL(PRED3_ALL(k14),3:5);
                                                    for k15=1:length(PRED4_ALL)
                                                        if PRED4_ALL(k15)>1
```

```
PRED5_ALL(k15,:)=PRED_MATRIX_ALL(PRED4_ALL(k15),3:5);
                                                    end
                                                end
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
            PRED_ALL=[PRED_IMM ;PRED1_ALL;PRED2_ALL;PRED3_ALL;PRED4_ALL;PRED5_ALL];
            Z2=unique(PRED_ALL)';
            Z2(Z2==0)=[];
            PRED_ALL=Z2;
```

%%% This function 'SSS_Multi_Skill' applies modified Serial Schedule Generation Scheme (SSS) to determine the makespan for a given Student %%%

```
 function[Makespan]=SSS_Multi_Skill(NumAct,Student,Staff_Act,PREDmtx,ACTmtx,SUCCmtx)

[EST,EFT] = ActTimes_Multi_Skill(ACTmtx,SUCCmtx,PREDmtx);
STmtx=zeros(NumAct,1);  % Initializing the start times of all activities
FTmtx=zeros(NumAct,1);  % Initializing the finish times of all activities
Npre=PREDmtx(:,2);
SCH_ACT=[];  % initializes the set of already scheduled activities
for i=2:NumAct-1

    if  i==2
        DEC_ACT=Student(1,i);
        STmtx(DEC_ACT)=EST(DEC_ACT);
        FTmtx(DEC_ACT)=STmtx(DEC_ACT)+ACTmtx(DEC_ACT,6);
        [STmtx FTmtx];
        SCH_ACT=[DEC_ACT];
        for j=1:SUCCmtx(DEC_ACT,2)
EST(SUCCmtx(DEC_ACT,j+2))=max(EST(SUCCmtx(DEC_ACT,j+2)),FTmtx(DEC_ACT));

EFT(SUCCmtx(DEC_ACT,j+2))=EST(SUCCmtx(DEC_ACT,j+2))+ACTmtx(SUCCmtx(DEC_ACT,j+2),6);
        end

    else
        DEC_ACT=[];        % empties the DEC_ACT to include the new current activity
                           to be scheduled
        PRED_DEC_ACT=[];   % empties the set of predecessor activities of current
                           activity to be scheduled
        PARRL_ACT=[];      % empties the set of parallel activities to be considered
                           while scheduling current activity
        DEC_ACT=Student(1,i);      % defines the new current activity to be scheduled
        PRED_DEC_ACT=PREDmtx(DEC_ACT,3:5); % defines the predecessor activities of
                                              current activity to be scheduled
```

```
        PARRL_ACT=setdiff(SCH_ACT,PRED_DEC_ACT); % defines the set of activities with
                                                 which current activity in hand 'can be'
                                                 scheduled in parallel
        if isempty(PARRL_ACT)
            EST(DEC_ACT)=max(FTmtx);
            STmtx(DEC_ACT)=EST(DEC_ACT);
            EFT(DEC_ACT)=EST(DEC_ACT)+ACTmtx(DEC_ACT,6);
            FTmtx(DEC_ACT)=EFT(DEC_ACT);

            for j=1:SUCCmtx(DEC_ACT,2)

             EST(SUCCmtx(DEC_ACT,j+2))=max(EST(SUCCmtx(DEC_ACT,j+2)),FTmtx(DEC_ACT));

             EFT(SUCCmtx(DEC_ACT,j+2))=EST(SUCCmtx(DEC_ACT,j+2))+ACTmtx(SUCCmtx(DEC_A
             CT,j+2),6);
            end
            SCH_ACT=[SCH_ACT DEC_ACT];
        else
            for k=1:length(PARRL_ACT)

                if (STmtx(PARRL_ACT(k))<=EST(DEC_ACT)&&
       EST(DEC_ACT)<FTmtx(PARRL_ACT(k)))  ||  (STmtx(PARRL_ACT(k))<EFT(DEC_ACT)&&
       EFT(DEC_ACT) <=FTmtx(PARRL_ACT(k)));

