

Solving Multi-Objective Assignment Problem with Decision Maker's Preferences by Using Genetic Algorithm

By

Md. Mahbubur Rahman

A thesis report is submitted in partial fulfilment of the requirements for the degree of
Master of Science in Mechanical Engineering.



Khulna University of Engineering & Technology

Khulna 9203, Bangladesh

June 2019

DECLARATION

This is to certify that the work entitled “Solving Multi-Objective Assignment Problem with Decision Maker’s Preferences by Using Genetic Algorithm” has been carried out by ‘Md. Mahbubur Rahman’ in the Department of Mechanical Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh. The above thesis work or any part of this work has not been submitted anywhere for the award of any degree or diploma.



Dr. Md. Kutub Uddin

Professor

Department of Mechanical Engineering

Khulna University of Engineering & Technology




Md. Mahbubur Rahman

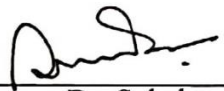
Roll No. 1505506


APPROVAL


This is to certify that the thesis work submitted by ‘Md. Mahbubur Rahman’ entitled “Solving Multi-Objective Assignment Problem with Decision Maker’s Preferences by Using Genetic Algorithm” has been approved by the Board of Examiners for the partial fulfilment of the requirements for the degree of ‘Master of Science in Mechanical Engineering’ in the Department of Mechanical Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh in June 2019.


BOARD OF EXAMINERS

1. 

Professor Dr. Md. Kutub Uddin
Department of Mechanical Engineering, KUET, Khulna
Chairman
(Supervisor)
2. 

Professor Dr. Sobahan Mia
Head, Department of Mechanical Engineering, KUET, Khulna
Member
3. 

Dr. Abdullah Al-Faruk
Associate Professor
Department of Mechanical Engineering, KUET, Khulna
Member
4. 

Subrata Talapatra
Associate Professor
Department of Industrial Engineering and Management, KUET, Khulna
Member
5. 

Professor Dr. Md. Mosharraf Hossain
Department of Industrial and Production Engineering
RUET, Rajshahi
Member
(External)

ACKNOWLEDGEMENTS

First of all the author would like to convey his gratitude to the Almighty ALLAH. Without his blessings, this activity would never be a success. The author would like to thank all the people who contributed in some way to the work described in this thesis. The author thanks supervisor Dr. Md. Kutub Uddin, Professor, Department of Mechanical Engineering (ME), Khulna University of Engineering & Technology (KUET), Khulna, whose active guidance through the thesis period enabled the author to complete it. Without his valuable suggestions and generous support from the beginning of the thesis, completion of this paper was hardly possible. During the tenure, he imparts me the intellectual freedom in my work, supporting my attendance at various conferences, engaging me in new ideas, and demanding a high quality of work in all my endeavours. Author also extend his gratitude to Dr. Sobahan Mia, Professor & Head, ME, KUET, for giving the opportunity of taking a part of such job. The author is grateful to all the teachers and staffs of the ME, KUET, Khulna, for their cordial cooperation during the course.

The author would like to thank all the committee members in the board of examiners, especially Dr. Md. Mosharraf Hossain, Professor, Department of Industrial and Production Engineering, Rajshahi University of Engineering & Technology (RUET), Rajshahi, for his interest in the given work. The author also pays homage to Prof. Dr. Quazi Sazzad Hossain, Vice-Chancellor, KUET, for the financial and non-financial support to complete the thesis.

June 2019

Md. Mahbubur Rahman

ABSTRACT

The multi-objective assignment problem is basically the N men – N tasks problem, where a single task has to be assigned to an individual with a view of optimizing the outcomes. A common challenge is to address the conflicting objectives which produce Pareto-optimal solutions. The main feature of the work is- normalizing all the criteria into a single scale regardless of their measurement units and their demand of minimum or maximum, which reliefs the researchers from careful attention in quantifying the quality criteria. The methodology also included the decision maker's preferences regarding the objectives. While solving the problem through a genetic algorithm, a new encoding scheme is used together with a partially matched crossover (PMX). The working principle of the proposed algorithm is illustrated with a numerical example and its effectiveness has been compared with some well-established methodologies. It is found that the proposed algorithm provides a better solution with minimal computational effort.

Keywords: Assignment problem, Multi-objective, Decision maker's preferences, and Genetic Algorithm.

CONTENTS

Content	Page
TITLE PAGE.....	i
DECLARATION.....	ii
APPROVAL.....	iii
ACKNOWLEDGEMENTS.....	iv
ABSTRACT.....	v
CONTENTS.....	vi
LIST OF FIGURES.....	viii
LIST OF TABLES.....	ix
CHAPTER I INTRODUCTION.....	1
1.1 General Introduction.....	1
1.2 Objectives of the Study.....	2
1.3 Organization of Thesis.....	2
CHAPTER II LITERATURE REVIEW.....	3
2.1 Literature Review.....	3
2.2 Summary of the Literature Review.....	7
CHAPTER III THEORETICAL CONSIDERATION.....	8
3.1 Types of Objective Criteria.....	8
3.2 Transformation of Multi-Objective to a Single Objective.....	8
3.2.1 Case One.....	9
3.2.2 Case Two.....	10
3.2.3 Case Three.....	14
3.2.4 Case Four.....	16
CHAPTER IV DEVELOPMENT OF MODEL AND SOLUTION METHODOLOGY.....	20
4.1 The Model.....	20
4.1.1 Notation.....	20
4.1.2 Mathematical Model.....	21
4.2 The Genetic Algorithm for the MOAP.....	23
4.2.1 Basis of the Formulation.....	23
4.2.2 The Details of the Proposed Algorithm.....	24
4.3 Numerical Illustration and Computational Experience.....	29
4.3.1 Performance Study.....	29

4.3.2	Comparisons of the Proposed Approach with Other Methodologies	31
CHAPTER V RESULTS & DISCUSSIONS		37
5.1	Results.....	37
5.2	Discussions.....	39
CHAPTER VI CONCLUSIONS		41
6.1	Conclusions	41
6.2	Limitation.....	41
6.3	Future Possible Works	42
REFERENCES		43
APPENDIX A		46

LIST OF FIGURES

Figure No.	The caption of the Figures	Page
Figure 4.1:	Convergence to the global maximum fitness	30
Figure 4.2:	Sensitivity of total cost, total time and total quality to their relative weightage	31
Figure 5.1:	Results in two methodologies for three examples	37
Figure 5.2:	Improvement by the proposed methodology for each criterion in equal priority	38

LIST OF TABLES

Table No.	The caption of the Tables	Page
Table 3.1:	A three objective assignment problem	9
Table 3.2:	Assigning negative sign before the value of quality	9
Table 3.3:	Translated single objective problem.....	10
Table 3.4:	Objective rating factor determination	11
Table 3.5:	Determination of relative importance of different Subjective factors	12
Table 3.6:	Determination of relative importance of each site for the skill of the worker	12
Table 3.7:	Determination of relative importance of each site for customer proximity	13
Table 3.8:	Determination of relative importance of each site for community attitude	13
Table 3.9:	Summary of subjective factors evaluation	13
Table 3.10:	A three-objective assignment problem	14
Table 3.11:	The three-objective assignment problem with the quantified quality.....	15
Table 3.12:	The normalized operation cost, time and quality	15
Table 3.13:	A three-objective assignment problem	16
Table 3.14:	Solution of the assignment problem	16
Table 3.15:	Performance data of four alternatives	17
Table 3.16:	Normalized rating.....	17
Table 3.17:	Weight of each criterion (Depending on decision maker)	17
Table 3.18:	Weighted decision criteria	18
Table 3.19:	Ideal alternative and nadir alternative.....	18
Table 3.20:	Distance measure of each alternative.....	19
Table 3.21:	Ranking of alternatives	19
Table 4.1:	A MOAP.....	22
Table 4.2:	A numerical example of a MOAP [29].....	29
Table 4.3:	Effect of population size changing on different parameters	30
Table 4.4:	6 tasks-6 workers MOAP solution	32
Table 4.5:	10 tasks-10 workers three-objective assignment problem	33
Table 4.6:	10 tasks-10 worker MOAP solution	34
Table 4.7:	14 tasks-14 workers three-objective assignment problem	35
Table 4.8:	14 tasks-14 workers MOAP solution.....	36

CHAPTER I

INTRODUCTION

1.1 General Introduction

The assignment problem (AP) is one of the fundamental topics in combinatorial optimization in the branch of operation research. There is the vast use of assignment problem in production planning, transportation, telecommunication, VLSI design, economics etc. It deals with the allocation of the various resources to the various activities on one to one basis so that an optimal assignment is made in the best possible way.

“The best person for the task” is a cleverness description of assignment model. The situation is demonstrated by assigning workers of varying degrees of skill to tasks. The accomplishment of a task with a skilled worker costs less than one in which an operator is not skilful. The objectives of the assignment model are to determine the minimum – cost assignment of workers to tasks [1].

In the real arena, management has many objectives for workers allocation to tasks. We often come in close contact with an assignment problem, where, cost and time are jointly co-related [2]. Multi-objective assignment model usually considers time, cost, safety, quality etc. simultaneously. Single objective optimization is easy to solve but the multi-objective problem is complex because of the conflicting nature of the objectives. These problems give rise to a set of trade-off among optimal solutions, popularly known as Pareto-optimal solutions. For the multi-objective assignment model, all the criteria are not equally important. Generally, decision maker imparts priority ranking among the objectives, *e.g.*, time is less important than quality. By incorporating the decision maker’s preferences into the problem, the problem becomes hard to solve.

In computer science and operations research, a genetic algorithm (GA) is metaheuristic optimization algorithm inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Generally, the genetic algorithm is used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover, and selection [3]. The genetic algorithm searches the space of solutions by combining the best features of two good solutions into a single one [4]. Genetic algorithms do not guarantee always to provide the exact optimal solutions, but they will definitely help to find better optimal solutions as compared to other methodologies within less amount of time [5].

1.2 Objectives of the Study

The main focus of the present work is to develop an efficient solution methodology for the multi-objective assignment problem. The methodology will include the decision maker's preferences. The specific objectives of the thesis are given below:

- i. To develop a methodology based on a genetic algorithm (GA) to solve the multi-objective assignment problem (MOAP) with decision maker's preferences.
- ii. To make the algorithm efficient with easier computation.

1.3 Organization of Thesis

The organization of the thesis is as follows: Chapter 2 discusses the overview of related research about assignment problem, MOAP and genetic algorithm (GA). Chapter 3 discusses the theoretical consideration. Chapter 4 demonstrates the methodology of proposed research in details. Chapter 5 presents the results and discussion of the proposed model. Chapter 6 includes conclusions, limitation and future scopes of the intended work.

CHAPTER II

LITERATURE REVIEW

2.1 Literature Review

The linear assignment problem is a special type of linear programming problem where assignees are being assigned to perform tasks in one to one basis [6]. It addresses the question of how to set assignee to tasks in an injective way so that the assignment cost (or profit) is minimum (or maximum).

Kuhn [7] proposed an algorithm for the linear assignment problem known as the Hungarian method. It is primarily designed for hand computation [8]. Its running time complexity is $O(n^3)$ that is very tedious with a large number of the task [9]. The problem can also be formulated as an integer-programming model and solved by Branch-and-Bound technique. The Hungarian algorithm for solving the assignment model is more efficient than branch-and-bound algorithms [10]. Hungarian method deals with a single objective. Solving multiple objectives optimization problem is beyond the capacity of this method. This methods basically works with the quantity which is needed to be minimized such as cost. Nowadays it is often requirement handling the quantity which needs to be maximized such as quality.

Sahu and Thapadar [11] solved Single Assignment Problem with genetic algorithm and Simulated Annealing. They did an experimental investigation into solving the Assignment model using Genetic Algorithm and Simulated Annealing. Various parameters affecting the algorithms are studied and their influence on convergence to the final optimum solution is shown.

