MULTIOBJECTIVE GENETIC ALGORITHM APPROACHES TO PROJECT SCHEDULING UNDER RISK

by
MURAT KILIÇ

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
Spring 2003

MULTIOBJECTIVE GENETIC ALGORITHM APPROACHES TO PROJECT SCHEDULING UNDER RISK

APPROVED BY:

Prof. Dr. Gündüz Ulusoy …….

(Thesis Supervisor)

Associate Prof. Dr. Funda Sivrikaya Şerifoğlu …….

Assistant Prof. Dr. Bülent Çatay ……...

DATE OF APPROVAL: 18.08.2003

ACKNOWLEGEMENTS

I am extremely grateful to my thesis supervisor, Prof. Gündüz Ulusoy, for his continuous support, encouragement, guidance and invaluable comments. This study would be impossible without his comments and guidance.

I would like to thank my thesis committee members, Associate Prof. Dr. Funda Sivrikaya Şerifoğlu and Assistant Prof. Dr. Bülent Çatay for their comments, their time spent on my thesis and serving on my thesis committee.

I would like to thank my friend Ercan Erol for his support during the software development process; his support enabled me to develop the software.

I would like to thank my office mates Ekim Özaydın and Gülay Arzu İnal for their support. I want to send my special thanks to Nuri Mehmet Gökhan for his encouragement, endless support and comments. His comments helped the development of this study most of the time.

Finally, I would like to thank my family and Helin El whose support keep me alive.

ABSTRACT

MULTIOBJECTIVE GENETIC ALGORITHM APPROACHES TO PROJECT SCHEDULING UNDER RISK

In this thesis, project scheduling under risk is chosen as the topic of research. Project scheduling under risk is defined as a biobjective decision problem and is formulated as a 0-1 integer mathematical programming model. In this biobjective formulation, one of the objectives is taken as the expected makespan minimization and the other is taken as the expected cost minimization.

As the solution approach to this biobjective formulation genetic algorithm (GA) is chosen. After carefully investigating the multiobjective GA literature, two strategies based on the vector evaluated GA are developed and a new GA is proposed. For these three GAs first the parameters are investigated through statistical experimentation and then the values are decided upon. The chosen parameters are used for the computational study part of this thesis.

In this thesis three improvement heuristics are developed also to further improve the GA solutions. The aim of these improvement heuristics is to decrease the expected cost of the project while keeping the expected duration of the project fixed. These improvement heuristics are implemented at the end of the proposed GA and used to improve the results of the proposed GA.

Finally the GAs and improvement heuristics are tested on three different sets of problems. The results are evaluated by pairwise comparisons of algorithms and of heuristics. Also an approximation of the true Pareto front is generated using the commercial mathematical modelling program, GAMS©. The results are compared to that approximation and they seem comparable to that solution. The results of the improvement heuristics are also compared against each other and the performance of the heuristics is reported in detail.

ÖZET

RİSK ALTINDA PROJE ÇİZELGELEME PROBLEMİNE GENETİK ALGORİTMA ÇÖZÜM YAKLAŞIMLARI

Bu tezde risk altında proje çizelgeleme problemi ele alınmıştır. Risk altında proje çizelgeleme problemi iki amaçlı karar problemi olarak tanımlanmış ve 0-1 tamsayılı matematiksel programlama modeli olarak formüle edilmiştir. İki amaçlı bu modelde, bir amaç beklenen proje süresinin en küçüklenmesi diğer amaç ise beklenen proje maliyetinin en küçüklenmesidir.

Bu probleme çözüm yaklaşımı olarak genetik algoritma (GA) seçilmiştir. Çok amaçlı GA literatürü detaylı olarak incelendikten sonra vektör değerlendirmeli GA üzerine iki strateji ve ayrıca yeni bir GA önerilmiştir. Bu GAlar için parametreler üzerinde yapılan istatistiki deneyler sonucunda uygun parametre değerleri seçilmiştir. Seçilen parametreler yapılan çalışmalarda kullanılmıştır.

Bu tezde ayrıca GA sonuçlarını geliştirmek üzere üç tane sezgisel yöntem önerilmiştir. Bu sezgisel yöntemlerin amacı, beklenen proje süresini sabit tutarken beklenen proje maliyetini azaltmaktır. Sezgisel yöntemler önerilen GA'nın sonuna eklenmiş ve bu algoritmanın sonuçlarını geliştirmek amacıyla kullanılmıştır.

Son olarak, GAlar ve sezgisel yöntemler üç farklı problem sınıfı üzerinde sınanmıştır. Sonuçlar üzerinden algoritmaların ve sezgisel yöntemlerin ikili karşılaştırmaları yapılmıştır. Ayrıca GAMS© ticari matematiksel programlama yazılımı kullanılarak Pareto yüzeyinin bir yaklaşımı yapılmıştır. Önerilen GA'nın sonuçlarının bu yaklaşımla da yakın olduğu görülmüştür. Sezgisel yöntemlerin ise ikili karşılaştırması yapılmış ve bu karşılaştırmaların sonuçları rapor edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION AND PROBLEM DEFINITION

The aim of this thesis is to develop an effective solution to the problem of project scheduling under risk. Project scheduling under risk has not been studied extensively in the literature (Ulusoy, 2002). The model for project scheduling under risk can be summarized as follows.

Each task (activity) contains different number of risks and each risk has an impact and a probability of occurrence associated with it. Risks only affect the duration of the related task when they occur. A project manager can decrease the probability of occurrence and impact of each risk by taking some preventive measures. These preventive measures have a cost. A penalty cost based on the tardiness of the project, an overhead cost based on the project duration and a labor cost based on the daily labor needs of each task are the components of the cost function of the model. The model has no resource constraints. The risks are assumed to be independent with their impacts being additive.

There are a number of objectives in project scheduling and most project managers are trying to achieve more than one objective simultaneously. Hence, multiobjective approach to this problem has been adopted in this thesis. Makespan minimization and cost minimization objectives are chosen as the two objectives to be adopted by the decision maker.

Chapter 2 of the thesis summarizes the basic concepts of the deterministic project scheduling problem elements. Chapter 3 of the thesis summarizes the basics of multiobjective optimization and introduces the multiobjective evolutionary algorithms. Chapter 4 explains the problem and the proposed solution approaches. Chapter 5 gives the details of the computational study and the results of this study. Chapter 6 includes the conclusion and the proposed future research directions.

2. DETERMINISTIC PROJECT SCHEDULING

2.1. Elements of Project Scheduling Problem (PSP)

2.1.1. Activities

Activities are non-divisible parts of project. Activities are also called as jobs, operations and tasks. Each activity must be completed in order to finish the project. Activities may have modes, which determine duration, resource and cash flows.

2.1.2. Precedence Relations

For some reason, some tasks may need a set of tasks to be completed in order to start. For example, these precedence relations may occur according to technological requirements. Consider, e.g., a building project. Clearly, activity “roof tiling” may only be started if another activity “erecting walls” has been finished. The precedence relations are given by sets of immediate predecessors indicating that an activity may not be started before each of its predecessors is completed (Hartmann, 1999).

Also some activities may have some other type of precedence relations. To handle these situations generalized precedence relations (GPRs) are defined. These are named as start-start (SS), finish-finish (FF), finish-start (FS) and start-finish (SF). Minimal time lag and maximal time lag are other features to describe the precedence relationship between two or more tasks.

Most of the time, the statement of the project is in the form of a set of activities and the immediate precedence relations among them. If activity u precedes activity v, it is written as $u \prec v$ (Elmaghraby, 1995).

In some cases GPRs are also used to define the relationship between two activities. While defining GPRs start times of activities (i.e. start time of activity a $s(a)$) and finish times of activities (i.e. finish time of activity b $f(b)$) must be identified. Some examples of GPRs can be stated as follows:

$s(b) \geq s(a) + 1$ (SS; denotes, b can start one time unit after a starts)

$s(b) \geq f(a) + 3$ (FS; denotes, b can start three time unit after a finishes)

$f(b) \geq f(a) + 5$ (FF; denotes, b can finish five time units after a finishes)

$f(b) \geq s(a) + 2$ (SF; denotes, b can finish two time units after a starts)

2.1.3. Resources

Material, money, manpower, which are needed to perform the tasks of the project are called the resources. Resources are very important in project scheduling since they define the type of the problem. If at least one of the resources is constrained, the problem is called resource-constrained project scheduling problem (RCPSP). Resources are mostly classified according to category. Category based classification includes four type of classes, which are renewable, nonrenewable, doubly constrained and partially renewable classes (Kolisch and Padman, 2001).

2.1.3.1. Renewable Resources

Renewable resources are constrained on a period basis only. That is, regardless of the project length, each renewable resource is available for every single period. Examples of this class are machine, manpower and equipment.

2.1.3.2. Nonrenewable Resources

Nonrenewable resources are limited over the entire planning horizon, with no restrictions within each period. The classic example is the budget of a project.

2.1.3.3. Doubly Constrained Resources

Doubly constrained resources are constrained both on the period and planning horizon basis. Budget constraints that limit capital availability for the entire project as well as limiting its consumption over each time period are an example of this type of resource.

2.1.3.4. Partially Renewable Resources

Partially renewable resources limit the utilization of some resources within a subset of planning horizon. An example is that of a planning horizon of a month with workers whose weekly working time, not the daily time, is limited by the working contract. It has been shown that partially renewable resources can depict both renewable and nonrenewable resources.

2.2. Objectives Employed in Project Scheduling Problems

2.2.1. Makespan Minimization

In this type of PSP, the objective is to minimize the makespan (i.e. the time span between the starting time and the ending time of the project). The solution of this type of problems generates a time-critical path.

2.2.2. Net Present Value Maximization

Maximization of the net present value (NPV) of cash flows throughout the project is taken as the objective in these types of problems. Expenses and payments are types of cash flows and the timing of these cash flows occur depending on contract types. For example, expenses might be paid at the beginning of tasks and progress payments might occur at the end of a defined set of tasks.

2.2.3. Quality Maximization

Maximizing the quality of the project is one of the more important objectives for project managers. That is why quality maximization is also taken as an objective in PSPs. The problem with this objective is its quantitative definition and the agreement of different stakeholders on this definition.

2.2.4. Cost Minimization

In the type of problems with this objective, costs such as those occurring from the realization of an activity, resource usage and earliness / tardiness penalties are to be minimized.