                    % we now check if there is(are) any common worker(s) between any
                    activity from set PARRL_ACT and activity in DEC_ACT
                    Staff_set1=[];
                    Staff_set2=[];
                    Staff_set1=Staff_Act(PARRL_ACT(k),2:13);
                    Staff_set1=Staff_set1(logical(Staff_set1));
                    Staff_set2=Staff_Act(DEC_ACT,2:13);
                    Staff_set2=Staff_set2(logical(Staff_set2));

                    if any(intersect(Staff_set1,Staff_set2))==1;
                        EST(DEC_ACT)=FTmtx(PARRL_ACT(k));
                        STmtx(DEC_ACT)=EST(DEC_ACT);
                        EFT(DEC_ACT)=EST(DEC_ACT)+ ACTmtx(DEC_ACT,6);
                        FTmtx(DEC_ACT)=EFT(DEC_ACT);
                    else
                        STmtx(DEC_ACT)=EST(DEC_ACT);
                        FTmtx(DEC_ACT)=STmtx(DEC_ACT)+ ACTmtx(DEC_ACT,6);
                    end
                        SCH_ACT=[SCH_ACT DEC_ACT];
                        for j=1:SUCCmtx(DEC_ACT,2)

EST(SUCCmtx(DEC_ACT,j+2))=max(EST(SUCCmtx(DEC_ACT,j+2)),FTmtx(DEC_ACT));

EFT(SUCCmtx(DEC_ACT,j+2))=EST(SUCCmtx(DEC_ACT,j+2))+ACTmtx(SUCCmtx(DEC_ACT,j+2),6);
 end
 else

CONDITION=(EFT(DEC_ACT)<=STmtx(PARRL_ACT(k))||EST(DEC_ACT)>=FTmtx(PARRL_ACT(k)));
                    STmtx(DEC_ACT)=EST(DEC_ACT);
                    FTmtx(DEC_ACT)=STmtx(DEC_ACT)+ ACTmtx(DEC_ACT,6);
                    SCH_ACT=[SCH_ACT DEC_ACT];
```

```
                    for j=1:SUCCmtx(DEC_ACT,2)

EST(SUCCmtx(DEC_ACT,j+2))=max(EST(SUCCmtx(DEC_ACT,j+2)),FTmtx(DEC_ACT));

EFT(SUCCmtx(DEC_ACT,j+2))=EST(SUCCmtx(DEC_ACT,j+2))+ACTmtx(SUCCmtx(DEC_ACT,j+2),6);
                    end
                end
            end
        end
    end
end
STmtx(NumAct)=EST(NumAct);
FTmtx(NumAct)=EFT(NumAct);
[STmtx FTmtx];
% SCHEDULE=[(1:NumAct)' EST EFT];
Makespan=max(EFT);
```

%% Code to generate Staff-Skill matrix

```
clc
clear all
P=11;  % Define the number of staff members
r1=randi([1,3],P,1); % Number of skills mastered by each person(staff) is varied in
set {1,2,3}
for i=1:P
    if r1(i)==1
        P1=[1 0 0 0];
    elseif r1(i)==2
        P1=[1 1 0 0];
    elseif r1(i)==3
        P1=[1 1 1 0];
    end
  STAFF_SKILL_MATRIX(i,:)=P1(randperm(4));
end
STAFF_SKILL_MATRIX
```

% Code to generate Activity-Skill Matrix for a given skill factor

```
clc
clear all
N=30; % Define the number of activities


%If SF is 0.5, 0.75 or 1.0 use this code
A=[1 1 0 0];% SF=0.50, so two 'ones' are included in A1; total skill types is 4
%A=[1 1 1 0];% SF=0.75, so three 'ones' are included in A2; total skill types is 4
%A=[1 1 1 1];% SF=1.0, so four 'ones' are included in A4; total skill types is 4
for i=1:N % where N is number of activities
    B(i,:)=A(randperm(4));
end
```