Kirubha [12] solved assignment problem under fuzzy environment. In his paper, the various aspects of fuzzy assignment problem and travelling salesman problem have been dealt with. The method of solving can be understood easily and applied for practice.

Further, the method can be slightly modified and used for solving other linear programming problem like Transportation problem.

The travelling salesman problem (TSP) was first proposed by Irish mathematician W.R. Hamilton in the 19th century. The travelling salesman problem (TSP) can be transformed into an assignment problem [13]. The goal of the problem is to find the shortest route of the salesman starting from a given city, visiting all other cities only once and finally coming back to the same city where he started. This is known as NP (non-deterministic polynomial time) problem in a combinatorial optimization. To solve this problem different researchers used heuristic, metaheuristic and optimal methods like dynamic programming, linear programming, branch and bound method, cutting plane algorithms, stimulated annealing and Markov chain.

Liu and Gao [14] proposed an equilibrium optimization problem and extended the assignment problem to the equilibrium multi-job assignment problem, and equilibrium multi-job quadratic assignment problem and used a genetic algorithm to solve the proposed models. In his paper, the equilibrium optimal problem is proposed and then some equilibrium problems such as the equilibrium multi-job assignment problem, the equilibrium multi-job quadratic assignment problem and the minimum costs and equilibrium multi-job assignment problem are studied. Furthermore, a genetic algorithm is designed for solving the proposed programming models and some numerical examples are given to verify the efficiency of the designed algorithm. The designed algorithm can be easily modified to solving the weighted minimum edge cover set problem of the complete bipartite graph.

Huang and Zhang [15] develop the method for fuzzy assignment problem with restriction of qualification. In that paper, they bring forward the person unequal to jobs assignment problem with restriction of qualification, make the mathematics model, and give the sufficient and necessary condition for judging the existence of solution; combine with the algorithm of traditional assignment problem, they give the method for assignment problem with restriction of qualification. As for the maximum case, the modelling and judging methods are the same, and the solution method is similar.

In the real arena, management has many objectives for tasks allocation to workers. We often come in close contact with an assignment problem, where, cost and time are jointly co-related [1]. Multi-objective assignment model usually considers time, cost, safety, quality etc. simultaneously. Single objective optimization is easy to solve but the multi-objective problem is complex because of the conflicting nature of the objectives. These problems give rise to a set of trade-off among optimal solutions, popularly known as Pareto-optimal solutions.

Pramanik and Biswas [1] solved the MOAP with Generalized Trapezoidal Fuzzy Numbers. It deals with the imprecise costs, time and effectiveness instead of its precise information. It also uses linear programming to find the solution. Euclidean distance is used for selecting a proper priority structure for obtaining a compromise optimal solution. The concept presented, in his paper, is illustrated with MOAP involving generalized trapezoidal fuzzy numbers to check the effectiveness of the proposed method.

Bao et al. [16] uses 0-1 programming to translate a MOAP into a linear programming problem. The provided method can handle non-quantification or quality situation. In the prerequisite that all tasks can be assigned effectively, the use of resources can be minimized and the assignment problem can be solved efficiently. Moreover, by adjusting the weight in each objective, the provided approach, in fact, is a general form to the original one-objective assignment problem. There may be multiple local optimum solutions. Linear programming has not the ability to avoid being trapped in local optimal solution as it starts searching from a single point and moves to nearby better solution point. This may lead to trapped in local optimal solution [17]. Since the demand for quality is maximal instead of minimal, the reciprocal of the quantified quality has been taken as the normalized quality.

Tsai et. al. [18] proposed a new methodology to solve the problem of multi-objective fuzzy deployment of manpower. They transform the multi-objective problem into a fuzzy linear programming. In this methodology, a careful attention must be paid to the determinations of the weights among the resources. The management emphasis may skew to an erroneous assignment for an inappropriate set of weights.

Geetha and Nair [19] shows that the assignment problem has two aspects, namely, the cost and the time. In that paper, an objective of minimizing the cost of assignment with an additional ‘supervisory’ cost, which depends on the total time of completion of the project is formulated and a method of finding an optimal solution for the problem is developed. They proposed that in the case of a transportation problem, a similar model can be developed and the algorithm can be modified easily to solve.

On the other hand, in computer science and operations research, genetic algorithm (GA) is a stochastic search and optimization technique inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA) [3]. GA is commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover, and selection [20]. GA is applicable for both constrained and unconstrained optimization problems [21]. GA has the ability to avoid being trapped in local optimal solution as it starts searching from multiple points and the mutation operator scatters solution points globally [22].

Deb and Kalyanmoy [23] solved multi-objective optimization using an evolutionary algorithm. He employed generative multiple-criteria decision-making (MCDM) methodology to solve multiple objectives optimization. He works with the fast-growing field of multi-objective optimization based on evolutionary algorithms. First, the principles of single-objective evolutionary optimization (EO) techniques have been discussed so that readers can visualize the differences between evolutionary optimization and classical optimization methods. The Evolutionary Multi-objective Optimization (EMO) principle of handling multi-objective optimization problems is to find a representative set of Pareto-optimal solutions. Since an EO uses a population of solutions in each iteration, EO procedures are potentially viable techniques to capture a number of trade-off near-optimal solutions in a single simulation run. This chapter has described a number of popular EMO methodologies, presented some simulation studies on test problems, and discussed how EMO principles can be useful in solving real-world multi-objective optimization problems through a case study of spacecraft trajectory optimization. Finally, his work has discussed the potential of EMO and its current research activities. The principle of EMO has been utilized to solve other optimization problems that are otherwise not multi-objective in nature. The diverse set of EMO

solutions have been analyzed to find hidden common properties that can act as valuable knowledge to a user. EMO procedures have been extended to enable them to handle various practicalities. Finally, the EMO task is now being suitably combined with decision-making activities in order to make the overall approach more useful in practice. EMO addresses an important and inevitable fact of problem-solving tasks. EMO has enjoyed a steady rise in popularity in a short time. EMO methodologies are being extended to address practicalities. He mentions that in the area of evolutionary computing and optimization, EMO research and application currently stands as one of the fastest growing fields.

For the multi-objective assignment model, all the criteria are not equally important. Generally, decision maker imparts priority ranking among the objectives, *eg.*, time is less important than quality. By incorporating the decision maker's preferences into the problem, the problem becomes hard to solve.

2.2 Summary of the Literature Review

From the literature review, it has been manifested that many research has been worked on this problem. Traditionally, the assignment problem is solved by considering a single objective. But in real life situation, it is observed that the application of MOAP is increased day by day. The attempt of this research is to solve MOAP using a genetic algorithm. A new methodology based on a genetic algorithm (GA) principles will be developed that will incorporate multi-objective and decision maker preferences to get a better result in less time than the other methods.

CHAPTER III

THEORETICAL CONSIDERATION

The following section will discuss different types of objective criteria and transformation of multi-objective to a single objective.

3.1 Types of Objective Criteria

Many types of research [4-7] have been developed to solve the assignment problem. Most of the developed methods for the assignment problem consider the only one-objective situation at a time, such as the minimum cost assignment problem, the minimum finishing time etc. assignment problem. The minimum cost assignment problem focuses on how to assign tasks to workers so that the total operation cost can be minimized. Such problems have been generally discussed and well developed in many operations research textbooks and in papers [1-7]. In the defence affair, often, the cost is less important than time. They may focus on how to assign the tasks to workers so that the total operation time can be minimized. There are some other criteria which are to be maximized rather than minimized such as quality, safety, precision etc. All of them are not quantifiable criteria but qualitative criteria.

3.2 Transformation of Multi-Objective to a Single Objective

In the industry, management has many objectives for the allocation of workers to tasks. We often come across with an assignment problem, where, cost and time are jointly co-related [4]. Time, cost, safety, quality etc. are often a simultaneous consideration in multi-objective assignment model. Single objective optimization is easy to solve but the multi-objective problem gets complex because of the conflicting nature of the objectives. These problems give rise to a set of trade-off among optimal solutions, popularly known as

Pareto-optimal solutions. Sometimes, it is necessary to take an aggregate decision regarding all the objectives criteria. This subsection describes the different ways to convert multi-objective to a single objective.

3.2.1 Case One

Yadaiah and Haragopal [24] proposed a method to translate multi-objectives assignment problem to single objective problem. They added a negative sign to the criteria that have to be maximized. For example Table 3.1 is written as Table 3.2.

Table 3.1: A three objective assignment problem

Worker \cost, time, quality\ Task	M1	M2
W1	15	16
	5	4
	3	5
W 2	13	14
	5	4
	3	1

Table 3.1 stands for a MOAP. Here two workers are to assign in two tasks so that cost and time are minimized meanwhile quality is maximized.

Table 3.2: Assigning negative sign before the value of quality

Worker \cost, time, quality\ Task	M1	M2
W1	15	16
	5	4
	-3	-5
W 2	13	14
	5	4
	-3	-1

Then they sum the cost, time and quality for an assignment of a worker to a task. This operation translates the multi-objective to the single objective problem. This is illustrated below (Table 3.3).

Table 3.3: Translated single objective problem

Worker \cost, time, quality\ Task	M1	M2
W1	17	15
W 2	15	17

Hungarian method or other methods can be used to solve the problem as a single objective problem in Table 3.3. This methodology gives the proper result when all the units of the criteria are equivalent *i.e.* 1 unit cost equivalent to 1 unit quality. But it is hardly possible to express all the criteria in an equivalent unit.

3.2.2 Case Two

Dervitsiotis [25] choose facility location based on objective factor (e.g. cost) and subjective factors (e.g. skill of worker, customer proximity, community attitude, and communication network). That method explained below.

Now it is to develop a measure of preference that combines both objective and subjective factors for each site to select the best location. A versatile method specially designed for the location-selection problem is a model developed by Brown and Gibson. Now it is being gone through following sequence of steps:

Step 1: There is the objective factor determination. Usually, all relevant costs are summed to compute the total annual cost for each site C_i . Next the objective factor OF_i is determined by multiplying C_i by the sum of the reciprocal site costs $\sum (1/C_i)$ and taking the reciprocal.

Table 3.4: Objective rating factor determination

Items		The site (<i>i</i>)	
		Location 1	Location 2
Annual costs (\$millions)	Human Resource	4.65	4.43
	Transportation	2.30	2.33
Total costs (C_i)		6.95	6.76
Reciprocal ($1/C_i$)		0.1439	0.1479
Sum of Reciprocal($\sum 1/C_i$)		0.2918	
Objective rating factor ($OF_i = \{ C_i(\sum 1/C_i) \}^{-1}$)		0.4931	0.5069

Step 2: Key subjective factors determination and their subjective-factor measure SF_i estimation is done for each site by

- a. Deriving a factor rating w_j for each subjective factor ($j= 1, 2, 3, \dots, n$) using a forced choice pairwise comparison procedure. Accordingly, one factor is selected over another, or they are rated equally.

The paired comparisons as follows:

- i. Skill of worker versus customer proximity: skill of worker as more important.
- ii. Skill of worker versus community attitude: skill of worker as more important.
- iii. Customer proximity versus community attitude: both judged equally important.

The information can be summarized in Table 3.5 in which it can compute the subjective-factor importance index w_j .

Table 3.5: Determination of relative importance of different Subjective factors

Factor j	Pairwise comparison			Relative importance, w_i
	1	2	3	
Skill of worker	1	1		0.50
Customer proximity	0		1	0.25
Community attitude		0	1	0.25

b. Ranking each site for each factor separately R_{ij} ($0 \leq R_{ij} \leq 1$, $\sum R_{ij} = 1$)

For each subjective factor separately it repeats the same pairwise comparisons with sites to determine their relative ranking R_{ij} . For the sites considered for location, selection is shown in Table 3.6, Table 3.7 and Table 3.8. A skilled worker is any worker who has a special skill, training, knowledge, and (usually acquired) ability in their work. A skilled worker may have attended a college, university or technical school. Or, a skilled worker may have learned their skills on the job. Examples of skilled labour jobs include software development, paramedics, police officers, painters, craftsmen, and accounting.