Besides these well-known objectives, some other objectives are also employed. These performance measures are represented based on timing of activities. Some examples of such objectives are "minimizing the total earliness of activities" and "minimizing the total tardiness of activities". The combinations of these objective functions are also employed in project scheduling leading to multiobjective project scheduling problems. In this thesis, *cost minimization* and *makespan minimization* are chosen as the objectives to achieve.

2.3. Network Representation of Projects

In general, two representations, activity-on-arc (AOA) and activity-on-node (AON), have been commonly used to capture project networks, resulting in an event-based or activity-based representation, respectively. In the AOA representation, nodes represent events and arcs represent activities. Dummy activities are used to preserve the precedence relations and dummy nodes capture the start and completion of the project. In the AON representation, activities are represented by nodes and precedence relations are represented by directed arcs (Kolisch and Padman, 2001).

In Figure 2.1(a) and (b), AOA and AON representations of a project, which has four activities (*a,b,c,d*) and the following precedence relations are illustrated respectively.

$\varnothing \prec a, b; \quad a \prec c, d; \quad b \prec d; \quad c, d \prec \varnothing .$

(a) (b)

Figure 2-1 (a) The AON representation; (b) AOA representation.

The AON representation of a project is more direct, more frugal and unique. On the other hand, AOA representation has some advantages against AON representation. These advantages can be summarized from two points of view.

From representational point of view, it is easy to graphically identify the events of the project in AOA representation. It is easier to visually identify the finished activities up to occurrence of an event. Finally, AOA representation is preferred when it is desired to give a visual representation of the duration of the activities, and then the arc length is made proportional to the duration of the activity (Elmaghraby, 1995).

From analytical point of view, it is easy to capture the information of more complex precedence relationships such as generalized precedence relationships. AOA type of representation is also advantageous when one tries to construct mathematical models that depend on the definition of nodes, such as linear models for optimal time-cost trade-off.

3. MULTIOBJECTIVE OPTIMIZATION PROBLEM

Most real world problems have multiple objectives to achieve. This situation creates a set of problems in Operations Research (OR) called multiobjective optimization problems (MOPs). In order to deal with MOPs, plenty of techniques have been developed in OR. Many approaches have been suggested, going all the way from naively combining objectives into one to the use of game theory to coordinate the relative importance of each objective. The fuzziness of this area lies in the fact that there is no accepted definition of "optimum" as in the single-objective optimization. Hence, it is difficult to even compare the results of one method to another method's results because, normally, the "best" answer corresponds to the most preferable solution by the so-called *decision maker* (DM) (Coello, 2000).

3.1. Statement of the Multiobjective Optimization Problem (MOP)

Multiobjective (also called multiperformance, multicriteria or vector) optimization can be defined as the problem of finding a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term "optimize" means finding such a solution which would give the values of all the objective functions acceptable to the designer (Coello, 2000).

Formally, we can state the problem as follows (in this thesis, if not otherwise stated, all the objectives of MOP are taken as minimization):

$$Min\ f(X) = \left[f_1(X), f_2(X), ..., f_k(X) \right]^T \tag{3.1}$$

subject to:

$$g_i(X) \geq 0 \qquad i = 1, 2, ..., m \tag{3.2}$$

$$h_i(X) = 0 \qquad i = 1, 2, ..., p \tag{3.3}$$

where $X=[X_1, X_2,\ldots, X_n]^T$ is the n dimensional vector of decision variables and T stands for the transpose. In the formulation, k represents the number of objectives, m is the number of inequality constraints and p is the number of equality constraints.

Some terms need to be defined to further investigate the MOP.

3.1.1. Ideal Vector and Ideal Decision Vector

Assume that we have k objective functions $f_i(X)$ (i=1,2,…,k) which can be solved on the decision vector space X separately. Let f_i^0 be the optimum for the ith objective. The decision vector $X^{0(i)}$ corresponding to this solution is denoted by:

$$X^{0(i)} = \left[X_1^{0(i)}, X_2^{0(i)}, ..., X_n^{0(i)} \right]^T \tag{3.4}$$

where $X_j^{0(i)}$ is the decision variable (j=1,2,…,n) of $X^{0(i)}$.

For this multiobjective problem, set of optimum solutions constitutes a vector of optimum solution values (f^0) in k dimensional space and this vector is called the *ideal vector*.

$$f^0 = \left[f_1^0, f_2^0, ..., f_k^0 \right]^T \tag{3.5}$$

The solution vector corresponding to this ideal set of solutions called the *ideal decision vector*.

3.1.2. Pareto Optimum

X^* is Pareto optimal, if there exists no feasible vector X that decreases some criterion without causing a simultaneous increase in at least one other criterion. Formally, X^* is Pareto optimal, if for every $X \in F$ (where F denotes the feasible region of the problem), either (Coello, 2000)

$$\bigwedge_{i \in \{1,...k\}} (f_i(X) = f_i(X^*)) \tag{3.6}$$

or there is at least one $i \in \{1,\ldots k\}$ such that

$$f_i(X) > f_i(X^*). \tag{3.7}$$

where $\wedge$ means some.

Another formal description can be given as follows; X^* is Pareto optimal if there is no $X \in F$ such that (Ehrgott, 2000)

$$f_i(X) \leq f_i(X^*) \qquad \text{for } i = 1,2,\ldots k \tag{3.8}$$

and

$$f_j(X) < f_j(X^*) \qquad \text{for some } j \in \{1,2,\ldots k\} \tag{3.9}$$

The set of Pareto optimum solutions is called the set of *noninferior* or *nondominated* solutions, also called the Pareto set.

3.1.3. Pareto Front

Pareto front is the union of all nondominated solutions of the problem. For example, in a biobjective problem if the problem is solveable in the continuous domain Pareto front would be a continuous curve. Most of the time it is not possible to find an analytical representation of the Pareto front. In such a case, an adequate number of solutions are calculated to represent the Pareto front through a discrete set of points. Figure 3.1 demonstrates the concept of Pareto front in a biobjective problem, where the Pareto front is marked with a bold line.

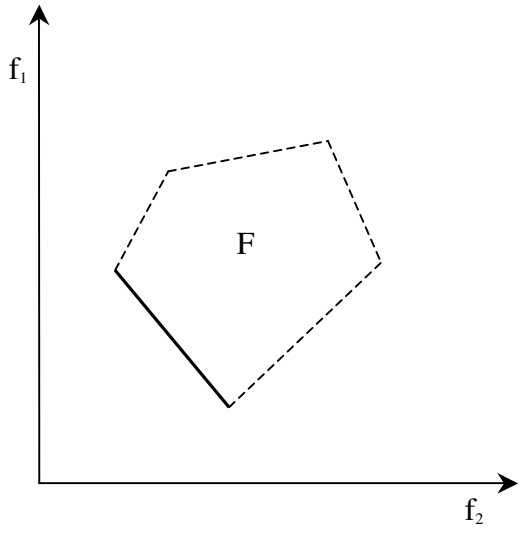

Figure 3-1 Pareto front of a biobjective problem

3.2. Multiobjective Optimization

Multiobjective optimization techniques, as it is mentioned, typically result in more than a single solution. For this reason, to decide on the optimum, we need a DM who is capable of choosing the right solution from the set of solutions. This selection is one of the most challenging activities in multiobjective optimization. Three types of multiobjective optimization solution technique are available depending on the timing of the DM's selection.

Priori Preference Articulation: DM combines the differing objectives into a scalar cost function. This effectively makes the MOP singleobjective prior to optimization.

Progressive Preference Articulation: Decision making and optimization are intertwined. Partial preference information is provided upon which optimization occurs, providing an "updated" set of solutions for the DM to consider.

Posteriori Preference Articulation: DM is presented with a set of Pareto optimal candidate solutions and chooses from that set (Van Veldhuizen and Lamont, 2000a).

3.2.1. Weighted Sum Approach

This method consists of adding all the objective functions together using weighting coefficients for each one. As a result, the multiobjective optimization problem is transformed into a scalar optimization problem and the problem is represented in the following form (Coello, 2000).

$$Min \sum_{i=1}^{k} w_i f_i(X) \tag{3.10}$$

subject to:

$$g_i(X) \geq 0 \qquad i = 1, 2, ..., m \tag{3.11}$$

$$h_i(X) = 0 \qquad i = 1, 2, ..., p \tag{3.12}$$

where $w_i \geq 0$ are the weighting coefficients.

It is usually assumed that $\sum_{i=1}^{k} w_i = 1$. But these weighting coefficients do not proportionally reflect the relative importance of the objectives, but are only factors which, when varied, locate points in the Pareto set.

If we want w_i to closely reflect the relative importance of the objective functions, we need to normalize the objective functions. This normalization is achieved by using a multiplier c_i ($c_i = 1/f_i^o$). After this normalization, the objective function becomes:

$$\min \sum_{i=1}^{k} w_i f_i(X) c_i \tag{3.13}$$

subject to constraints as represented in Equations 3.11 and 3.12.

3.2.2. Goal Programming

In goal programming (GP), DMs have to assign targets or goals (b_i) that they wish to achieve for each objective. Then, these b_i values and the associated objectives are used to form a constraint. In order to represent the constraints in equality form, the positive (n_i) and the negative (p_i) deviation variables are added to constraints. Thus the problem is transformed to the following form:

$$Min \sum_{i=1}^{k} (n_i + p_i) \tag{3.14}$$

subject to:

$$f_i(X) + n_i - p_i = b_i \qquad i = 1, 2, ...k \tag{3.15}$$

$$X \in F \qquad n \geq 0 \qquad p \geq 0$$

The aim in GP is to minimize the deviations between the achievements of the goals. The achievement process can be accomplished with different methods. Each one of these methods leads to a GP variant. Three variants, weighted goal programming (WGP), lexicographic goal programming (LGP) and MINMAX GP are mentioned below (Romero, 1991).

3.2.2.1. Weighted Goal Programming

In WGP, different than GP the objective function is generated from the sum of weighted deviations. To form the objective function of the WGP, the DM must assign different weights to the negative and positive deviations. After these additions to the GP, the objective function for WGP becomes the following, where the other constraints remain the same as in GP.

$$Min \sum_{i=1}^{k} (\alpha_i n_i + \beta_i p_i) \tag{3.16}$$

Obviously, the weights β will be zero when the desired achievement of the goal is greater than the established target. Similarly, the weights α will be zero when the desired achievement of the goal is less than the established target.