```
% If SF is variable use this code
%SF variable means number of skill types needed by an activity is randomly chosen in
set {2,3,4}.
r2=randi([2,4],N,1); % Since there are three possibilities in {2,3,4} i.e. number of
types of skills needed by each activity
for i=1:N
if r2(i)==2
         A=[1 1 0 0];
     elseif r2(i)==3
        A=[1 1 1 0];
     elseif r2(i)==4
        A=[1 1 1 1];
     end
   B(i,:)=A(randperm(4));
end
C=randi([1,3],N,4); % No. of persons required by each activity is varied in set
{1,2,3}
ACT_SKILL_MATRIX=B.*C
P=randi([1,10],N,1);
```

% %% Now follows codes to assign feasible set of staff members to meet activities' skill requirements%%

```
function[Staff_Act]=Staff_Assignment(NumAct,Student,ACTmtx,STAFFmtx)

STAFF_SKILL1=find(STAFFmtx(:,2)==1)';
STAFF_SKILL2=find(STAFFmtx(:,3)==1)';
STAFF_SKILL3=find(STAFFmtx(:,4)==1)';
STAFF_SKILL4=find(STAFFmtx(:,5)==1)';

    Staff_Act=zeros(NumAct,12);
    for j=2:NumAct-1
        b=[];
        S1=[];
        S2=[];
        S3=[];
        S4=[];
        A=[];
        for k=1:4
            b=ACTmtx(Student(1,j),k+1);
            if b==0
                A=[];
            elseif b~=0 && k==1
                r1=randperm(length(STAFF_SKILL1),b);
                for n1=1:b
                    S1(n1)=STAFF_SKILL1(r1(n1));
                end


            elseif b~=0 && k==2
                r2=randperm(length(STAFF_SKILL2),b);
```

```
            for n2=1:b
                S2(n2)=STAFF_SKILL2(r2(n2));
            end


        elseif b~=0 && k==3
            r3=randperm(length(STAFF_SKILL3),b);
            for n3=1:b
                S3(n3)=STAFF_SKILL3(r3(n3));
            end

        elseif b~=0 && k==4
            r4=randperm(length(STAFF_SKILL4),b);
            for n4=1:b
                S4(n4)=STAFF_SKILL4(r4(n4));
            end
        end

    end
    A=[S1 S2 S3 S4];
```

% We now apply while loop to ensure that for each activity no staff member is assigned to perform two or more skills simultaneously or in other words staff assignment is unique in nature.%

```
    LIMIT=1;
    while length(unique(A))~=length(A)&& LIMIT<1000
        b=[];
        S1=[];
        S2=[];
        S3=[];
        S4=[];
        A=[];
        for k=1:4
            b=ACTmtx(Student(1,j),k+1);
            if b==0
                A=[];
            elseif b~=0 && k==1
                r1=randperm(length(STAFF_SKILL1),b);
                for n1=1:b
                    S1(n1)=STAFF_SKILL1(r1(n1));
                end


            elseif b~=0 && k==2
                r2=randperm(length(STAFF_SKILL2),b);
                for n2=1:b
                    S2(n2)=STAFF_SKILL2(r2(n2));
                end

            elseif b~=0 && k==3
                r3=randperm(length(STAFF_SKILL3),b);
                for n3=1:b
```

```
                    S3(n3)=STAFF_SKILL3(r3(n3));
                end

            elseif b~=0 && k==4
                r4=randperm(length(STAFF_SKILL4),b);
                for n4=1:b
                    S4(n4)=STAFF_SKILL4(r4(n4));
                end
            end

        end

        LIMIT=LIMIT+1;
        A=[S1 S2 S3 S4];
    end
         Staff_Act(j,1:length(A))=A;
end