Table 3.6: Determination of relative importance of each site for the skill of the worker

Site i	Pairwise comparison	Sum of preferences	Relative importance, R_{i1}
	1		
Location 1	1	1	1
Location 2	0	0	0

Customer proximity is inevitable for business success. Higher customer proximity reduces transportation cost and minimizes advertising and awareness creating cost. It also builds a relationship with the customer. Community attitudes are attitudes held by groups of people that live in communities. Community attitudes play an important role in industrialization. For example, Tata Nano is not established in a west bangle.

Table 3.7: Determination of relative importance of each site for customer proximity

Site, i	Pairwise comparison	Sum of preferences	Relative importance, R_{i2}
	1		
Location 1	1	1	0.50
Location 2	1	1	0.50

Table 3.8: Determination of relative importance of each site for community attitude

Site, i	Pairwise comparison	Sum of preferences	Relative importance, R_{i3}
	1		
Location 1	1	1	0.50
Location 2	1	1	0.50

In the column for each pairwise comparison possible, we assign 1 to the factor preferred and 0 to other, while for the case equivalence both factors are assigned a value of 1. It is important in using such a procedure to check preferences for consistency. Thus if factor 1 is preferred to factor 2 and factor 2 is preferred to factor 3, factor 1 must be preferred to 3; otherwise, the responses are inconsistent [8].

- c. Combining for each site the subjective factor rating,

$$SF_i = w_1 R_{i1} + w_2 R_{i2} + \dots + w_n R_{in} \quad (3.1)$$

Step 3: Combining Objective rating factor and Subjective rating factor to determine overall location-preference measure:

Having completed the evaluation of both objective and subjective factors, we can now proceed to combine the results and determine an overall location preference measure (LPM_i) for each site.

Table 3.9: Summary of subjective factors evaluation

Factor	Site, i		Relative importance
	Location 1	Location 2	
Skill of Worker	1	0	0.50
Customer proximity	0.5	0.5	0.25
Community attitude	0.5	0.5	0.25
SF_i	0.75	0.25	

Then the location-preference measure will be

$$LPM_{Location\ 1} = 0.4931 + 0.75 = 1.2431 \quad (3.2)$$

$$LPM_{Location\ 2} = 0.5069 + 0.25 = 0.7569 \quad (3.3)$$

Step 4: The site with the maximum *LPM* is to select. Here site Location 1 has the highest *LPM* value and it is selected for the location facility.

This methodology eliminates the barriers of case one methodology. It does not require the expression of all criteria in equivalent units. An alternative solution gets the value in criteria as a percentage, where 100% is 1. The total sum of all the alternative solutions value in criteria is 1 in a normalized table.

3.2.3 Case Three

Bao et al. [16] uses 0-1 programming to translate a MOAP into a linear programming problem. The objective is to minimize cost, time and maximize quality simultaneously.

Table 3.10: A three-objective assignment problem

Worker \ cost, time, quality \ Task	M1	M2
W1	15 5 Fair	16 4 Good
W2	13 5 Fair	14 4 Poor

It needs to evaluate the quality criteria such as “good”, “fair”, and “poor”. One of the ways for this purpose is to quantify these quality criteria. For example: assign the criterion “good” as “5” point, the criterion “fair” as “3” point, and the criterion “poor” as “1” point and then rewrite Table 3.10 as Table 3.11.

Table 3.11: The three-objective assignment problem with the quantified quality

Worker \cost, time, quality\ Task	M1	M2
W1	15	16
	5	4
	3	5
W2	13	14
	5	4
	3	1

Note that the units for measuring time and cost and quality are different, it is not suitable to put “operation time” directly into the objective function. One direction for solving such a dilemma is to normalize first both the operation cost and operation time, that is, to divide both cost and time in Table 3.11 by the maximum cost and time, respectively. For example, the maximum operation cost, time in Table 3.11 is 16 and 5 respectively, so all the cost and time must be divided by 16 and 5 respectively. Since the demand for quality is maximal instead of minimal, the reciprocal of the quantified quality can be taken as the normalized quality.

Table 3.12: The normalized operation cost, time and quality

Worker \cost, time, quality\ Task	M1	M2
W1	0.938	1.000
	1.000	0.800
	0.333	0.200
W2	0.831	0.875
	1.000	0.800
	0.333	1.000

This normalized value will be used in the linear programming objective function. The methodology explained, in this case, give the results similar to case two. An alternative solution gets the value in criteria as a percentage, where 100% is 1. An alternative solution having a criteria value highest possess the value 1 in a normalized table. This method does not normalized quality in a better way when there is no quality value equal to 1 in the problem, then the normalized table lacks of the maximum value of quality 1.

3.2.4 Case Four

Hwang et al. [26] transform multi-objective to single objective by the technique for an order of preference by similarity to ideal solution (TOPSIS) methodology. It is based upon the concept that the chosen alternative should have the shortest distance from the positive ideal solution and the farthest from the negative ideal solution. TOPSIS can incorporate relative weights of criterion importance.

The idea of TOPSIS can be expressed in a series of steps [27].

- (1) Obtain performance data for n alternatives over m criteria x_{ij} ($i=1,\dots,n, j=1,\dots,K$).

Table 3.13: A three-objective assignment problem

Worker \cost, time, quality\ Task	M1	M2	M3
W1	15	18	14
	10	7	6
	3	5	3
W2	19	14	20
	5	6	7
	2	1	5
W3	18	15	19
	7	5	5
	5	3	2

The solution of the assignment problem given in Table 3.13 is shown in Table 3.14

Table 3.14: Solution of the assignment problem

Solution/ Worker/ Task	M1	M2	M3
Alternative 1	1	2	3
Alternative 2	1	3	2
Alternative 3	2	3	1
Alternative 4	2	1	3
Alternative 5	3	1	2
Alternative 6	3	2	1

Total cost, total time and total quality of each solution is given in Table 3.15

Table 3.15: Performance data of four alternatives

Solution/Performance/ Criteria	Total cost	Total time	Total quality
Alternative 1	48	21	6
Alternative 2	50	22	11
Alternative 3	48	16	8
Alternative 4	56	17	9
Alternative 5	56	21	15
Alternative 6	46	19	9

(2) Calculate normalized rating (vector normalization is used), r_{ij} .

Table 3.16: Normalized rating

Solutions	Cost	Time	Quality
Alternative 1	0.4998	0.5348	0.3453
Alternative 2	0.5206	0.5602	0.6330
Alternative 3	0.4998	0.4075	0.4603
Alternative 4	0.5831	0.4329	0.5179
Alternative 5	0.5831	0.5348	0.8632
Alternative 6	0.4790	0.4839	0.5179

(3) Develop a set of importance weights W_k , for each of the criteria. The basis for these weights can be anything, but, usually, is *ad hoc* reflective of relative importance.

$$V_{ij} = w_j \cdot r_{ij} \quad (3.4)$$

Table 3.17: Weight of each criterion (Depending on decision maker)

Criteria	Cost	Time	Quality
Weight	1/3	1/3	1/3

Table 3.18: Weighted decision criteria

Solutions	Cost	Time	Quality
Alternative 1	0.1666	0.1783	<u>0.1151</u>
Alternative 2	0.1735	<u>0.1867</u>	0.2110
Alternative 3	0.1666	0.1358	0.1534
Alternative 4	<u>0.1944</u>	0.1443	0.1726
Alternative 5	0.1944	0.1783	0.2877
Alternative 6	0.1597	0.1613	0.1726

(4) Identify the ideal alternative (extreme performance on each criterion) S^+ .

$$S^+ = \{v_1^+, v_{12}^+, \dots, v_j^+, \dots, v_k^+\}$$

$$= \{(\max v_{ij} \mid j \in J_1), (\min v_{ij} \mid j \in J_2), i = 1, 2, \dots, n\}$$

Where J_1 is a set of benefit attributes and J_2 is a set of cost attributes.

(5) Identify the nadir alternative (reverse extreme performance on each criterion) S^- .

$$S^- = \{v_1^-, v_{12}^-, \dots, v_j^-, \dots, v_k^-\}$$

$$= \{(\min v_{ij} \mid j \in J_1), (\max v_{ij} \mid j \in J_2), i = 1, 2, \dots, n\}$$

Table 3.19: Ideal alternative and nadir alternative

	Alternative	Cost	Time	Quality
Ideal	A*= 0.1597	0.1597	0.1358	0.2877
Nadir	A-= <u>0.1944</u>	<u>0.1944</u>	<u>0.1867</u>	<u>0.1151</u>

(6) Develop a distance measure over each criterion to both ideal (D^+) and nadir (D^-).

$$D_i^+ = \sqrt{\sum_j (v_{ij} - v_j^+)^2} \quad (3.5)$$

$$D_i^- = \sqrt{\sum_j (v_{ij} - v_j^-)^2} \quad (3.6)$$

Table 3.20: Distance measure of each alternative

Variable	Distance	Variable	Distance
D_1^+	0.1779	D_1^-	0.0290
D_2^+	0.0931	D_2^-	0.0981
D_3^+	0.1344	D_3^-	0.0695
D_4^+	0.1205	D_4^-	0.0715
D_5^-	0.0548	D_5^-	0.1728
D_6^-	0.1179	D_6^-	0.0719

(7) For each alternative, determine a ratio R equal to the distance to the nadir divided by the sum of the distance to the nadir and the distance to the ideal,

$$R = \frac{D^-}{D^- + D^+} \quad (3.7)$$

Table 3.21: Ranking of alternatives

Solutions	Ratio, R	Value of ratio, R	Rank
Alternative 1	R_1	0.1403	6
Alternative 2	R_2	0.5131	2
Alternative 3	R_3	0.3409	5
Alternative 4	R_4	0.3724	4
Alternative 5	R_5	0.7592	1
Alternative 6	R_6	0.3788	3

(8) Rank alternative according to ratio R (in Step 7) in descending order.

(9) Recommend the alternative with the maximum ratio: Alternative 5 is selected.

This methodology formulated the MOAP considering quality. The fuzzy method optimizes cost–time–quality more effectively.

CHAPTER IV

DEVELOPMENT OF MODEL AND SOLUTION METHODOLOGY

In this section, a model is developed and the solution methodology for solving the multi-objective problem is introduced.

4.1 The Model

The representation of notation and the mathematical model are done as follows.

4.1.1 Notation

Subscripts

i	Worker number
j	Task number
n	Number of total worker/ task
r	Chromosome's number
m	Number of total solutions/ chromosomes
c	Cost
t	Time
q	Quality

Parameters and matrixes

M_c	The Cost matrix
M_t	The Time matrix
M_q	The Quality matrix
C_{ij}	The element of the i^{th} row and j^{th} column in the cost matrix
T_{ij}	The element of the i^{th} row and j^{th} column in the time matrix

Q_{ij}	The element of the i^{th} row and j^{th} column in the quality matrix
b	Gene
L	Number of total genes in a chromosome
C_{max}	Maximum cost
T_{max}	Maximum time
Q_{max}	Maximum quality
N_c	Normalized cost matrix
N_t	Normalized time matrix
N_q	Normalized quality matrix
Cn_{ij}	The element of the i^{th} row and j^{th} column in the normalize cost matrix
Tn_{ij}	The element of the i^{th} row and j^{th} column in the normalize time matrix
Qn_{ij}	The element of the i^{th} row and j^{th} column in the normalize quality matrix
W_c	Weightage of cost
W_t	Weightage of time
W_q	Weightage of quality
P_m	Probability of mutation

4.1.2 Mathematical Model

A MOAP deals with cost, time, quality etc. [13]. The objectives of an assignment problem are to minimize both operating cost and operating time and to maximize quality simultaneously [14]. Suppose we have to assign n workers to n tasks in such a way that the overall operation cost, labour-time, and quality level are optimized.