3.2.2.2. Lexicographic Goal Programming

In LGP, the DM generates a lexicographic objective function that has an importance ranking of objective function deviations. At each phase of LGP solution, the element of lexicographic objective function at this rank tried to be achieved.

The lexicographic objective function of the general MOP is as follows (assuming that lexicographic objective function has *q* elements).

$$Lex\ Min\ a = \left[h_1(n, p), h_2(n, p), \ldots h_q(n, p) \right] \tag{3.17}$$

The LGP is solved through multi-phase approach. At first step the first element is minimized, at this level some variables are fixed and then second model is solved. This operation goes until the solution of q models has been made. If there are resources in the problem, the solution process may stop when the resources are exhausted. The model for solution's first step is given below as an example.

First Step Model of Solution:

$$Min\ h_1(n, p) \tag{3.18}$$

subject to:

$$f_i(X) + n_i - p_i = b_i \qquad i = 1,2,\ldots k \tag{3.19}$$

$$X \in F \qquad n \geq 0 \qquad p \geq 0$$

3.2.2.3. MINMAX Goal Programming

In this GP variant, the aim is to minimize the upper level of total weighted deviation for all of the objectives. The following model summarizes the aim of the MINMAX GP at a glance.

$$Min \quad d \tag{3.20}$$

subject to:

$$\alpha_i n_i + \beta_i p_i \le d \qquad i = 1,2,\ldots k \tag{3.21}$$

$$f_i(X) + n_i - p_i = b_i \qquad i = 1,2,\ldots k \tag{3.22}$$

$$X \in F \qquad n \ge 0 \qquad p \ge 0$$

3.2.3. Goal Attainment

In this approach, DM decides on two vectors: The weight vector $w = [w_1, w_2, ..., w_k]$ and the goal vector $b = [b_1, b_2, ..., b_k]$. To find the best compromise solution X^*, we solve the following problem:

$$Min\ \alpha \tag{3.23}$$

subject to:

$$g_j(X) \le 0 \qquad j = 1,2,\ldots,m \tag{3.24}$$

$$h_l(X) = 0 \qquad l = 1,2,\ldots,p \tag{3.25}$$

$$b_i + \alpha w_i \ge f_i(X) \qquad i = 1,2,\ldots,k \tag{3.26}$$

where α is a scalar variable and is unrestricted in sign. The weights are positive and are normalized as follows:

$$\sum_{i=1}^{k} w_i = 1 \tag{3.27}$$

Figure 3.2 describes how this approach behaves in the context of a biobjective problem. It is obvious from the Figure 3.2, that the solution to the MOP by goal attainment approach occurs at the intersection point of the feasible region and the sum vector (Coello, 2000).

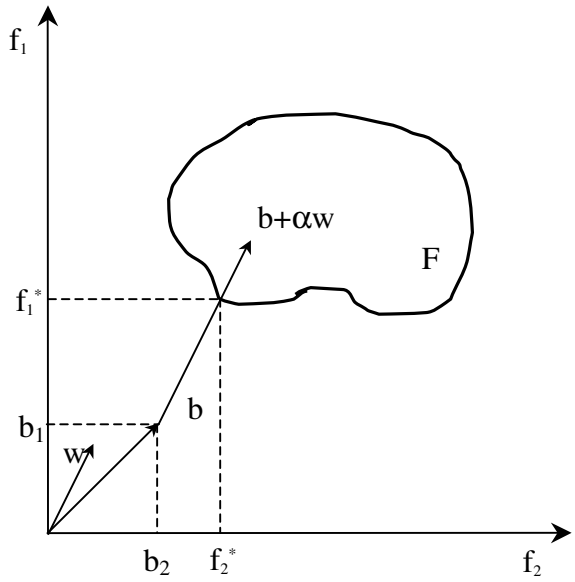

Figure 3-2 Goal attainment approach sample graph (Coello, 2000)

3.2.4. The ε-Constraint Method

This method is based on minimizing one (the most preferred or primary) objective function and considering the other objectives as constraints bound by some allowable levels ε_i. The method may be formulated as follows:

1) Find the minimum of the r^{th} objective function, i.e., find X^* such that

$$f_r(X^*) = \underset{X \in F}{Min}\, f_r(X) \tag{3.28}$$

subject to additional constraints of the form

$$f_i(X) \le \varepsilon_i \qquad \text{for } i=1,2,\ldots,k \quad \text{and} \quad i \ne r \tag{3.29}$$

where ε_i are assumed values of the objective functions, which we do not wish to exceed.

2) Repeat step (1) for different values of ε_i. The information derived from a well-chosen set of ε_i can be useful in making the decision. The search is stopped when the decision maker finds a satisfactory solution.

It may be necessary to repeat the above procedure for different indices of r (Quagriella and Vicini, 1998).

3.2.5. Genetic Algorithm Based Solution Approaches to MOP

GA solution approach to multiobjective optimization is one of the most widely used in the OR literature. GAs constitute approximately 70% of the metaheuristic approaches published between 1991 and 2000 (Jones *et al.* 2002).

There are a large number of GA based solution approaches for MOPs. These approaches will be summarized in the following sections.

3.2.5.1. Vector Evaluated Genetic Algorithm

Vector evaluated genetic algorithm (VEGA) is the first algorithm which is presented to solve MOPs. In this algorithm, k subpopulations of (N/k) individuals are created where N is the total population size and k is the number of objectives. An individual in subpopulation j is evaluated according to the performance on j^{th} objective function to form its fitness value. After this step all the individuals in sub-populations are shuffled together and genetic operators are applied to these to form the next generation. VEGA is demonstrated in Figure 3.3 for a better understanding of the algorithm.

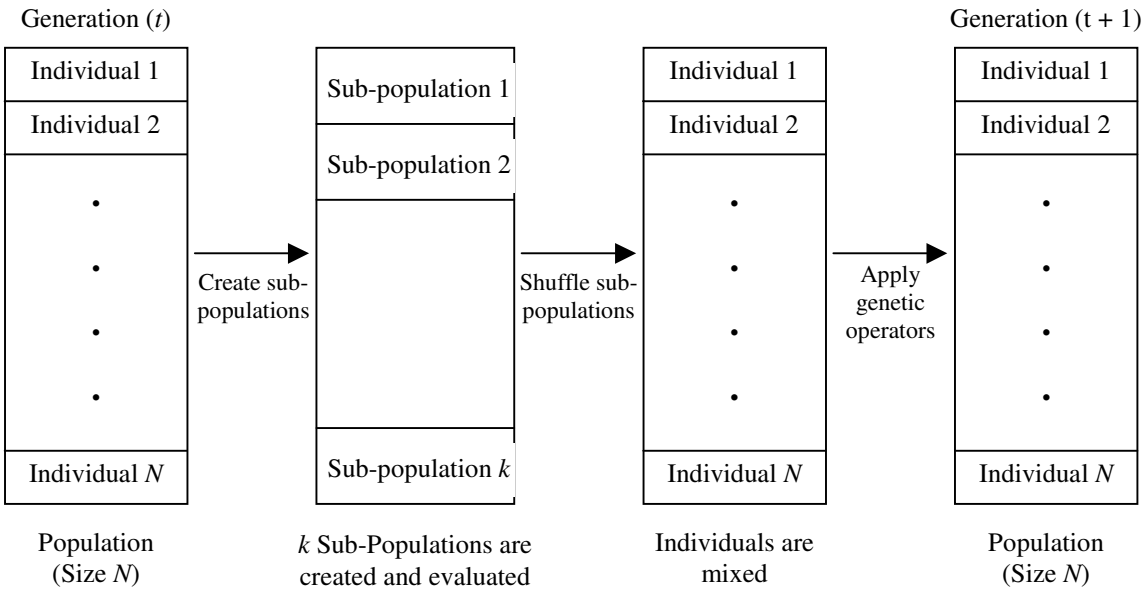

Figure 3-3 Schematic of VEGA selection (Coello, 2000)

VEGA is an easy algorithm to implement. On the other hand, it has some problems. This problem is speciation, which is described as “the evolution of species

within a population that excels in some respect" in genetics. This problem arises because this technique selects individuals who excel in one dimension without looking at other dimensions. The potential danger is that we could evolve with middling performance individuals. Middling implies an individual with acceptable performance, perhaps above average in all objectives, but not outstanding when measured by any particular function. Speciation is undesirable because it is opposed to goal of finding a compromise solution (Coello, 2000).

In some GAs genders are also used to model the subpopulation based fitness assignment of VEGA. In these algorithms, each individual is assigned one of the k different genders at initial population. Fitness values of the individuals are calculated according to their genders just as in VEGA. For mating, sexual attractors are used to model the sexual attraction that occurs in nature. The mutation operator is restricted only slightly, to avoid changes in the sex of an individual. The reproduction operator does not change the sex of the individual that is copied (Coello, 2000).

3.2.5.2. Nash Genetic Algorithms: Noncooperative Approach

For an optimization problem with k objectives, a Nash strategy consists of k players, each optimizing its own criterion. However, each player has to optimize his criterion given that all the other criteria are fixed by the rest of the players. When no player can further improve its criterion, it means that that the system reached a state of equilibrium called Nash equilibrium. For a biobjective problem, let E be the search space for the first criterion and W the search space for the second criterion. A strategy pair $(X,Y) \in ExW$ is said to be a Nash equilibrium if and only if:

$$f_E(X,Y) = \inf_{X \in E} f_E(X,Y) \tag{3.30}$$

$$f_W(X,Y) = \inf_{Y \in W} f_W(X,Y) \tag{3.31}$$

where *inf* means inferior or nondominated.

Figure 3.4 describes how this approach works in the context of a biobjective problem.

It is obvious that exchanges between players must be as frequent as possible to speed up the convergence of the algorithm (Périaux *et al.*, 1998).