    Staff_Act=[Student(1,:)' Staff_Act];
for i=1:NumAct
    MINIMUM=min(Staff_Act(:,1));
    index=find((MINIMUM==Staff_Act(:,1)));
    Staff_Act2(i,:)=Staff_Act(index,:);
    Staff_Act(index,:)=[];
end
Staff_Act=Staff_Act2;
```

% Now follows codes for the MO-MSRCPSP to calculate the TOTAL TIME ELAPSED IN UNDER-SKILLED STAFF ASSIGNMENT (denoted as TIME UnderSkill) for the project

```
clc
clear all
ACTmtx=textread('ACTmtx.txt');
STAFFmtx=textread('STAFFmtx2.txt');
Staff_Act=textread('Staff_Act.txt');
NumAct=12;

TIME_UnderSkill_Matrix=zeros(NumAct,4);
for i=1:NumAct
    b=[];
    ASSIGNED_STAFF=0;
    for      k=1:4  % as there are a total of four skill types
        b=ACTmtx(i,k+1);
        if  b~=0
            Staff_Index=[];
            TIME_UnderSkill_Array=[];
            for n=1:b
                Staff_Index(n)=Staff_Act(i,n+1+ASSIGNED_STAFF);
                TIME_UnderSkill_Array(n)=(1-
(STAFFmtx(Staff_Index(n),k+1)))*ACTmtx(i,6);
            end
            ASSIGNED_STAFF=ASSIGNED_STAFF+b;
```

```
            TIME_UnderSkill_Matrix(i,k)=sum(TIME_UnderSkill_Array);
        end
    end
end
TIME_UnderSkill_Matrix
TIME_UnderSkill=sum(sum(TIME_UnderSkill_Matrix))

**************************************************************************
```

List of Publications

International Journals

1. **Joshi, D**., Mittal, M.L., Kumar M. An efficient teaching-learning-based optimization algorithm (TLBO) for the resource-constrained project scheduling problem, *International Journal of Industrial and Systems Engineering (Accepted)* (SCOPUS Indexed)
2. **Joshi, D**., Mittal, M.L., Sharma M. K., Kumar M. An effective teaching-learning-based optimization algorithm for the multi-skill resource-constrained project scheduling problem, *Journal of Modelling in Management (Accepted) (*SCOPUS Indexed)
3. **Joshi, D**., Mittal, M.L., Kumar M. A teaching-learning-based optimisation algorithm for the multi-objective multi-skill resource-constrained project scheduling problem. (*To be communicated*)
4. **Joshi, D**., Mittal, M.L., Kumar M. Resource flexibility in project scheduling: Facets and dimensions. (*To be communicated*)

International Conferences

1. **Joshi, D.**, Mittal, M.L., Kumar M., An improved teaching-learning-based algorithm (TLBO) for the resource-constrained project scheduling problem (RCPSP), In: *XX Annual International Conference of Society of Operations Management- 2016* held at ABV-IIITM Gwalior from 22nd to 24th December, 2016.

2. **Joshi, D**., Mittal, M. L., & Kumar, M. (2019). A Teaching–Learning-Based Optimization Algorithm for the Resource-Constrained Project Scheduling Problem. In *Harmony Search and Nature Inspired Optimization Algorithms* (pp. 1101-1109). Springer, Singapore.(SCOPUS Indexed)

Author's Biographical Sketch

Name	:	Dheeraj Joshi
Date of Birth	:	July 25, 1979
Address	:	Department of Mechanical Engineering Swami Keshvanand Institute of Technology, Management & Gramotham, Jaipur-302017. E-mail: dheerajjoshi25@gmail.com Phone: 9460355422
Qualifications	:	B.E. (Production and Industrial Engineering) (2002). M.B.M. Engineering College, Jai Narain Vyas University,Jodhpur. M.E. (Production and Industrial Engineering) (2011). M.B.M. Engineering College, Jodhpur. Jai Narain Vyas University,Jodhpur.
Experience	:	Associate Professor, Department of Mechanical Engineering Swami Keshvanand Institute of Technology, Management & Gramotham, Jaipur-302017 (July, 2005- till now)
Areas of Interest	:	Engineering Optimization Operations Management Project Management and Scheduling Artificial Intelligence Techniques Manufacturing Science