It is noted that the units for measuring time, cost and quality are different. Generally, the quality criteria are expressed as “good”, “fair”, and “poor”. Therefore, it is necessary to quantify this quality criterion in terms of numerical value [21]. We assign 1 for “good”, 3 for “fair” and 5 for “poor” or researcher can express quality into more level in any

interval. This assignment imparts the highest value to the lowest quality and the lowest value to the highest quality [16]. It converts the requirement of maximum quality in MOAP into a minimum value of quality. Now the demand for the value of all the criteria viz. cost, time and quality is minimal. The assignment cost, time and quality are given in Table 4.1.

Table 4.1: A MOAP

Criteria	Worker, i	Task, j			
		1	2	..	n
Cost, C_{ij}	1	C_{11}	C_{12}	..	C_{1n}
	2	C_{21}	C_{22}	..	C_{2n}

	n	C_{n1}	C_{n2}	..	C_{nn}
Time, T_{ij}	1	T_{11}	T_{12}	..	T_{1n}
	2	T_{21}	T_{22}	..	T_{2n}

	n	T_{n1}	T_{n2}	..	T_{nn}
Quality, Q_{ij}	1	Q_{11}	Q_{12}	..	Q_{1n}
	2	Q_{21}	Q_{22}	..	Q_{2n}

	n	Q_{n1}	Q_{n2}	..	Q_{nn}

The problem can be stated as,

$$\text{Minimize } \sum_{i=1}^n \sum_{j=1}^n X_{ij} C_{ij} \quad (4.1)$$

$$\text{Minimize } \sum_{i=1}^n \sum_{j=1}^n X_{ij} T_{ij} \quad (4.2)$$

$$\text{Minimize } \sum_{i=1}^n \sum_{j=1}^n X_{ij} Q_{ij} \quad (4.3)$$

Where

$$X_{ij} = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ worker is assigned to } j^{\text{th}} \text{ task} \\ 0 & \text{Otherwise} \end{cases} \quad (4.4)$$

$$\sum_{i=1}^n x_{ij} = 1; j = 1, 2, \dots, n \text{ (only one person is assigned to the } j^{\text{th}} \text{ task)} \quad (4.5)$$

$$\sum_{j=1}^n x_{ij} = 1; i = 1, 2, \dots, n \text{ (only one task is done by the } i^{\text{th}} \text{ worker)} \quad (4.6)$$

4.2 The Genetic Algorithm for the MOAP

In this sub-section, we present the basis and the details of the proposed genetic algorithm.

4.2.1 Basis of the Formulation

The values of one criterion (*e.g.* cost) may be very high and for another criterion (*e.g.* time) may be very low. In the case of minimization, the criteria having high values play an important role by ignoring the criteria having low value [18]. So it requires the conversion of all the criteria into a similar scale. The process of normalization translate all the criteria into a similar scale. For the purpose of normalization, first, maximum operation cost and maximum operation time and maximum operation quality are determined. To find the normalized matrix, all the cost, time and quality are divided by the maximum operation cost, maximum operation time, and maximum operation quality respectively.

Maximum *cost*,

$$C_{max} = \text{Max} (C_{ij}) \quad (4.7)$$

Maximum *time*,

$$T_{max} = \text{Max} (T_{ij}) \quad (4.8)$$

Maximum *quality*,

$$Q_{max} = \text{Max} (Q_{ij}) \quad (4.9)$$

Normalized *cost matrix*,

$$N_c = Cn_{ij} = M_c/C_{max} = [C_{ij}/C_{max}] \quad (4.10)$$

Normalized *time matrix*,

$$N_t = Tn_{ij} = M_t/T_{max} = [T_{ij}/T_{max}] \quad (4.11)$$

Normalized *quality matrix*,

$$N_q = Qn_{ij} = M_q/Q_{max} = [Q_{ij}/Q_{max}] \quad (4.12)$$

4.2.2 The Details of the Proposed Algorithm

The basic structure of the proposed GA algorithm to solve MOAP is as follows:

- Step 1.** Create an initial population of m chromosomes where $\frac{m}{2}$ is an even number (generation 0).
- Step 2.** Evaluate the fitness of each chromosome.
- Step 3.** Select parents from the current population via *proportional selection* (*i.e.* the selection probability is proportional to the fitness). The number of total parents is $\frac{m}{2}$.
- Step 4.** Choose at random a pair of parents for mating and apply partially mapped crossover (PMX) to create two offspring. A parent is chosen for mating for one time.
- Step 5.** Apply mutation operator to offsprings, and insert the resulting offsprings in the new population with their parents.
- Step 6.** Repeat steps 5 and 6 until all parents are selected and mated. (*i.e.* offspring are created).
- Step 7.** Find the best chromosome from all the parents and child of the new population. Replace the ‘best chromosome so far’ by the best chromosome of the new population when the later one is superior.
- Step 8.** Replace the old population of chromosomes by the new one.
- Step 9.** Go back to step 2 if the last generation does not provide a better solution for several iterations. Otherwise, the final solution is the ‘best chromosome so far’ created during the search.

In the forthcoming section, we will describe the details of the implementation of GA in solving MOAP.

4.2.2.1 Representation of solution

The representation of the solution structure of the MOAP is discussed here. Symbolic ordered gene (*i.e.* the value of two alleles cannot be same [19]) strings of length n (total number of tasks) are used to represent solution (chromosome), henceforth called tasks chromosome. The chromosome has one allele for each task. The position of an allele in task chromosome represent the task number while the allele value is the worker number who is assigned to that task [20]. For example, let a string consists of genes (4, 3, 1, 5, 2). The allele at the first locus of the string signifies that the worker number 4 is assigned to task number 1, worker number 3 is assigned to task number 2, and so on.

4.2.2.2 Initial population

The encoded solution is represented as a chromosome. The initial set of solutions *i.e.* the population size of m is generated randomly where $\frac{m}{2}$ is an even number, allowing the entire range of possible solutions (the *search space*).

For a particular solution *i.e.* chromosome, the genes,

$$\begin{aligned} b_j &= i \text{ where } i^{\text{th}} \text{ worker is assigned to } j^{\text{th}} \text{ task for} \\ j &= 1, 2, \dots, n \end{aligned} \tag{4.13}$$

And the chromosome r ,

$$chr_{m,r} = b_j \text{ for } r = 1, 2, \dots, m \text{ and } j = 1, 2, \dots, n \tag{4.14}$$

4.2.2.3 Fitness function

The fitness function works as objective an function which is needed to be maximized. In this study, the function F_r is used to form fitness functions for chromosome number r .

First, the total normalized cost, the total normalized time, and the total normalized quality are determined. It is often given different priority on cost, time and quality. The Fitness of a chromosome depends on this priority. The W_c , W_t and W_q are a weightage of the cost, time and quality respectively. Since the MOAP is a minimization problem, the high fitness value is associated with minimized cost, minimized time and minimized quality value. It is to make the fitness choice criteria maximum, thus the inverse of sum product of priority and respective total normalized values is taken in equation (4.18).

For a particular chromosome *i.e.* solution,

The total normalized cost,

$$Tot_c = \sum_{j=1}^n Cn_{b_j j} \quad (4.15)$$

The total normalized time,

$$Tot_t = \sum_{j=1}^n Tn_{b_j j} \quad (4.16)$$

The total normalized quality,

$$Tot_q = \sum_{j=1}^n Qn_{b_j j} \quad (4.17)$$

The fitness of chromosome,

$$F_r = 1 / (W_c * Tot_c + W_t * Tot_t + W_q * Tot_q) \quad (4.18)$$

4.2.2.4 Reproduction

In the present implementation, the *proportional selection* (*i.e.* the selection probability is proportional to the fitness) is used is use [21]. The expected number of chromosomes going from the parent generation to the mating pool depends on the individual fitness values [22]. The probability of selection for chromosome r is

$$pselect_r = \frac{F_r}{\sum_{r=1}^m F_r} \quad (4.19)$$

4.2.2.5 Crossover

Every chromosome is an order list of the workers, so the direct swap is not possible. Partially Matched Crossover (PMX) and cycle crossover (CX) are widely used for the crossover of ordered chromosomes [23]. PMX and CX are not really competitive with the order-preserving crossover operators [28]. Partially Matched Crossover (PMX) which was initially developed for tackling the “Travelling Salesman Problem”, is chosen as the crossover operator in this model. The crossover in the proposed methodology is explained below.

Each individual in the mating pool has the same chance of being parent independent of its fitness. Two parent chromosomes from the mating pool are chosen randomly. Crossover occurs between this two parents. The locus of the cross-over points is generated randomly. For example, it is to crossover between,

$\text{chrom}_1 = (1 \ 8 \ 2 \ 4 \ 7 \ 6 \ 5 \ 3)$ and

$\text{chrom}_2 = (2 \ 7 \ 5 \ 3 \ 1 \ 6 \ 8 \ 4)$.

Two random number is generated between 1 and L ($L=7$ in this case). Let it ‘3’ and ‘5’. The locus of the crossover point is shown by ‘dot’ before position ‘3’ and after position ‘5’.

$\text{chrom}_1 = (1 \ 8 \ . \ 2 \ 4 \ 7 \ . \ 6 \ 5 \ 3)$

$\text{chrom}_2 = (2 \ 7 \ . \ 5 \ 3 \ 1 \ . \ 6 \ 8 \ 4)$

Now the portion between the selected crossover points is swapped and the rest of the values are changed according to the PMX rule [25]. After exchanging the information, the two offspring are,

$\text{chrom}_1' = (7 \ 8 \ . \ 5 \ 3 \ 1 \ . \ 6 \ 2 \ 4)$

$\text{chrom}_2' = (5 \ 1 \ . \ 2 \ 4 \ 7 \ . \ 6 \ 8 \ 3)$

The resulting two chromosomes, called the offspring, added to the population with their parents. The offspring cannot be chosen for crossover until the next generation. The process is repeated until the mating pool is not empty, where a parent in the mating pool take part in crossover for only one time.

4.2.2.6 Mutation

This mutation operator is the closest in philosophy to the biological mutation operator because it only slightly modifies the original chromosome [26]. In this accomplishment, we have done the two alleles swapping for each chromosome, in offspring, with the probability of mutation, pm. For illustration, let us consider the chromosome, from the previous example,

$$\text{chr}_{m_1}' = (7 \ 8 \ 5 \ 3 \ 1 \ 6 \ 2 \ 4)$$

Suppose the locus chosen for mutation is 2 and 5. Then, after mutation, the new chromosome (offspring) will be,

$$\text{chr}_{m_1}'' = (7 \ 1 \ 5 \ 3 \ 8 \ 6 \ 2 \ 4)$$

4.2.2.7 Termination

When there is no improvement of the highest fitness value attained for successive $(n+10)$ generations, it stops creating a new generation. And the chromosome having the highest fitness in all the generations is taken as a solution of the MOAP.

4.2.2.8 Extracting the values of decision variables from the best chromosome

Chromosomes *i.e.* solution is made of genes, like r^{th} chromosome,

$$\text{chr}_{m_r} = b_j \text{ for } j = 1, 2, \dots, n \quad (\text{here } b_j \text{ is the value of } j^{th} \text{ gene}) \quad (4.20)$$

Now, we will set the values of decision variables according to the chromosome, as follows.

For $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, n$

$$X_{ij} = \begin{cases} 1 & \text{for } i = b_j \\ 0 & \text{Otherwise} \end{cases} \quad (4.21)$$

4.3 Numerical Illustration and Computational Experience

4.3.1 Performance Study

The genetic algorithm was coded in ANSI C programming language using only simple array data structure and implemented on a PC (Core i3, Intel processor of 2.2 GHz). Let us consider a problem where cost, time and quality are considered. And we have to assign 6 workers to 6 tasks based on the data in Table 4.2.