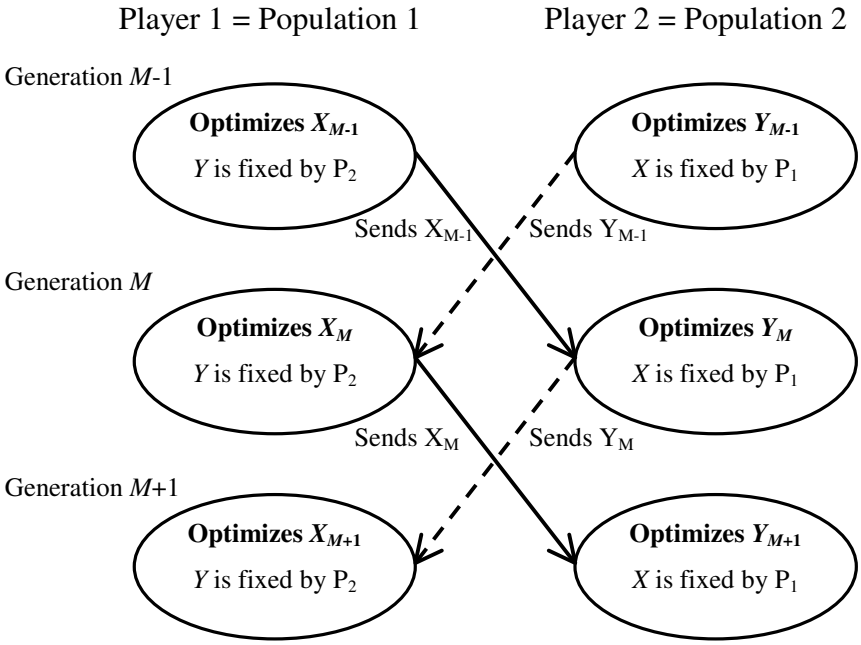

Figure 3-4 Noncooperative Nash genetic algorithm (Périaux *et al.*, 1998)

3.2.5.3. Weighted Min-Max Approach Based GA

In this approach, the first generation is generated randomly. Chromosomes are formed to represent a solution and a corresponding weight list for objectives. For each generation min-max optimum solution procedure, described below, is processed.

A point X^* is min-max optimal, if for every X (where $X \in F$) the following recursive formula is satisfied.

Step 1:

$$v_1(X^*) = \underset{X \in F}{Min} \underset{i \in I}{Max} \{z_i(X)\} \tag{3.32}$$

and then $I_1 = \{i_1\}$, where i_1 is the index for which the value of $z_i(X)$ is maximal where $z_i(X)$ is described as follows.

$$z_i'(X) = \frac{\left| f_i(X) - f_i^o \right|}{\left| f_i^o \right|} \tag{3.33-a}$$

$$z_i''(x) = \frac{\left| f_i(X) - f_i^o \right|}{\left| f_i(X) \right|} \tag{3.33-b}$$

where $z_i'(X)$ and $z_i''(X)$ are relative deviations from the objectives' optimum value and $z_i(X)$ found from the formula below.

$$\forall_{i\in I}(z_i(X)) = Max\{z_i^{'}(X), z_i^{''}(X)\} \tag{3.34}$$

If there is a set of solutions $X_1 \subset F$ that satisfies Step 1, then apply:

Step 2:

$$v_2(X^*) = \underset{X\in X_1}{Min}\ \underset{i\in I, i\notin I_1}{Max}\{z_i(X)\} \tag{3.35}$$

and then $I_2 = \{i_1, i_2\}$, where i_2 is the index for which the value of $z_i(X)$ in this step is maximal.

After the intermediate steps the k^{th} step is as follows.

Step k:

$$v_k(X^*) = \underset{X\in X_{k-11}}{Min}\ \underset{i\in I, i\notin I_{k-11}}{Max}\{z_i(X)\} \tag{3.36}$$

where $\{v_1(X^*), \ldots, v_k(X^*)\}$ is the set of optimal values of fractional deviations ordered nonincreasingly.

After this solution procedure is employed for all of the chromosomes, the following utility function U is used to evaluate the fitness of the chromosomes.

$$U = \sum_{i=1}^{k} W_i \frac{F_i}{F_i^*} \tag{3.37}$$

where F_i^* are the scaling parameters for the objective criterion, k is the number of objective functions and W_i are the weighting factors for each objective function F_i.

In this approach, a sharing function with the form below is also used.

$$\phi(d_{ij}) = \begin{Bmatrix} 1-\left(\dfrac{d_{ij}}{\sigma_{sh}}\right)^{\alpha}, & d_{ij} < \sigma_{sh} \\ 0, & otherwise \end{Bmatrix} \tag{3.38}$$

where normally α=1, d_{ij} is a metric indicative of the distance between designs i and j, and σ_{sh} is the sharing parameter that controls the extent of sharing allowed. The fitness of a chromosome i is then modified to

$$f_{s(i)} = \frac{f_i}{\sum_{i=1}^{M}\phi(d_{ij})} \tag{3.39}$$

where M is the number of chromosomes located in the vicinity of the i^{th} chromosome.

The performance of the algorithm is closely related to the parameter values that are chosen. The authors use α=1 and chose a value between 0.01 and 0.1 for σ_{sh}.

Finally a mating restriction is used not to make crossover between chromosomes within a certain radius. It is also suggested not to make crossover between individuals in

a radius of 0.15 (σ_{mat}=0.15 where σ_{mat} represent the radius of mating restriction) (Coello, 2000).

3.2.5.4. Two Variations of the Weighted Min-Max Strategy

These two variations of Min-Max based approach are given as parts of Multiobjective Optimization of Systems in the Engineering Sciences (MOSES) by Coello and Christiansen (1999). First of these variants is described by the following steps.

1. The initial population is formed such a way that none of the individuals are infeasible.
2. The user should give a list of weights for *k* objectives and a generation is solved by the min-max optimum approach. For each of the weight lists provided by the user, a generation is solved and the best compromise solution is selected to list for the DM. Different from the weighted Min-Max based GA in this variant the weights are not coded as a part of the chromosomes, they are given by the user for each generation.
3. After the *n* processes are employed (*n*=number of weight combinations provided by the user, also number of generations), a final file is generated for the DM containing *n* best results.

This algorithm uses crossover and mutation, which are not restricted to give only feasible solutions. If an operator (crossover or mutation) gives an infeasible solution, it is replaced by one of its parents.

Second variant employed in MOSES can be summarized by the similar following steps. Different from the first variant the second variant uses sharing and binary tournament selection.

1. The initial population is formed such a way that none of the individuals are infeasible.
2. By exploring the population at each generation, the local ideal vector is produced. This is done by comparing the values of each objective function in the entire population.
3. The binary tournament selection is done by comparing the two individuals with the local ideal vector. The individual, which is less deviated from the local ideal vector,

wins the tournament. If a tie occurs sharing is used to decide the winner. The individual, which is in a less crowded region, wins the tournament in case of a tie.

Just as in the first variant this algorithm also gives *n* best solutions to the DM to decide on (Coello and Christiansen, 1999).

3.2.5.5. The Contact Theorem to Detect Pareto Optimal Solutions

This algorithm is based on the contact theorem to determine relative distances of a solution vector with respect to the Pareto set. A solution is initially generated at random, and is considered to be Pareto optimal. Its fitness is d_1, which is an arbitrarily chosen value called the starting distance. Then more solutions are generated and a distance value is computed according to the formula below.

$$z_l(X) = \sqrt{\sum_{i=1}^{k} \left(\frac{f_{il} - \phi_i(X)}{f_{il}} \right)} \qquad \text{for } l=1,2,\ldots l_p \tag{3.40}$$

where l_p is the number of Pareto optimal solutions found so far, $\phi_i(X)$ is the solution's i^{th} objective value and f_{il} is the i^{th} objective value for l^{th} Pareto solution.

In the following step, the minimum value of the set $\{z_l(X)\}$ and its corresponding index l^* are found. This value is called $z_{l^*}(X)$. The procedure identifies the Pareto solution closest to the newly generated solution. If the generated solution is Pareto optimal, the fitness is assigned according to the formula below.

$$Fitness = d_{l^*} + z_{l^*}(X) \tag{3.41}$$

After the first generation, d_l is defined using the maximum value of the distances from all existing Pareto solutions. If the newly generated solution is not a Pareto solution, then its fitness is computed using

$$Fitness = d_{l^*} - z_{l^*}(X) \tag{3.42}$$

and $Fitness$=0 in case a negative value results from this expression (Coello, 2000).

3.2.5.6. A Nongenerational Genetic Algorithm

A nongenerational GA uses nongenerational selection in which fitness of an individual is calculated incrementally. The idea comes from the learning classifier

systems, where it was shown that a simple replacement of the worst individual in the population followed by an update on the fitness of the rest of the population works better than a traditional (generational) GA. In this approach, the MOP with k objective functions is transformed into a biobjective problem. One of the objectives is the minimization of domination count (weighted average of the number of individuals that have dominated this individual so far when the individual is compared with a random group of individuals) and the other is the minimization of the moving niche count (weighted average of the number of individuals that lie close according to a sharing function). This biobjective optimization problem is then transformed into a single objective optimization problem by taking a linear combination of these two objectives (Coello, 2000).

3.2.5.7. Randomly Generated Weights and Elitism

This algorithm uses randomly generated weights and elitism to solve the MOP. Randomly generated weights transform the MOP objectives to a scalar objective to form fitness and by the help of elitism some part of the nondominated set is passed to the next generation. The algorithm uses the following steps to solve the MOP.

1. Generate the initial population randomly.
2. Compute the values of *k* objectives for each individual in the population. Then determine the nondominated solutions and keep them in the set *NOND* and keep the other solutions in the set *CURRENT*.
3. If *L* represents the number of individuals in *NOND* and *M* is the size of *CURRENT*, then select (*M-L*) individuals for crossover using the procedure below.
   - Let $r_1, r_2, \ldots, r_k$ random numbers in the interval [0,1]. The fitness function for each individual is

     $$f(X) = \sum_{i=1}^{k} w_i f_i(X) \tag{3.43}$$

     and w_i is

     $$w_i = \frac{r_i}{r_1 + r_2 + \ldots + r_k}. \tag{3.44}$$

   - Select a parent with probability:

$$P(X) = \frac{f(X) - f_{\min}(CURRENT)}{\sum_{X \in CURRENT} \{f(X) - f_{\min}(CURRENT)\}} \tag{3.45}$$

where $f_{min}(CURRENT)$ is the minimum fitness in the current population.

4. Apply crossover to the selected (*M-L*) pairs of parents. Apply the mutation to the newly generated solutions.
5. Randomly select *L* solutions from *NOND*. Then add *L* solutions to the (*M-L*) solutions generated in the previous step to construct a population of size M.
6. Go to step 2, if stopping condition is not satisfied. If stopping condition is satisfied, report the solutions (Coello, 2000).

3.2.5.8. Multiple Objective Genetic Algorithm

The Multiobjective Optimization Genetic Algorithm (MOGA) developed by Fonseca and Fleming (1993) is an algorithm which uses Pareto ranking and sharing on fitness values.