Table 4.2: A numerical example of a MOAP [29]

Criteria	Worker, i	Task, j					
		1	2	3	4	5	6
Cost, C_{ij}	1	6	3	5	8	10	6
	2	6	4	6	5	9	8
	3	11	7	4	8	3	2
	4	9	10	8	6	10	4
	5	4	6	7	9	8	7
	6	3	5	11	10	12	8
Time, T_{ij}	1	4	20	9	3	8	9
	2	6	18	8	7	17	8
	3	2	8	20	7	15	7
	4	12	13	14	6	9	10
	5	9	8	7	14	5	9
	6	17	13	3	4	13	7
Quality, Q_{ij}	1	1	3	1	1	1	5
	2	3	5	3	5	7	5
	3	1	7	5	3	5	7
	4	5	9	3	5	7	3
	5	3	9	7	5	3	3
	6	3	3	5	7	5	7

Using the methodology provided in this work with the equal priority of cost, time and quality the solution is following which corresponds to the total cost of 42 units, the total time of 41 units and the total quality of 14 units. The solution is

$$x_{14} = x_{23} = x_{31} = x_{46} = x_{55} = x_{62} = 1$$

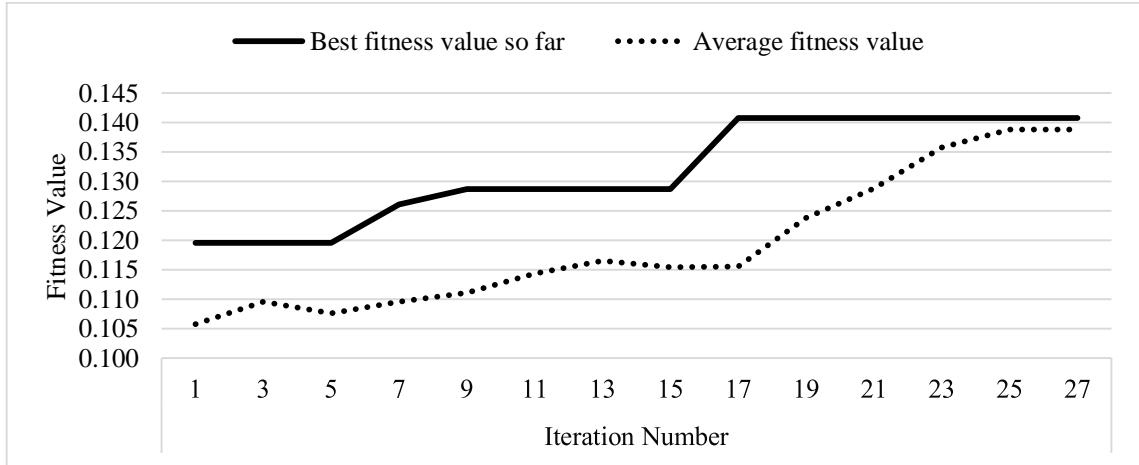


Figure 4.1: Convergence to the global maximum fitness

We get the optimum result at the generation number 17 for the population size of 24. It is shown in the Figure 4.1. We also test the problem with various population sizes. The effect population size changing on the other parameters is shown in Table 4.3.

Table 4.3: Effect of population size changing on different parameters

Trial No.	Population size, m	CPU time to get the best solution (second)	No. of iteration required to get the best solution	The best fitness value attained so far
1	24	0.016	17	0.0140735
2	40	0.016	22	0.0140735
3	60	0.016	15	0.0140735
4	100	0.016	4	0.0140735
5	200	0.016	7	0.0140735
6	400	0.016	3	0.0140735

In the following figure we will see how different values of priority weightages (independent variable) affect the dependent values viz. total cost, total time and total

quality. The weightage of one criteria has been changed while keeping the weightage of other criteria ‘1’ in this experiment.

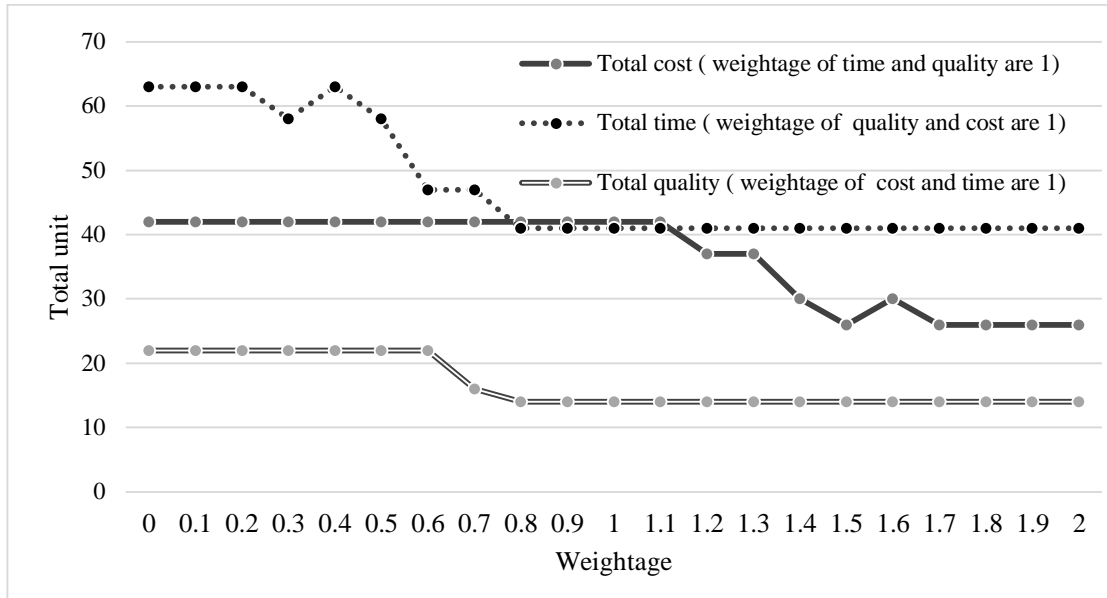


Figure 4.2: Sensitivity of total cost, total time and total quality to their relative weightage

4.3.2 Comparisons of the Proposed Approach with Other Methodologies

To further review of the proposed approach, the results of the developed approach has been compared with the experimental result produced by the multi-objective fuzzy deployment methodology developed by Tsai et al. [11]. and a new approach of Bao et al. [9]. The method of Tsai et al. and the method of Bao et al. have produced similar results. So we will mention multi-objective fuzzy deployment methodology of Tsai et al. in the comparisons.

4.3.2.1 Example 1

At first, we will consider a small size problem *viz.* 6 tasks-6 workers three-objective assignment problem (Table 4.4). After solving this problem, we get the following result.

Table 4.4: 6 tasks-6 workers MOAP solution

Methodology	Worker order	Total normalized cost	Total normalized time	Total normalized quality	Total normalized value
Proposed methodology	W3, W6, W2, W1, W5, W4	3.5000	2.0500	1.5556	7.01055
Multi-objective fuzzy Deployment methodology	W2, W6,W1, W3,W5,W4	3.0000	2.5000	1.7778	7.2778

This solution results in 42 units operation cost, 41 units operation time and 14 units quality. When the above problem is solved by multi-objective fuzzy deployment methodology [13], it finds the solution that results in 36 units operation cost, 50 units operation time and 16 units quality.

4.3.2.2 Example 2

Now we considering medium size problem *viz.* 10 tasks-10 worker three-objective assignment problem, shown in Table 4.5.

Table 4.5: 10 tasks-10 workers three-objective assignment problem

Criteria	Worker, <i>i</i>	Tasks, <i>j</i>									
		1	2	3	4	5	6	7	8	9	10
<i>c_{ij}</i>	1	6	3	5	8	4	10	8	4	4	6
	2	6	4	6	5	8	9	7	14	3	7
	3	11	7	4	8	7	8	6	16	12	6
	4	9	10	8	6	10	9	7	14	2	7
	5	10	13	7	4	8	7	10	7	14	10
	6	11	12	10	8	4	4	6	5	4	13
	7	12	4	6	5	8	7	4	8	8	12
	8	13	7	4	8	7	10	7	14	5	4
	9	14	10	8	6	16	13	6	16	8	7
	10	18	11	7	4	8	11	7	14	14	6
<i>t_{ij}</i>	1	4	20	9	3	10	4	20	9	3	10
	2	6	18	8	7	12	6	18	8	7	12
	3	7	8	9	7	14	3	8	20	7	14
	4	12	14	8	6	16	12	13	14	6	16
	5	11	8	9	7	14	2	8	9	7	18
	6	13	8	15	16	8	18	8	8	6	15
	7	12	13	8	6	16	12	13	8	13	9
	8	11	14	9	7	8	13	14	6	4	8
	9	13	13	8	6	7	14	2	8	19	20
	10	9	7	8	13	6	16	12	13	14	14
<i>q_{ij}</i>	1	1	3	1	1	1	3	5	3	5	3
	2	3	5	3	5	3	1	7	5	3	5
	3	1	7	5	3	5	5	9	3	5	3
	4	5	9	3	5	3	1	7	5	3	5
	5	1	7	5	5	5	5	9	3	5	3
	6	9	3	3	3	1	5	5	3	3	5
	7	7	5	5	5	5	7	3	3	5	3
	8	3	3	7	3	7	9	3	5	3	3
	9	5	5	9	5	9	5	5	1	5	5
	10	3	5	7	5	3	1	7	5	3	7

After solving the 10 tasks-10 workers three-objective assignment problem (Table 4.5), the result is shown in Table 4.6.

Table 4.6: 10 tasks-10 worker MOAP solution

Methodology	Worker order	Total normalized cost	Total normalized time	Total normalized quality	Total normalized value
Proposed methodology	W3, W10, W2, W1, W6, W5, W9, W7, W4, W8	3.7222	2.9500	3.3333	10.0056
Multi-objective fuzzy Deployment methodology	W3, W7, W1, W5, W10, W2, W9, W6, W4, W8,	3.2222	3.6000	3.3333	10.1556

The proposed methodology results in 67 units operation cost, 59 units operation time and 30 units quality. When the above problem is solved by multi-objective fuzzy deployment model, it finds the following solution that results in 58 units operation cost, 72 units operation time and 30 units quality.

4.3.2.3 Example 3

At last, we are considering large size problem *viz.* 14 tasks-14 workers three-objective assignment problem mentioned in Table 4.7.

Table 4.7: 14 tasks-14 workers three-objective assignment problem

Criteria	Worker, <i>i</i>	Tasks, <i>j</i>													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
<i>c_{ij}</i>	1	6	3	5	8	4	10	8	4	4	6	7	4	8	7
	2	6	4	6	5	8	9	7	14	3	7	10	8	4	4
	3	11	7	4	8	7	8	6	16	12	6	6	5	8	7
	4	9	10	8	6	10	9	7	14	2	7	4	8	7	10
	5	10	13	7	4	8	7	10	7	14	10	8	6	16	13
	6	11	12	10	8	4	4	6	5	4	13	7	4	8	7
	7	12	4	6	5	8	7	12	4	7	4	8	10	8	4
	8	13	7	4	8	7	10	7	8	10	8	6	6	16	7
	9	14	10	8	6	16	13	6	5	11	7	4	7	14	10
	10	18	11	7	4	6	5	8	8	7	4	8	4	8	7
	11	13	7	4	7	4	8	10	8	6	16	7	8	6	16
	12	12	10	8	10	8	6	6	16	14	2	10	7	4	6
	13	4	6	5	11	7	4	7	14	7	14	6	4	7	4
	14	7	4	8	7	4	8	10	7	5	4	4	8	7	10
<i>t_{ij}</i>	1	16	8	18	8	10	4	20	9	13	8	8	8	20	8
	2	6	16	12	13	12	6	18	8	14	13	13	13	14	15
	3	7	8	13	8	19	3	8	20	8	4	8	8	9	8
	4	6	7	14	13	14	12	13	14	15	6	18	8	20	8
	5	6	3	8	4	20	2	8	9	8	6	16	15	16	8
	6	13	8	15	6	18	8	20	8	6	15	6	18	8	20
	7	12	13	8	6	16	15	16	8	13	8	6	16	15	16
	8	11	14	9	7	8	8	6	6	4	8	13	8	19	3
	9	20	9	3	8	13	14	7	8	19	20	14	13	14	12
	10	18	8	7	7	14	2	10	13	14	14	8	4	20	2
	11	8	20	7	3	8	20	12	4	20	13	8	19	3	8
	12	8	6	16	16	2	7	14	6	18	14	13	14	12	13
	13	9	7	8	18	18	6	16	7	8	8	4	20	2	8
	14	3	8	13	15	12	7	18	12	14	15	6	18	8	20

Table 4.7: 14 tasks-14 workers three-objective assignment problem (Cont.)

Criteria	Worker, <i>i</i>	Tasks, <i>j</i>													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
<i>q_{ij}</i>	1	1	3	1	1	1	3	5	3	5	3	5	5	3	3
	2	3	5	3	5	3	1	7	5	3	3	3	1	3	5
	3	1	7	5	3	5	5	9	3	5	3	3	1	3	5
	4	5	9	3	5	3	1	7	5	3	3	3	7	7	5
	5	1	7	5	5	5	5	9	3	5	5	5	7	5	3
	6	9	3	3	3	1	5	5	3	5	3	3	1	3	5
	7	7	5	5	5	5	7	3	3	5	5	5	5	5	5
	8	3	3	7	3	7	9	3	5	3	3	3	7	3	7
	9	5	5	9	5	9	5	5	3	3	1	3	5	3	9
	10	3	5	7	5	3	1	7	5	3	7	5	7	5	3
	11	7	5	5	5	3	3	5	3	3	3	5	5	5	3
	12	3	3	3	1	3	5	3	3	5	5	3	7	3	7
	13	5	5	5	5	5	1	3	5	3	3	1	3	5	3
	14	3	7	3	7	7	5	5	1	5	3	5	7	5	3

After solving the 14 tasks-14 workers three-objective assignment problem (Table 4.7), the result is shown in Table 4.8.

Table 4.8: 14 tasks-14 workers MOAP solution

Methodology	Worker order	Total normalized cost	Total normalized time	Total normalized quality	Total normalized value
Proposed methodology	W14, W1, W2, W5, W12, W10, W9, W7, W6, W4, W13, W3, W11, W8	4.3333	3.8000	5.3333	13.4667
Multi-objective fuzzy Deployment methodology	W5, W1, W2, W6, W11, W10, W9, W7, W8, W4, W14, W3, W12, W13	4.4444	5.0500	4.444	13.9388

This solution by proposed methodology results in 78 units operation cost, 76 units operation time and 48 units quality. When the above problem is solved by multi-objective fuzzy deployment it finds the solution that results in 80 units operation cost, 101 units operation time and 40 units quality.

CHAPTER V

RESULTS & DISCUSSIONS

5.1 Results

In this section, we will demonstrate the solutions of the above three examples in two methodologies *i.e.* proposed methodology and multi-objective fuzzy deployment methodology at glance. It reveals that in the first example, solution by proposed methodology reduces 9 units time and 2 units quality with sacrificing of 6 unit cost with respect to the result of multi-objective fuzzy deployment methodology. Similarly, in the second example *i.e.* 10 tasks-10 workers multi-objective problem, the solution by proposed methodology reduces 13 units time with the increase of 9 units cost. And, in the third example solution *i.e.* 14 tasks-14 workers multi-objective problem, the solution by proposed methodology reduces 2 units cost and 25 units time with the increasing 8 units quality.

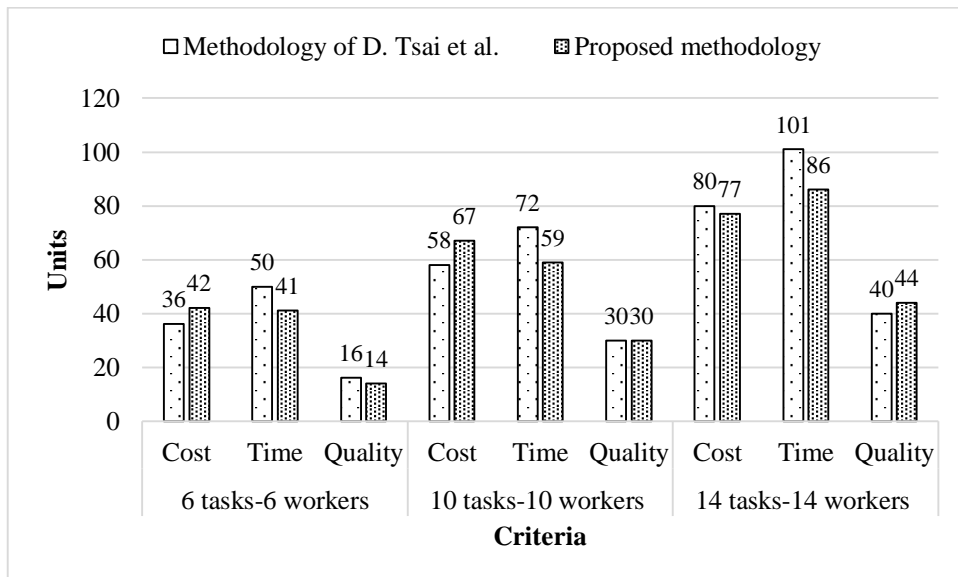


Figure 5.1: Results in two methodologies for three examples

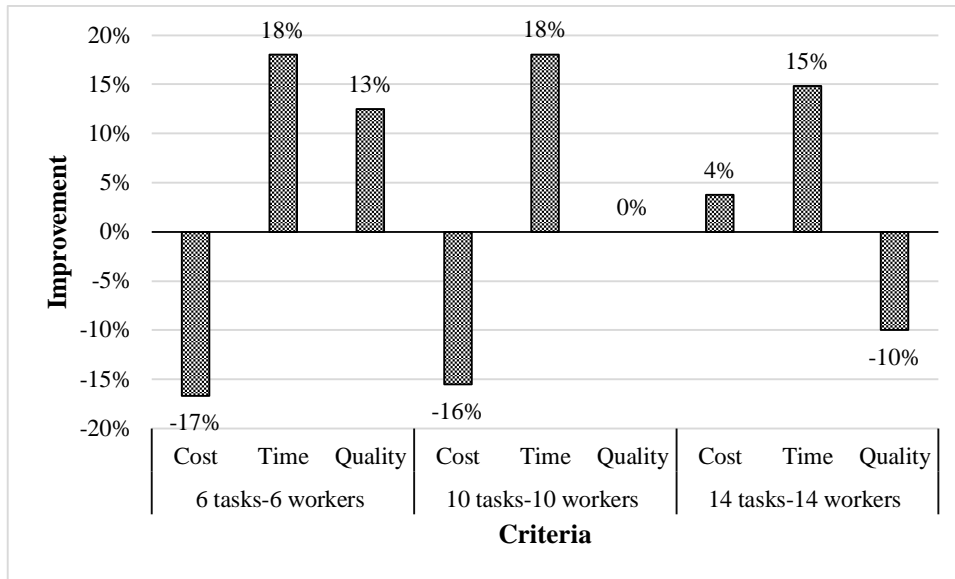


Figure 5.2: Improvement by the proposed methodology for each criterion in equal priority

The above three multi-objective problems have been solved with the equal priority of cost, time and quality. There is the demonstration of percentage improvement for each criterion by proposed model from the multi-objective fuzzy deployment methodology in Figure 5.2. The above comparisons give the manifestation that the solutions of multi-objective problems by proposed model are acceptable and even better than the solutions by multi-objective fuzzy deployment model.

New Approach of Bao et al. [9] cannot normalize quality properly when there is no assignment having the quality of '1'. In this case, the reciprocal of quality distributes normalize quality in an interval $[0,1)$ while the others normalized criteria is distributed in an interval $[0,1]$. The proposed methodology distributes all the criteria into a normalized value of interval $[0,1]$.

The benefit of the proposed methodology is that firstly, it doesn't require careful attention to the determinations of the weight among the resources like the methodology of Tsai et al. Careful attention requires experts i.e. time and cost, thus the proposed methodology reduces the time for weightage allocation among the resources in MOAP. Secondly, its calculation is easier than other methods. Thirdly, this method incorporates priority of the

resources in decision making. The above comparisons manifest that the outcomes of the proposed methodology are very encouraging.

It is revealed that the proposed methodology is acceptable to solve MOAP. This methodology imparts better solutions with the determination of decision variables that give well-optimized cost, time and quality than multi-objective fuzzy deployment methodology [18]. This research provides simplicity, flexibility and structured thinking to MOAP. The proposed methodology structures the preferences of the decision maker.

5.2 Discussions

MOAP can be solved by finding the best solution by testing all the possible solutions (by permutation) for hours on a powerful computer, but it would be costly. The proposed methodology find the optimum solution for small problem and closet to optimum solution for a big problem very quickly on a small microcomputer. Excel is available for the solutions up to 14 tasks-14 workers three-objective assignment problem by multi-objective fuzzy deployment methodology. The developed software for applying the proposed methodology can solve up to 99 tasks-99 workers MOAP within five seconds in 2 GB ram, 2.2 GHz personal computer.

There were the multi-objectives which were conflicting among them to get a better solution. These conflicting nature of objectives give rise to a set of trade-off among optimal solutions. The quality objective has been normalized in a new way which can handle any non-quantified entity. Generally, the demand for the quality need to be maximized, in this methodology quality is quantified in such a way that converts the demand of quality need to be minimized. Then the demand for the value of all the criteria viz. cost, time and quality are to be minimized.

The total cost, total time and total quality is sensitive to each priority weightage in a certain range. Beyond this range, there are the saturation point of total cost, total time and total quality whether it increase or decrease the priority i.e. weightage of a criteria (Figure 4.2). It is manifested that the total cost and the total time are more sensitive to its priority

than the total quality. The elements in the cost matrix and the time matrix have more variation than the quality matrix, which results in the less sensitiveness of total quality than the total coast and total time.

This method enriched with comprising this decision maker priority weightage. The decision maker often adds priority to different criteria e.g. speed is two times preferable to load for a fighter plane. At the last portion of the thesis, the comparison of proposed method with multi-objective fuzzy deployment method results that assignment problem has been formulated and solved effectively.

CHAPTER VI

CONCLUSIONS

6.1 Conclusions

In the present paper, the MOAP has been formulated to solve by genetic algorithm. This works formulates the conflicting objectives (Pareto-optimal), converts maximal criteria into minimal criteria and imparts priority to each objective. The quality objective has been normalized in a way which can handle any non-quantified entity [16]. Generally, the demand for the quality need to be maximized, in this methodology quality is quantified in such a way that converts the demand of quality need to be minimized. Then the demand for the value of all the criteria viz. cost, time and quality are to be minimized. A procedure has been demonstrated to solve the problem based on a genetic algorithm. The algorithm seems to be quite effective in finding the globally optimal solution to the MOAP within a reasonable time.

6.2 Limitations

The developed software can handle up to 99 tasks-99 workers multi-objective assignment. There is the comparison of solutions between proposed methodology and fuzzy deployment methodology up to 14 tasks-14 workers MOAP, because, free software is available for the solution of linear programming of 300 decision variables. It would be better if the comparison took place for the solutions of a large number of tasks and a large number of workers MOAP. Additionally, Genetic algorithms can often escape from the local optimums if they are shallow enough [27]. GA cannot able to guarantee that our genetic algorithm has found the global optimum solution to our problem, the best it can do is hope for is a close approximation of the optimal solution.