In this algorithm, an individual's rank corresponds to the number of individuals in the current population by which it is dominated (Fonseca and Fleming, 1995). Consider, for example, an individual X_i of generation *t* dominated by $p_i^{(t)}$ individuals in the current generation (Coello, 2000).

$$rank(X_i, t) = 1 + p_i^{(t)} \tag{3.46}$$

Nondominated individuals are, therefore, all assigned the same rank, while dominated ones are penalized according to the population density in the corresponding region of the trade-off surface. Fitness is assigned by interpolating, for instance, linearly, from the best to the worst individuals in the population, and then averaging it between individuals with the same multiobjective rank.

By combining Pareto dominance with partial preference information in the form of a goal vector in MOGA, Fleming and Fonseca have also provided a means of evolving only a given region of the trade-off surface. While the basic ranking scheme remains unaltered, the now Pareto-like comparison of the individuals selectively excludes those objectives that already satisfy their goals. Specifying fully unattainable goals causes objectives never to be excluded from comparison, which is the original Pareto ranking. Changing the goal values during the search alters the fitness landscape

accordingly and allows the DM to direct the population to zoom in on a particular region of the trade-off surface (Fonseca and Fleming 1995).

MOGA pseudocode is given in the Appendix-A in Figure A.1.

3.2.5.9. Nondominated Sorting Genetic Algorithm

Nondominated sorting genetic algorithm (NSGA) is based on the ranking of nondominated solutions. Beside this ranking concept, in NSGA, a "dummy fitness" is also defined. In NSGA, the initial population is generated randomly and the nondominated solutions of this population are assigned rank 1. After this step, rank 1 individuals are temporarily taken out and the nondominated solutions are identified which are assigned rank 2. This ranking mechanism (Figure 3.5) continues until all the individuals in the population are ranked. According to their ranking all the individuals are assigned a dummy fitness value starting from N (N=population size) for rank 1 and smaller values as the rank increase (Bagchi, 1999).

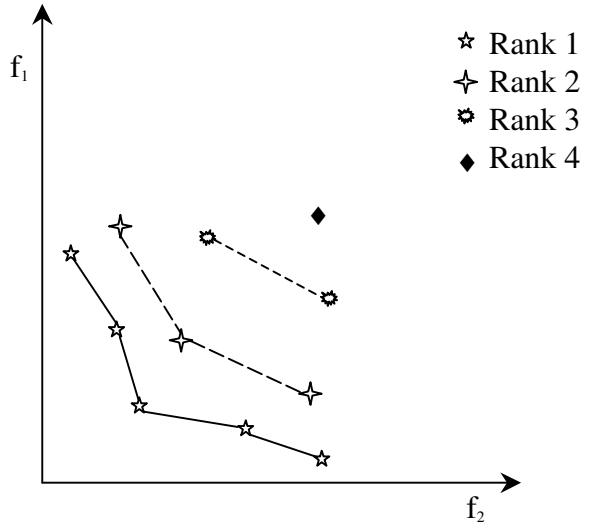

Figure 3-5 NSGA ranking mechanism for a biobjective problem

NSGA also employs fitness sharing and niche formation techniques. In NSGA, individuals are sharing the dummy fitness according to a niche count. The niche count m_i is an estimate of how crowded is the neighborhood (niche) of an individual i (Horn *et al.*, 1994). So, the niche count for an individual is based on the distance between the individual and the others. Distance (d_{ij}) may be defined in two possible ways. The phenotypic distance between two individuals is measured based on the difference in the decoded problem variables while their genotypic distance is measured based on the

difference in the coded problem variables between those two individuals (Bagchi, 1999). The shared fitness and niche count calculations for NSGA are as follows:

$$f_i^{'} = f_i / m_i \tag{3.47}$$

where $f_i^{'}$ is the shared fitness function, f_i is dummy fitness value and m_i is niche count.

$$m_i = \sum_{j=1}^{N} Sh(d_{ij}) = \sum_{j=1}^{N} Sh(d(X_i, X_j)) \tag{3.48}$$

Sh (d_{ij}) is the sharing function. Sharing function (*Sh* (d_{ij})) is a decreasing function of d_{ij}, such that *Sh* (0)=1 and *Sh* [$d \geq \sigma_{share}$]=0. For such a sharing function σ_{share} is called niche radius. A typical sharing function is the triangular sharing function given as follows (Horn *et al*., 1994).

$$Sh\,[d] = 1 - d/\sigma_{share} \qquad \text{for} \quad \sigma_{share} \geq d \tag{3.49}$$

$$Sh\,[d] = 0 \qquad \text{for} \quad \sigma_{share} < d \tag{3.50}$$

A detailed flowchart that explains how NSGA works is given in Figure 3.6.

NSGA does not use elitist strategy to reach a nondominated set. Unlike NSGA, elitist nondominated sorting algorithm (ENGA - an enhancement of NSGA) uses elitist strategy. Like NSGA, ENGA uses nondominated sorting, niche formation, and sharing of fitness based on Pareto ranking. Also like NSGA, ENGA first produces the progenies through crossover and mutation but it uses a different selection procedure. It first ranks the candidate constituents of the next generation by performing an additional nondominated sorting of the combined parents and progenies pool. A controlled fraction of the individuals in this combined pool is then selected to form the next generation, ready to mate and propagate their nondominating schema characteristics. Thus each generation may end up containing several members of the parent chromosomes if they are good enough to outrank (in the nondomination sense) some of the newly created progenies. This selection procedure let the good parents live in the next generation and this makes the algorithm elitist (Bagchi, 1999).

NSGA pseudocode is given in the Appendix-A as Figure A.2.

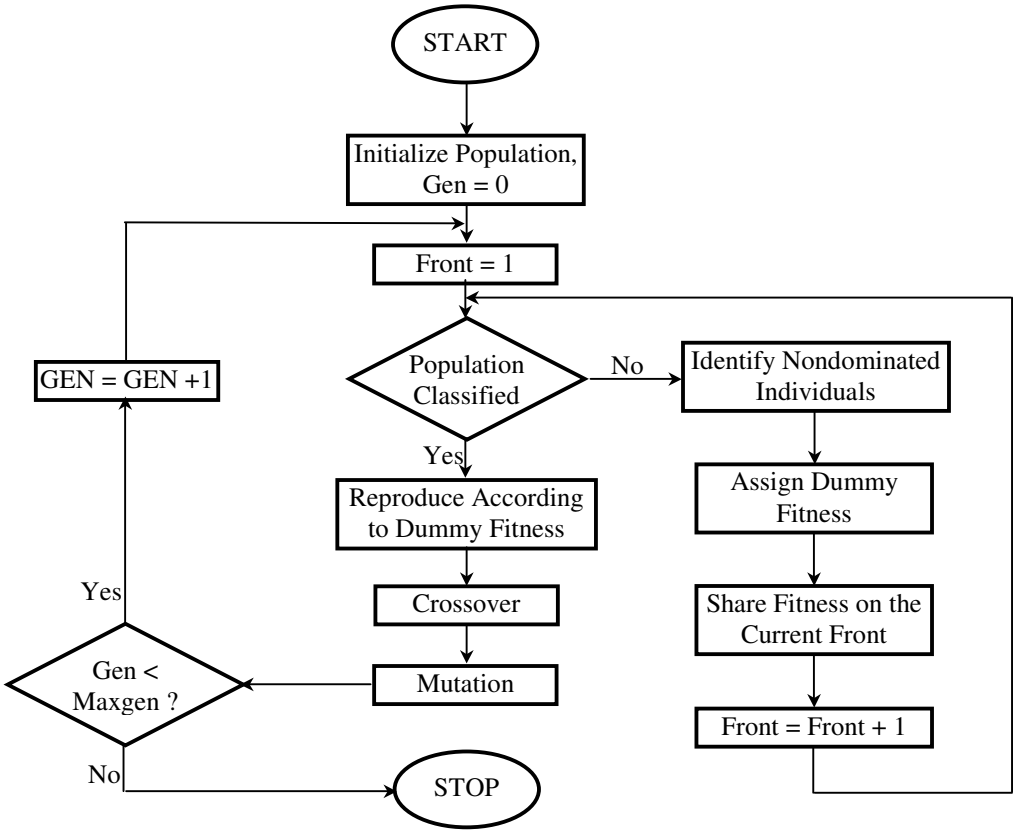

Figure 3-6 The nondominated sorting genetic algorithm (Bagchi, 1999)

3.2.5.10. Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II

NSGA-II is a multiobjective GA based on NSGA. However, Deb *et al.* (2002) called this algorithm as NSGA-II. This new algorithm differs from NSGA in a number of different points. In NSGA-II, nondominated sorting mechanism has been changed, density estimation and crowded comparison operator is used instead of niche formation and finally elitist strategy is added to algorithm. These new concepts should be summarized as follows.

In NSGA-II, for sorting purposes, we calculate two entities: (1) domination count n_p, the number of solutions which dominate the solution p; and (2) S_p, a set of solutions that the solution p dominates.

All solutions in the first nondominated front will have their domination count as zero. For each solution p with n_p=0, each member (q) of the set S_p is visited and their domination count is reduced by one. In doing so, if for any member q the domination

count becomes zero, it is put in a separate list Q. These members belong to the second nondominated front. This process continues until all fronts are identified.

In NSGA-II, density estimation metric and the crowded comparison operator are used to preserve diversity. Density estimation metric provides an estimate of the density of the solutions surrounding a particular solution in the population and is calculated as the average distance of two points on the either side of this point along each of the objectives. This quantity $i_{distance}$ serves as an estimate of the perimeter of the cuboid formed by using the nearest neighbors as the vertices (crowding distance). In Figure 3.7, the crowding distance of the i^{th} solution in its front (marked with solid circles) is the average side length of the cuboid (shown in a dashed box). In the figure, points marked in filled circles are solutions of the same front.