6.3 Future Possible Works

The traditional direction of solving the assignment problem is 'minimum time', which may not be true in a real situation. In the real world, the manager may want to assign tasks so that each doer has standard time to finish the task which is fixed for all the worker. There is the requirement of research that how to solve the assignment problem with the direction of 'standard time' instead of 'minimized'.

REFERENCES

- [1] Pramanik and Biswas, Multi-objective Assignment Problem with Generalized Trapezoidal Fuzzy Numbers, *International Journal of Applied Information System*, vol 2, pp 13-20 (2012).
- [2] Reddy, S., Multi-objective optimization considering cost, emission and loss objectives using PSO and fuzzy approach, *International Journal of Engineering & Technology*, Vol 7, No. 3, pp 1552-1557 (2018).
- [3] Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Reading, *Addison-Wesley Publishing Company, Inc.*, 1st edition, Boston (1989).
- [4] Ortiz-Boyer, Hervás-Martínez and García-Pedrajas, A Crossover Operator for Evolutionary Algorithms Based on Population Features, *Journal of Artificial Intelligence Research*, Vol 24, pp 1-48 (2005).
- [5] Soni and Kumar, Dr., Study of Various Crossover Operators in Genetic Algorithms, *International Journal of Computer Science and Information Technologies*, vol 5, pp 7235-7238 (2014).
- [6] Singh, Dubey, G.C. and Shrivastava, A Comparative Analysis of Assignment Problem, *Journal of Engineering (IOSRJEN)*, Vol 2, No 8, pp 01-15 (2012).
- [7] Khun, H.W., The Hungarian method for assignment problem, *Naval Research Logistics*, vol 2, No. Quarterly, pp 83-97 (1955).
- [8] Taha, Operation Research: An Introduction, 8th edition, New Delhi (2006).
- [9] BERTSEKAS, A New Algorithm for the Assignment Problem, Mathematical Programming, *North-Holland Publishing Company*, vol 21, pp 152-171 (1981).
- [10] Gillett, Introduction to Operations Research A Computer-Oriented Algorithmic Approach. *Tata McGraw-Hill Publishing Company Limited*, New Delhi, ch 3, ch 4 (1979).
- [11] Sahu and Thapadar, Solving the Assignment problem using Genetic Algorithm and Simulated Annealing, *International Journal of Applied Mathematics I*, vol 36, No 1, pp 7-11, (2017).
- [12] Kirubha, Prompting Method for Solving Fuzzy Assignment Problem under Fuzzy Environment, *Imperial Journal of Interdisciplinary Research (IJIR)*, Vol 2, Issue-

- 4, ISSN: 2454-1362 (2016).
- [13] Yiwen and Yaodong, Fuzzy random TSP, *Fifth international conference on electronic business*, Hong Kong, vol 5, pp 901-907 (2005).
 - [14] Liu and Gao, Fuzzy weighted equilibrium multi job assignment problem and genetic algorithm. *Applied mathematical modelling*, vol 33, pp 3926-3935 (2009).
 - [15] Huang and zhang, Solution method for fuzzy assignment problem with restriction of qualification. *6th international conference on intelligent system design and application* (2006).
 - [16] Bao, Tsai and Tsai, A New Approach to Study the Multi-Objective Assignment Problem, *An Interdisciplinary Journal*, vol 53, pp 123-132 (2007).
 - [17] Ishizuka and Matsuo, SL method for computing a near-optimal solution using linear and nonlinear programming in cost based hypothetical reasoning, *Knowledge based system, Elsevier*, vol 15, pp 369-376 (2002).
 - [18] Tsai, Wei and Cheng, Multiobjective fuzzy deployment of manpower, *International Journal of the Computer, the Internet and Management*, vol 7, pp 1-7 (1999).
 - [19] Geetha, S. and Nair, K. P. K., A variation of the assignment problem, *European Journal of Operational research*, Vol 68, No 3, pp 422-426 (1993).
 - [20] Boussaïd, Lepagnot and Siarry, A survey on optimization metaheuristics, *Elsevier, information Sciences*, vol 237, pp 82-117 (2013).
 - [21] Yeniay, Penalty Function Methods for Constrained Optimization with Genetic Algorithms, *Mathematical and Computational Applications*, Vol 10, No 1, pp 45-56 (2005).
 - [22] Mitchell, An Introduction to Genetic Algorithms, *MIT Press*, ISBN 0-262-13316-4 (HB), 0-262-63185-7 (PB), Fifth printing (1999).
 - [23] Deb, Kalyanmoy, Multi-Objective Optimization using Evolutionary Algorithms, *John Wiley & Sons Ltd*, New York, ISBN:047187339X (2001).
 - [24] Yadaiah and Haragopal, V.V., Multi-Objective Optimization of Time-Cost-Quality Using Hungarian Algorithm, *American Journal of Operations Research*, vol 6, pp 31-35 (2016).
 - [25] Dervitsiotis, Operation Management, *McGraw-Hill Ltd*, New York (1981).
 - [26] Hwang, Ching-lai and Yoon, Multiple Attribute Decision Making: Methods and Applications, *Springer-Verlag*, New York (1981).

- [27] Mousa, A. A., Using genetic algorithm and TOPSIS technique for multi-objective transportation problem: a hybrid approach, *International Journal of Computer Mathematics*, vol 87, No 13, pp 3017-3029 (2014).
- [28] Starkweather, T., McDaniel, S., Mathias, K. and Whitley, D., A comparison of genetic sequencing operators, *4th International Conference on Genetic Algorithms (ICGA '91)*, University of California at San Diego, San Diego, pp 69-76 (1991).

APPENDIX A

C code for MOAP solution software:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

int n3=20; //population size
float pr1=1,pr2=1,pr3=1;// Priority
int n77=400; // heighest no of excess generation
int num=6; //number of machine
int n1,n2,n4;
char fname[50]; //target file name
FILE *fp;
int mut; // %chance of mutation
int gen; // result at the generation of
typedef struct Chrom
{ int bit[100]; float fit; int parent;
}chrom;
typedef struct Tablecell
{ int cost; float ncost;
int time; float ntime;
int quality; float nquality;
}cell;

void *TakeInput(cell[100][100] );
void *processTable(cell node[100][100]);
void *printTable(cell node[100][100]);

void *printProcesTable(cell node[100][100]);
void *Generategens(chrom currentGen[n4],cell node[100][100]);
void *SelectNextGenPar(chrom currentGen[n4],chrom nextGen[n4]);
void *crossover(chrom currentGen[n4],cell node[100][100]);

void *fitness(chrom currentGen[n4],cell node[100][100]);
void *sort(chrom currentGen[n4]);
void *mutation(chrom currentGen[n4]);

void *result(chrom currentGen[n4],cell node[100][100]);

int main()
{ system("COLOR F0");//n1=n3/2; n2=n1+1; n4=n3+1;
int i,h; float priority;

cell node[100][100];
TakeInput(node);

//solution start
```

```

    enter2: printf("\n\n Please enter the priority of criteria cost,
time and quality respectively:");
    printf("\n (Default Priority: Cost=1,Time=1,Quality=1)\n For
Default Priority press: 9977\n ");
    scanf("%f",&priority);
    if(priority==9977) goto enter3;
        pr1=priority;
        scanf("%f%f",&pr2,&pr3);
    enter3: printf("\n");
    printf("\nEnter the population size p \n(where p/2 is an even
number): ");
    scanf("%d",&n3);
    n1=n3/2; n2=n1+1; n4=n3+1;

    printf("\nEnter the percent chance of mutation: ");
    scanf("%d",&mut);

    chrom currentGen[n4], nextGen[n4];
    currentGen[0].fit=0;
    Generategens(currentGen,node);
    fitness(currentGen,node); //store fitness of each chromosome to
her body and store best chromosome at
// at currentGen[0].fit

    int ResistIteration=n77;
    float fitnessPrevious=0, fitnessNow=1;
    for(i=1;i<=500;i++)
    { if( (fitnessNow-fitnessPrevious)==0)
        ResistIteration=ResistIteration-1;
        if(ResistIteration==0) break;
        //printf("\nI AM HERE"); getch();
        fitnessPrevious=currentGen[0].fit;
        SelectNextGenPar(currentGen, nextGen);
        crossover(nextGen,node);
        mutation(nextGen);
        for(h=1;h<=n3;h++)
        {
            currentGen[h]=nextGen[h];
        }
        fitness(currentGen,node);
        fitnessNow=currentGen[0].fit;
    }

    printf("\n\n No. of Generation:%d",i-n77);
    gen=i-n77;
    printf("\n\n Maximum Fitness:%f",currentGen[0].fit);
    result(currentGen,node);
    return(0);
}

void *SelectNextGenPar(chrom currentGen[n4],chrom nextGen[n4])
{ //printf("\nNow in Select next gen parent ");

```



```

int a=0,b,c, e, h, i,j,k ; float totFit=0,cumFit=0, randN;
// finding total fitness// cumFit=cumulative fitness
for(i=1;i<=n3;i++)
{ totFit= totFit+currentGen[i].fit;
}
//getch();
nextGen[1]=currentGen[0];
for(i=2;i<=n1;i++)
{ randN=((float)rand()/RAND_MAX)*totFit;

cumFit=0;
for(j=1;j<=n3;j++)
{ cumFit=cumFit+currentGen[j].fit;

if(cumFit>=randN)
{ //printf("\nc%2.3f r%f", cumFit,randN);
nextGen[i]=currentGen[j];
break;

}

}
}
}

void *TakeInput(cell node[100][100])
{ int i,j,d, row=num, column=num, random, correction,
considerTime,considerCost,considerQuality;

printf("\n\n Do You Have Data Table(Press 1) or Want to Create New
One (Press 0)?:\n");
scanf("%d",&d);
if (d==1) {goto ReadData;}

/*130 Creating New data table
int v; //num=No of tasks/worker, v= element in the matrix
printf("Insert the Number of Tasks:");
scanf("%d",&num);
printf("\nEnter New file name to write data\n");
scanf("%s",&fname);

fp=fopen(fname,"w");
fprintf(fp,"Tasks_No:\t%d\n",num);

fprintf(fp,"\nCost_Matrix\n");
for(i=1;i<=num;i++)
{
for(j=1;j<=num;j++)
{
//scanf("%d",&v);
v=40+rand()%10;
fprintf(fp, "%d\t", v);
}
}
}

```

```

        }
        fprintf(fp, "\n");
    }

    fprintf(fp, "\nTime_Matrix\n");
    for(i=1; i<=num; i++)
    {
        for(j=1; j<=num; j++)
        {
            // scanf("%d", &v);
            v=30+rand()%10;
            fprintf(fp, "%d\t", v);
        }
        fprintf(fp, "\n");
    }

    fprintf(fp, "\nQuality_Matrix\n");
    for(i=1; i<=num; i++)
    {
        for(j=1; j<=num; j++)
        {
            //scanf("%d", &v);
            v=10+rand()%10;
            fprintf(fp, "%d\t", v);
        }
        fprintf(fp, "\n");
    }

    fclose(fp);
    printf("\n\nYour Document is ready, \n\n\t\tThank You\n\n");

//130*/

/*input data manually
    printf("\n\n Please enter the Table info viz cost, time and quality
    respectively\n");
    for(i=1; i<=column; i++)
    {   printf(" Column-%d: \n", i);
        for(j=1; j<=row; j++)
        {   if(pr1>0)
            scanf("%d", &node[j][i].cost);
            if(pr2>0)
                scanf("%d", &node[j][i].time);
            if(pr3>0)
                scanf("%d", &node[j][i].quality);
            printf("\n\n");
        }
    }
*/