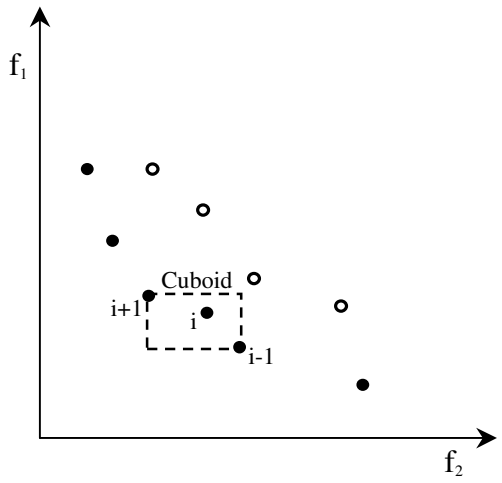

Figure 3-7 Crowding distance calculation (Deb *et al.*, 2002)

The crowding distance computation requires sorting the population according to each objective function value in ascending order of magnitude. Thereafter, for each objective function, the boundary solutions (the solutions with smallest and largest function values) are assigned an infinite distance value. All other intermediate solutions are assigned a distance value equal to the absolute normalized difference in the function values of the two adjacent solutions. This calculation is continued with other objectives. The overall crowding distance value is calculated as the sum of individual distance values corresponding to each objective. Each objective function is normalized before calculating the crowding distance. A solution with a smaller value of this distance measure is, in some sense, more crowded by other solutions.

Crowded comparison operator ($\prec_n$) guides the selection process at the various stages of the algorithm toward a uniformly spread-out Pareto optimal front. Assume

every individual i in the population has two attributes: (1) nondomination rank (i_{rank}) and (2) crowding distance ($i_{distance}$).

A partial order of $\prec_n$ can be defined as follows.

$i \prec_n j$ if ($i_{rank} < j_{rank}$) or;

$i_{rank} = j_{rank}$ and ($i_{distance} > j_{distance}$)

That is, between two solutions with differing nondomination ranks, the solution with a better rank is chosen. Otherwise, if the two solutions belong to the same front, then we prefer the solution that is located in a less crowded region.

The algorithm starts with a randomly generated population (P_0). Each solution is assigned a fitness (or rank) equal to its nondomination level and minimization of the fitness is assumed. At first, by using binary tournament selection, recombination, and mutation operators, an offspring population of Q_0 is created with a size N (population size). After this step elitist strategy is implemented (Deb *et al.*, 2002).

For a generation (t), first a combined population of $R_t = P_t \cup Q_t$ is formed with a size of $2N$. Then the R_t is sorted according to nondomination and grouped. Best nondominated set is called F_1, second best set is called F_2 and so on. The first set (F_j) that the sum of individuals (beginning from F_1) is determined. The set F_j is sorted according to crowding distance and k of the individuals of F_j (where $k = \sum_{i=1}^{j-1} |F_i|$) is passed to generation $t+1$. (Figure 3.8)

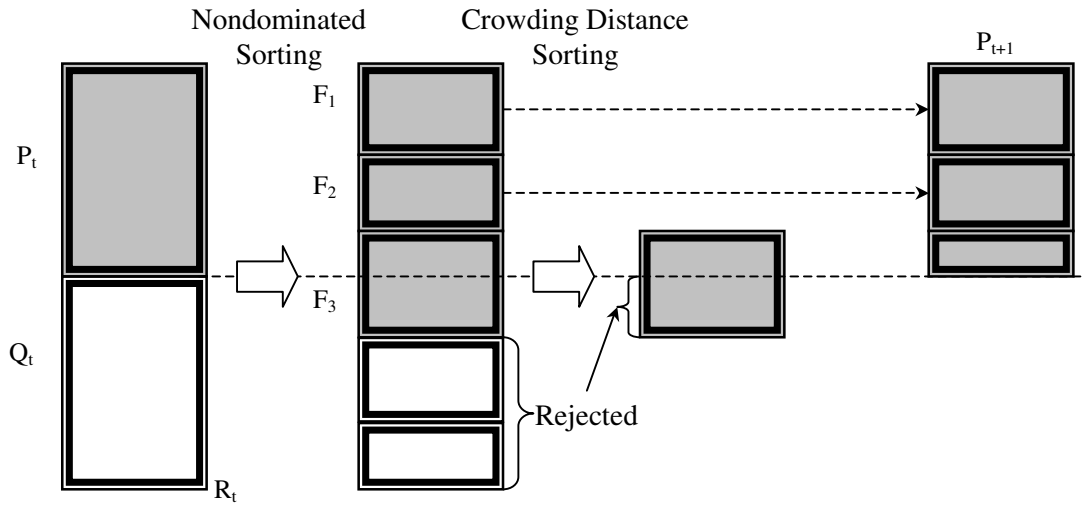

Figure 3-8 NSGA-II procedure (Deb *et al.*, 2002)

NSGA-II pseudocode is given in the Appendix-A in Figure A.3.

3.2.5.11. Niched Pareto Genetic Algorithm

The Niched Pareto Genetic Algorithm (NPGA) is an approach that employs fitness sharing and niche formation techniques. In NPGA, to avoid convergence and maintain multiple Pareto optimal solutions, tournament selection is altered in two ways. Firstly, Pareto domination tournament is added and then, actually when a tie occurs, sharing is implemented to determine the winner.

In Pareto domination tournaments, two candidates for selection are picked at random from the population. These two candidates are compared with a sample set of solutions, which is randomly selected from the population. If one candidate is dominated by the comparison set (with a size t_{dom}) , and the other is not, the latter is selected for reproduction. If neither or both are dominated by the comparison set, then sharing is used to choose the winner.

Sharing which is employed in NPGA is not different from the technique that is employed in NSGA, but in NPGA the fitness function decision is left to the person who implements the algorithm, i.e., there is no defined fitness function. Horn *et al.* (1994) suggest triangular sharing, but different functions can also be employed. NPGA degrades the fitness to the shared fitness in the same manner that is used in NSGA ($f_i^{'} = f_i / m_i$).

When the candidates are either both dominated or both nondominated, it is likely that they are in the same equivalence class, i.e., in the partial order induced by the domination relation. Since the purpose is to maintain diversity it is not necessary to degrade the fitness function if the tournament selection is used. The niche count will be used to order the two candidates. The candidate with a lower niche count will be the winner of the tournament. This type of sharing is called equivalence class sharing.

The performance of the NPGA is somewhat sensitive to the amount of domination versus sharing pressure applied. This means the parameters t_{dom} and σ_{share} play a critical role in the success of NPGA (Horn *et al.*, 1994).

In NPGA-II, Pareto ranking and tournament selection are used. Niche counts in the NPGA-II are calculated using individuals in the partially filled next generation. This is called continuously updated fitness sharing (Coello *et al.*, 2002).

NPGA and NPGA-II pseudocodes are given in the Appendix-A in Figure A.4 and Figure A.5, respectively.

3.2.5.12. Strength Pareto Evolutionary Algorithm

The Strength Pareto Evolutionary Algorithm (SPEA) uses an archive containing nondominated solutions previously found (so called the external nondominated set). At each generation, nondominated individuals are copied to the external nondominated set. For each individual in this external set, a strength value is computed. This strength is similar to the ranking value of MOGA, since it is proportional to the number of solutions to which a certain individual dominates. The fitness of each member of the current population is computed according to the strengths of all external nondominated solutions that dominate it. Additionally, a clustering technique called "average linkage method" is used to keep diversity (Zitzler and Thiele, 1999, Coello *et al.*, 2002).

SPEA-II has three main differences with respect to its predecessor. First, it incorporates a fine grained fitness assignment strategy which takes into account for each individual the number of individuals that dominate it and the number of individuals by which it is dominated. Second, it uses a nearest neighbor density estimation technique, which guides the search more efficiently. Third, it has an enhanced archive truncation method that guarantees the preservation of boundary solutions (Zitzler *et al.*, 2001, Coello *et al.*, 2002).

SPEA and SPEA-II pseudocodes are given in the Appendix-A in Figure 8.6 and Figure 8.7, respectively.

3.2.5.13. Pareto Archived Evolution Strategy

Pareto archived evolution strategy (PAES) consists of a (1+1) evolution strategy (i.e., a single parent that generates a single offspring) in combination with a historical archive that records some of the nondominated solutions previously found. This archive is used as a reference set against which each mutated individual is being compared, just like the tournament competitions that are used in NPGA.

PAES also uses a novel approach to keep diversity, which consists of a crowding procedure that divides objective space in a recursive manner. Each solution is placed in

a certain grid location based on the values of its objectives. A map of such grid is maintained, indicating the number of solutions that reside in each grid location. Since the procedure is adaptive, no extra parameters are required except for the number of divisions of objective space. Furthermore, the procedure has a lower computational complexity than traditional niching methods (Coello *et al.*, 2002).

PAES pseudocode is given in the Appendix-A in Figure A.8.

3.2.5.14. Pareto Envelope-based Selection Algorithm

Pareto envelope-based selection algorithm (PESA) uses a small internal population and a larger external (or secondary) population. PESA uses the same hypergrid division of phenotype space to maintain diversity. However, its selection mechanism is based on the crowding measure used by the hypergrid previously mentioned. This same crowding measure is used to decide what solutions to introduce into the external population (i.e, the archive of nondominated vectors found along the evolutionary process).

The revised form of PESA is also generated as PESA-II, the only difference of PESA-II is that it uses region-based selection. In region-based selection, the unit of selection is a hyperbox rather than an individual. The procedure consists of selecting a hyperbox and then randomly selecting an individual within such hyperbox.

PESA pseudocode is given in the Appendix-A in Figure A.9.

3.2.5.15. The Micro-Genetic Algorithm for Multiobjective Optimization

A micro-genetic algorithm is a GA with a small population size and reinitialization process. The way in which micro-GA works is illustrated in Figure 3.9. First, a random population is generated. This random population feeds the population memory, which is divided into two parts: a replaceable and a non-replaceable portion. The non-replaceable portion of the population memory never changes during the entire run and is meant to provide the required diversity for the algorithm. In contrast, the replaceable portion experiences changes after each cycle of the micro-GA.