/*112 read data from txt file
    ReadData: i=1;
    char s[50]; //s to store string

```

```

int a[50][50]; // num= Number of tasks

if (d==1)
{
    printf("\nEnter the file name to read data\n");
    scanf("%s",&fname);
    fp=fopen(fname,"r");
}
fp=fopen(fname,"r");
fscanf(fp,"%s",&s);    printf("%s\t",s);
fscanf(fp,"%d",&num);  printf("%d\n\n",num);

fscanf(fp,"%s",&s); // printf("%s\n",s); //take "Cost Matrix"
for(i=1;i<=num;i++)
{
    for(j=1;j<=num;j++)
    {
        fscanf(fp,"%d",&node[i][j].cost);
        // printf("%.2d\t",node[i][j].cost);
    }
    // printf("\n");
}

fscanf(fp,"%s",&s); // printf("\n%s\n",s); //take "Cost
Matrix"
for(i=1;i<=num;i++)
{
    for(j=1;j<=num;j++)
    {
        fscanf(fp,"%d",&node[i][j].time);
        //printf("%.2d\t",node[i][j].time);
    }
    // printf("\n");
}

fscanf(fp,"%s",&s); // printf("\n%s\n",s); //take "Cost
Matrix"
for(i=1;i<=num;i++)
{
    for(j=1;j<=num;j++)
    {
        fscanf(fp,"%d",&node[i][j].quality);
        // printf("%.2d\t",node[i][j].quality);
    }
    //printf("\n");
}
fclose(fp);

//112*/

//*/

```

```

printTable(node);

/*// making corrections if input has error
printf("\n Enter Column No and Row No For Correction\n (For No
Correction press: 9977 ) \n ");
scanf("%d",&correction);

for(;correction!=9977;)
{
    i=correction;
    scanf("%d",&j);
    printf(" Enter cost time and quality for
column=%d,row=%d\n",i,j);
    if(pr1>0)
        scanf("%d",&node[j][i].cost);
    if(pr2>0)
        scanf("%d",&node[j][i].time);
    if(pr3>0)
        scanf("%d",&node[j][i].quality);
    printTable(num,node);
    printf("\n Enter Column No and Row No For Correction\n (For No
Correction press: 9977 ) \n ");
    scanf("%d",&correction);
} // */

processTable(node);
}

void *printTable(cell node[100][100])
{
    int i,j, row=num, column=num;
    printf("\n\n Your table is: Cost/ Time/ Quality\n");
    printf("          mc1      mc2      mc3      mc4.....\n");
    for(i=1;i<=row;i++)
    {
        printf("wc%d:",i);
        for(j=1;j<=column;j++)
            printf("%7d",node[i][j].cost);
        printf("\n      ");
        for(j=1;j<=column;j++)
            printf("%7d",node[i][j].time);
        printf("\n      ");
        for(j=1;j<=column;j++)
            printf("%7d",node[i][j].quality);
        printf("\n\n");
    }
}

void *processTable(cell node[100][100])
{
    int i,j, row=num, column=num,
largestCost=1,largestTime=1,largestQuality=1;
    for(i=1;i<=row;i++)
        {
            for(j=1;j<=column;j++)
                {

```

```

        if(node[i][j].cost>largestCost)
            largestCost=node[i][j].cost;

    }
    for(j=1;j<=column;j++)
    {
        if(node[i][j].time>largestTime)
            largestTime=node[i][j].time;
    }
    for(j=1;j<=column;j++)
    {
        if(node[i][j].quality>largestQuality)
            largestQuality=node[i][j].quality;
    }
}
for(i=1;i<=row;i++)
{
    for(j=1;j<=column;j++)

node[i][j].ncost=(float)(node[i][j].cost)/(float)largestCost;
    for(j=1;j<=column;j++)

node[i][j].ntime=(float)node[i][j].time/(float)largestTime;
    for(j=1;j<=column;j++)

node[i][j].nquality=(float)node[i][j].quality/(float)largestQuality;
    }
    //printProcesTable(num,node);
}

void *printProcesTable(cell node[100][100])
{
    int i,j, row=num, column=num;
    printf("\n\n Your Normalized table is\n");
    printf("      mc1      mc2      mc3      mc4.....\n");
    for(i=1;i<=row;i++)
    {
        printf("wc%d:",i);
        for(j=1;j<=column;j++)
            printf("%7.3f",node[i][j].ncost);
        printf("\n      ");
        for(j=1;j<=column;j++)
            printf("%7.3f",node[i][j].ntime);
        printf("\n      ");
        for(j=1;j<=column;j++)
            printf("%7.3f",node[i][j].nquality);
        printf("\n\n");
    }
}

void *Generategens(chrom currentGen[n4],cell node[100][100])
{
    int h,i,j, value, random;
    //printf("\n\n First Generation is:");

```

```

for(h=1;h<=n3;h++)
{
    for(i=1;i<=num;i++)
    {
        random=rand();
        random=1+(random%num);
        recheck:
        for(j=1;j<=i;j++)
        {
            if(currentGen[h].bit[j]==random)
            {
                if(random==num)
                    random=1;
                else
                    random=random+1;
                goto recheck;
            }
        }
        currentGen[h].bit[i]=random;
    }
}
return(0);
}
void *fitness(chrom currentGen[n4],cell node[100][100])
{
    int i,j,h;

    float sumCost,sumTime,sumQuality;

    for(h=1;h<=n3;h++) // h=no of chromosome
    {
        sumCost=0.0;
        sumTime=0.0;
        sumQuality=0.0;
        for(i=1;i<=num;i++)//i= index for column or mc, j=row or
worker no.
        {
            j=currentGen[h].bit[i];
            sumCost=sumCost+node[j][i].ncost;
            //printf("\nh%d j%d i%d ncost
%3.3f",h,j,i,node[j][i].ncost);
            sumTime= sumTime+node[j][i].ntime;
            sumQuality=sumQuality+node[j][i].nquality;
        }
        currentGen[h].fit=1.0/(sumCost*pr1+sumTime*pr2+sumQuality*pr3);
    }

    // Store the best chromosome at currentGen[0]

    for(h=1;h<=n3;h++)
    {
        if ( currentGen[0].fit<currentGen[h].fit)
            currentGen[0]=currentGen[h];
    }
}
void *mutation(chrom nextGen[n4])
{
    //printf("\nNow in Mutation ");//getch();
}

```

```

int h, p,q, temp,decide;

decide=rand()%100;
if(decide<=mut) // % of mutation
{
    h=n2+rand()%n1;
    p=1+rand()%num;
    q=1+rand()%num;
    if(h==1) h=h+1;
    temp=nextGen[h].bit[p];
    nextGen[h].bit[p]=nextGen[h].bit[q];
    nextGen[h].bit[q]=temp;
    //printf("\n Mutation Ocured at chrom:%d    bit=%d
bit=%d",h,p,q); //getch();

}
}

void *crossover(chrom nextGen[n4],cell node[100][100])
{ int a,b,c, e, i,j,k, randN, p,q, m,n, h, count,d,
test2;
//printf("\nNow in crossover "); //getch();
for(i=n2;i<=n3;i++) // make all bit 0, initially
{
    for(j=1;j<=num;j++)
        nextGen[i].bit[j]=0;
    //nextGen[i].parent=0;
}
for(i=1;i<=n2;i++) // make parent bit 0, initially
{ nextGen[i].parent=0;
}
for(e=n2;e<n3;) //crossover start
{ c=e;
// set a; a is the 1st chromosome to met
randN=1+rand()%n1;
for(;;)
{if(nextGen[randN].parent==1)
    { if(randN==n1)
        { randN=1;
        continue;
        }
        randN=randN+1;
        continue;
    }

a=randN;
nextGen[a].parent=1;
break;
}
// set b , b is the 2nd chromosome to met
randN=1+rand()%n1;
for(;;)

```

```

    {if(nextGen[randN].parent==1)
      {
        if(randN==n1)
          {
            randN=1;
            continue;
          }
        randN=randN+1;
        continue;
      }
    b=randN;
    nextGen[b].parent=1;
    break;
  }
  // select the crossover point
  p=1+rand()%num;
  q=1+rand()%num;

  if(p>q)//Swappig so that p is less than q
  {
    p=p+q;
    q=p-q;
    p=p-q;
  }
  //printf("\nFor chrom%2.d and chrom%2.d):  p=%d,
q=%d",e,e+1,p,q);

  count=0;
  CC: count=count+1;
  for(i=p;i<=q;i++)
    nextGen[c].bit[i]=nextGen[b].bit[i];
  for(i=p;i<=q;i++)
  {
    for(j=p;j<=q;j++)
      {
        if(nextGen[a].bit[i]==nextGen[b].bit[j])
          {
            // printf("\nTest i=%d",i);
            goto BB;
          }
      }
    k=i;
    AA:
    for(m=0;m<=num;m++)
      {
        if(nextGen[a].bit[m]==nextGen[b].bit[k])
          {
            if(m<p||m>q)
              {
nextGen[c].bit[m]=nextGen[a].bit[i];
//printf("\nTest2 i=%d",i);
              }
            else
              {
                k=m;
                goto AA;
              }
          }
      }
    }
  BB: test2=8;

```



```

    }
    for(i=1;i<=num;i++)
        if(nextGen[c].bit[i]==0)
            nextGen[c].bit[i]=nextGen[a].bit[i];

    if(count==1)
        {d=a;a=b;b=d;c=c+1; goto CC; }//swap a,b. c go to next
currentGen
    e=e+2;
    }
}
void *sort(chrom currentGen[n4])
{
    int e, i,h;
    chrom temp;

    for(e=1;e<=n3;e++)
    {
        for(i=n3;i>=2;i--)
        {
            if(currentGen[i].fit>currentGen[i-1].fit)
            {
                temp=currentGen[i];
                currentGen[i]=currentGen[i-1];
                currentGen[i-1]=temp;
            }
        }
    }
}

void *result(chrom currentGen[n4],cell node[100][100])
{
    int i,j,h, sumCost=0,sumTime=0,sumQuality=0;

    printf("\n\n\nResult:\nBest combination is:");
    printf(" T1 T2 T3 T4 .. .. .. \n Worker
No:");
    for(i=1;i<=num;i++)
        printf("%4.0d ",currentGen[0].bit[i]);
    //printf("\n\nTotal Cost:");
    for(i=1;i<=num;i++)//i=column or for a mc, gene=row or worker
no.
        {
            j=currentGen[0].bit[i];
            sumCost=sumCost+node[j][i].cost;
            //printf("\nh%d j%d i%d ncost
%3.3f",h,j,i,node[j][i].ncost);
            sumTime= sumTime+node[j][i].time;
            sumQuality=sumQuality+node[j][i].quality;
        }
    printf("\n\n Total Cost: %d \n Total Time: %d\n Total
Quality:%d",sumCost,sumTime,sumQuality);
    printf("\n\n\n .....The
End.....\n\n");

    //111 write some information to data file for future

```

```

fp=fopen(fname,"a");
fprintf(fp,"\n\nResult:\nNumber of task is:%d",num);
fprintf(fp,"\npriority of cost, time and quality are:%.2f,
%.2f and %.2f",pr1,pr2,pr3);

fprintf(fp, "\nMutation percent chance:%d\n",mut);

fprintf(fp,"\nBest combination is:");
fprintf(fp," T1 T2 T3 T4 .. .. .. \n
Worker No:");
for(i=1;i<=num;i++)
    fprintf(fp,"%4.0d ",currentGen[0].bit[i]);

fprintf(fp,"\nTotal Cost: %d \nTotal Time: %d\nTotal
Quality:%d",sumCost,sumTime,sumQuality);
fprintf(fp,"\nPopulation size: %d",n3);
fprintf(fp,"\nNo. of Generation:%d",gen);

fclose(fp);
//111 fprintf(fp,"\n");
}

```

--The End--