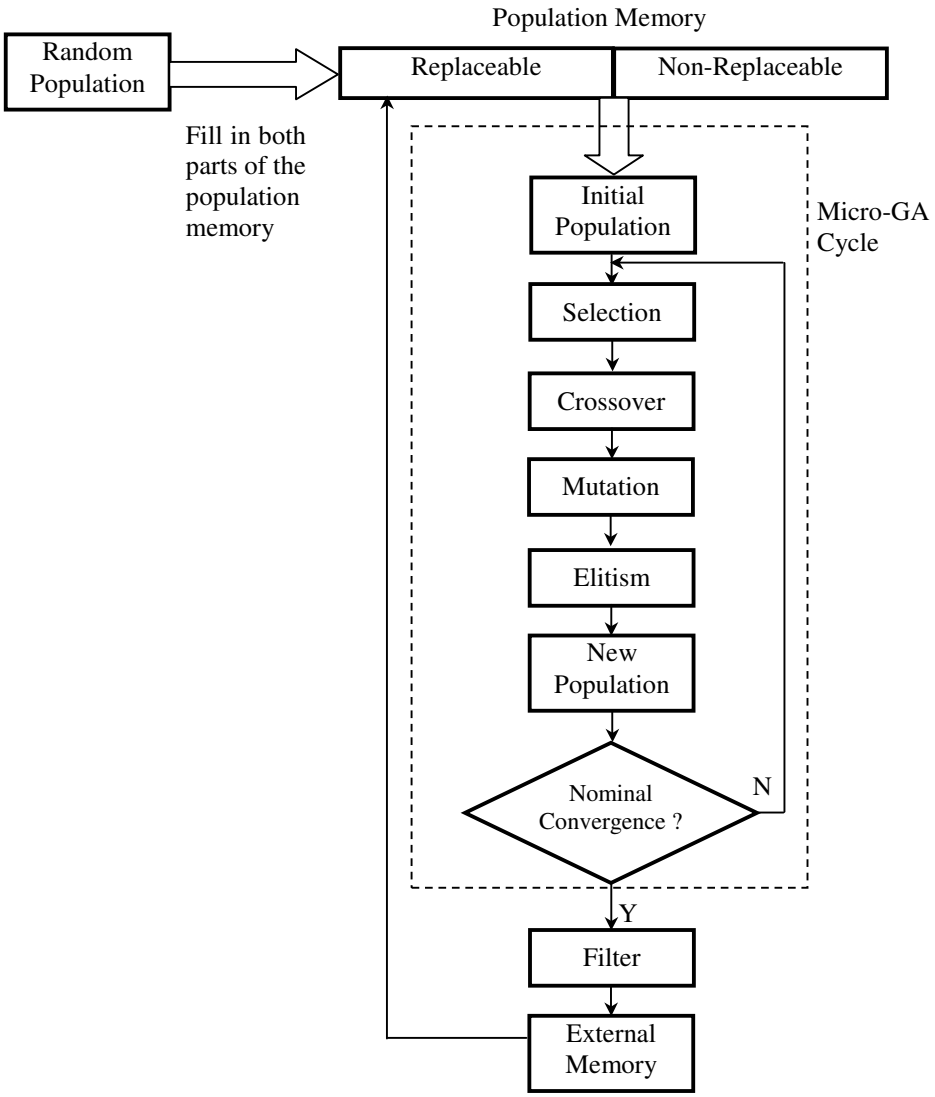

Figure 3-9 Micro-GA for multiobjective optimization (Coello *et al.*, 2002)

The population of the micro-GA at the beginning of each of its cycles is taken from both portions of the population memory so that there is mixture of randomly generated individuals (non-replaceable portion) and evolved individuals (replaceable portion). During each cycle micro-GA undergoes conventional genetic operators. After the micro-GA finishes one cycle, two nondominated vectors are chosen from the final population and they are compared with the contents of the external memory. If either of them remains as nondominated after comparing it against the vectors in this external memory, then they are included there. This is the historical archive of nondominated vectors. All dominated vectors contained in the external memory are eliminated.

The micro-GA uses three forms of elitism: (1) retain nondominated solutions found within the internal cycle of the micro-GA, (2) use a replaceable memory whose

content is partially refreshed at certain intervals and (3) replace the population of the micro-GA by the nominal solutions produced, i.e., the best solutions found after a full internal cycle of the micro-GA.

3.2.6. Multiobjective Evolutionary Algorithm Performance Metrics

What metrics might adequately measure a Multiobjective Evolutionary Algorithm's (MOEA) results or allow meaningful comparison of specific MOEA implementations? Appropriate metrics must be selected upon which to base MOEA performance claims, and as the literature offers few quantitative MOEA metrics, proposed metrics must be carefully defined to be useful. Additionally, no single metric can entirely capture total MOEA performance, as some measure algorithm effectiveness and others efficiency. Temporal effectiveness and efficiency may also be judged, e.g. measuring a MOEA's progress each generation. All may be considered when judging a MOEA against others. Following are possible metrics developed for use in analyzing these experiments, but they should not be considered as a complete list (Coello *et al.*, 2002).

3.2.6.1. Error Ratio (ER)

Error ratio is the ratio of the number of solutions that are in the true nondominated front (PF_{true}) and to the number of solutions in the algorithm's nondominated front (PF_{known}). The following is the mathematical formula of error ratio.

$$ER = \frac{\sum_{i=1}^{n} e_i}{n} \tag{3.51}$$

where n is the number of solutions in PF_{known} and

$$e_i = \begin{cases} 0 & \text{if solution } i \in PF_{true}, \qquad i = 1,...,n \\ 1 & \text{otherwise.} \end{cases} \tag{3.52}$$

3.2.6.2. Two Set Coverage (CS)

Two set coverage is a comparative metric, which can be termed relative coverage comparison of two sets. Consider X, $Y \subseteq X$ as two sets of phenotype of decision vectors. CS is defined as the mapping of the order pair (X,Y) to the interval [0,1].

$$CS(X,Y) = \frac{\left|\{a \in Y; \exists b \in X : b \geq a\}\right|}{\left|Y\right|} \tag{3.53}$$

where $b \geq a$ means b dominates a.

If all the points in Y are dominated or are equal to points in X, then by definition CS=1. CS=0 implies the situation when none of the points in Y is dominated by X (Knowles and Corne, 2001).

3.2.6.3. Generational Distance (GD)

This metric is a value representing in the average how far PF_{known} is from PF_{true} and is defined as :

$$GD = \frac{(\sum_{i=1}^{n} d_i^{\,p})^{1/p}}{n} \tag{3.54}$$

where n is the number of vectors in $PF_{known.}$ For the case of $p=2$ and d_i is the Euclidean distance in the objective space between each vector and the nearest member of PF_{true}. GD = 0 indicates that PF_{true}= PF_{known}; any other value of GD indicates that PF_{known} deviates from PF_{true} with a higher value of GD implying higher deviation.

Also, the kernel can be modified as $(d_{rel(i)}\text{-}d_{ave})$ for a relative comparison where $d_{rel(i)}$ is the relative distance between two consecutive PF_{known} fronts for the last two generations. Here, d_{ave} is the average of the distances $d_{rel(i)}$ across a region. This is similar to an empirical convergence metric.

3.2.6.4. Maximum Pareto Front Error (ME)

It is difficult to measure how well a set of vectors compares to another. For example, in comparing PF_{known} to PF_{true}, one wishes to determine how far apart the two

sets are and how well they conform in shape. This particular metric determines a maximum error band which, considered with respect to PF_{known}, encompasses every vector in PF_{true}. Put in another way, this is the largest minimum Euclidian distance between each vector in PF_{known} and the corresponding closest vector in PF_{true}. This metric is defined as:

$$ME = \max_j (\min_i \left| f_1^i(X) - f_1^j(X) \right|^p + \left| f_2^i(X) - f_2^j(X) \right|^p)^{1/p}, \quad (3.55)$$

where $i=1,\ldots,n_1$ and $j=1,\ldots,n_2$ index vectors respectively in PF_{known} and PF_{true}, and $p=2$. A result of $ME=0$ indicates $PF_{known} \subseteq PF_{true}$; any other result indicates at least one vector in PF_{known} is not in PF_{true}.

3.2.6.5. Average Pareto Front Error

This metric also attempts to measure the convergence property of an MOEA by using distance to PF_{true}. From each solution in PF_{known}, its perpendicular distance to PF_{true} is determined by approximating PF_{true} as a combination of piecewise linear segments with the average of these distances defining the metric value.

3.2.6.6. Spacing (S)

This metric aims to measure the spread (distribution) of vectors throughout PF_{known}. Spacing is proposed to measure the range (distance) variance of neighbouring vectors in PF_{known}. Spacing is defined as:

$$S = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\bar{d} - d_i)^2}, \quad (3.56)$$

where $d_i = \min_j \left(\left| f_1^i(X) - f_1^j(X) \right| + \left| f_2^i(X) - f_2^j(X) \right| \right), i, j = 1,...,n,$ $\bar{d}$ is the mean of all d_i and n is the number of vectors in PF_{known}. A value of zero for this metric indicates all members of PF_{known} are equidistantly spaced. Note that the vectors composing PF_{true} in objective space may not be uniformly spaced.

3.2.6.7. Distributed Spacing (DS)

Distributed spacing is similar to spacing and it aims to measure how well a MOEA has distributed Pareto optimal solutions over a nondominated region. This metric is defined as:

$$DS = \left(\sum_{i=1}^{q+1} \left(\frac{n_i - \overline{n}_i}{\sigma_i}\right)^p\right)^{1/p} \tag{3.57}$$

where q is the number of desired optimal points and the $(q+1)^{th}$ subregion is the dominated region, n_i is the actual number of individuals in the i^{th} subregion of the nondominated region, $\overline{n}_i$ is the expected number of individuals in the i^{th} subregion of the nondominated region, p=2 and σ_i^2 is the variance of the individuals serving the i^{th} subregion of the nondominated region. For this metric, a low performance measure characterizes an algorithm with a good distribution capacity.

3.2.6.8. Hyperarea and Hyperarea Ratio (H, HR)

Hyperarea (H) metric calculates the hyper volume of the multi-dimensional region enclosed by the PF_{known} and a "reference point", hence computing the size of the region PF_{known} dominates. Hyperarea calculation for a biobjective minimization problem is given in Figure 3-10.

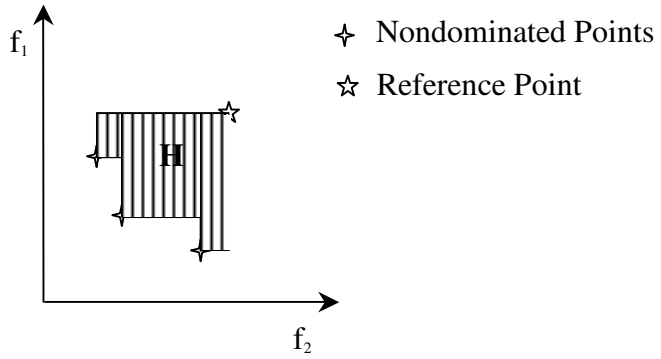

Figure 3-10 Hyperarea calculation for a biobjective minimization problem (Knowles & Corne, 2001)

Hyperarea ratio is the ratio of the hyperarea of PF_{known} (H_1) to the hyperarea of the PF_{true} (H_2).

$$HR = \frac{H_1}{H_2}. \tag{3.58}$$

3.2.6.9. Overall Nondominated Vector Generation and Ratio (ONVG, ONVGR)

Most MOEAs add $PF_{current}$ to PF_{known} each generation, possibly resulting in different cardinalities for PF_{known}. This metric then measures the total number of nondominated vectors found during MOEA execution and is defined as:

$$ONVG = \left|PF_{known}\right| \tag{3.59}$$

It is difficult to specify what good values for ONVG might be. PF_{known}'s cardinality may change for different MOPs. Reporting the ratio of PF_{known}'s cardinality to the discretized PF_{true}'s gives some feeling for the number of nondominated vectors found versus how many exist to be found. This metric is then defined as:

$$ONVGR = \frac{\left|PF_{known}\right|}{\left|PF_{true}\right|}. \tag{3.60}$$

3.2.6.10. Generational Nondominated Vector Generation (GNVG)

This metric tracks how many nondominated vectors are produced at each MOEA generation and is defined as:

$$GNVG = \left|PF_{current}(t)\right|. \tag{3.61}$$

3.2.6.11. Nondominated Vector Addition (NVA)

As globally nondominated vectors are sought, one hopes to add new nondominated vectors to PF_{known} at each generation *t*. This metric is then defined as:

$$NVA = \left|PF_{known}(t)\right| - \left|PF_{known}(t-1)\right|. \tag{3.62}$$

However, this metric may be misleading. A single vector added to $PF_{known}(t)$'s size may also remain constant for several successive generations even if $GNVG \neq 0$.

4. PROBLEM DEFINITION AND SOLUTION APPROACHES

4.1. Problem Description

In the problem under consideration, project scheduling under risk is modelled in order to represent the effects of identified risks that may occur during activities. The model is a mixed integer programming model whose aim is to minimize the expected cost of the project.

The model contains different elements when it is compared to traditional project scheduling models. In the hierarchical order, the model contains depicted elements in Figure 4.1.

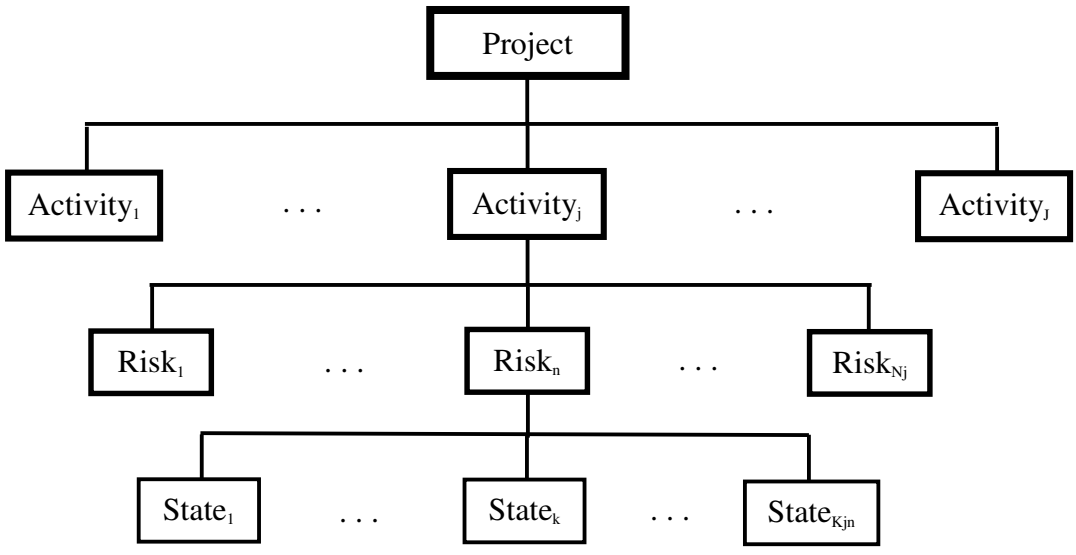

Figure 4-1 Project scheduling model elements

In this model, the activities of the project have identified risks. These risks represent the events that may occur during the activities and it is assumed that these events affect only the duration of the activities.

As is the case in real life, the model also covers the preventive measures that may be taken against the risks. These preventive measures are modelled by the states of the risks. States have a probability of occurrence and an impact. In the model, if there is no preventive measure taken against the risk, this situation is represented by choosing state 1. In the model, increasing the level of preventive measure corresponds to increasing the index of the state chosen which decreases the level of the risk.

The cost function of the model covers four different costs that may occur during the project, these are overhead cost, labor cost, risk reducing cost and penalty cost. These cost functions are explained in greater detail in the mathematical formulation of the model.

4.2. Mathematical Formulation of the Problem

The problem is formulated here as an optimization problem to minimize the expected project cost under risks. It is assumed that the risks are independent and their impacts are additive at the activity level. It is further assumed that all the risks associated with an activity are identified and the risks are static throughout the project life. The problem is represented on an activity-on-node (AON) network with one starting and one ending node.

For a complete understanding of the model, first the notation is explained, then the complete model is stated and finally the important parts of the model are explained step by step.

Notation:

$\{J\}$: Set of activities $j=1,\ldots,J$;

$\{P_j\}$: Set of immediate predecessors of activity j;

$\{L_j\}$: Set of resource types for activity j;

$\{N_j\}$: Set of risks n assigned to activity j;

d_j: Duration of activity j with no risks involved;

C_p: Unit penalty cost of being late;

C_o: Unit cost of overhead;

T_{plan}: Due date set for the project;

K_{jn}: Number of states for the probability of occurrence of risk n on activity j;

P_{jnk}: Probability of risk n's occurrence, for activity j at state k;

I_{jnk}: Impact of risk n, if it occurs, for activity j at state k;

C_{lj}: Unit cost of resource type l_j;

C_{jnk}: Cost of reducing the risk level from state *1* level to state k level for risk n at activity j;

W_{lj}: Number of workers of type l_j assigned to activity j;

$E(TC)$: Expected total cost;

EST_j: Earliest start time of activity j;

EFT_j: Earliest finish time of activity j;

EFT_J: Expected makespan of the project;

$d_j^{'}$: Expected duration of activity j under risk;

y: Expected lateness of project;

$$X_{jnk} = \begin{cases} 1, & \text{if the } k^{th} \text{ state is chosen for } n^{th} \text{ risk of } j^{th} \text{ activity} \\ 0, & \text{otherwise} \end{cases};$$

Model:

$$Min E(TC) = y * C_p + \sum_{j=1}^{J} \sum_{n=1}^{N_j} \sum_{k=1}^{K_{jn}} C_{jnk} * X_{jnk} + C_o * EFT_J + \sum_{j=1}^{J} \sum_{l_j=1}^{L_j} W_{l_j} * d^{'}_{j} * C_{l_j} \quad (4.1)$$

subject to:

$$EST_1 = 0 \quad (4.2)$$

$$EST_j = Max\{EFT_i \mathrm{I} i \in P_j\} \qquad j = 2,...,J \quad (4.3)$$

$$EFT_j = EST_j + d^{'}_{j} \qquad j = 1,...,J \quad (4.4)$$

$$d^{'}_{j} = d_j + d_j * \sum_{n=1}^{N_j} \sum_{k=1}^{K_{jn}} X_{jnk} * I_{jnk} * P_{jnk} \qquad j = 1,...,J \quad (4.5)$$

$$C_{jn1} = 0 \qquad j = 1,...,J; n = 1,...,N_j \quad (4.6)$$

$$\sum_{k=1}^{K_{jn}} X_{jnk} = 1 \qquad j = 1,...,J; n = 1,...,N_j \quad (4.7)$$

$$y = \begin{cases} EFT_J - T_{plan}, & \text{if } EFT_J > T_{plan} \\ 0, & \text{otherwise} \end{cases} \quad (4.8)$$

$$X_{jnk} \in \{0,1\} \qquad j = 1,...,J;\ n = 1,...,N_j;\ k = 1,...,K_{jn}$$

This is a 0-1 integer programming model which aims to minimize expected total cost (Equation 4.1) of the project. This cost is represented as the sum of four cost components. The first cost component is the penalty cost and formulated as the product

of unit penalty cost and lateness. Second component of the cost function is the risk reducing cost. The third component is the overhead cost and it is the product of unit overhead cost and makespan. Final component is the labor cost of the project.

Equations through 4.2 to 4.4 are the critical path method (CPM) equations for forward recursion.

Equation 4.5 is used to calculate the expected duration of an activity. While calculating the expected duration of an activity, the additional risk related durations are added to normal activity duration. As an example, assume we have an activity having one risk and three states (Table 4.1) where TU and MU stand for time unit and monetary unit, respectively. The expected duration of the activity can be calculated as follows, if the second state is chosen.

$d'_x = 20 + 0.6 * 0.5 * 20 = 26$ TU

Table 4-1 Risk states for an activity

Activity X	Duration (d): 20 (TU)	$\lvert L_X \rvert = 1$	$W_{1X} = 3$
State	Probability of Occurrence (P_{jnk})	Impact (I_{jnk})	Cost (MU) (C_{jnk})
1	0.7	0.5	0
2	0.6	0.5	150
3	0.6	0.4	300

As it is seen in Table 4.1 and stated by Equation 4.6 the first state of the risks has a zero cost and this state is named as the base case. This corresponds to the real life situation of taking no preventive measures against a risk. Thus no cost is incurred.

Finally, Equation 4.7 assures the selection of one and only one state for each of the risks.

The decision variables for the model are 0-1 variables. There are $(\sum_{j=1}^{J}\sum_{n=1}^{N_j}\sum_{k=1}^{K_{jn}} 1)$ number of variables. For small sized problems, it is straightforward to solve such a problem with a mathematical programming solver. But for large problems, this problem is a computationally costly problem to solve. The solution approach suggested for this problem is the topic of next section.

4.3. Solution Approach

The problem solved in this thesis is biobjective. The first objective is to minimize the expected total project cost. The second objective to achieve is the minimization of the expected makespan. These two objectives are obviously conflicting. Equations 4.9 and 4.10 represent the two objectives to be achieved.

$$MinE(TC) = y * C_p + \sum_{j=1}^{J} \sum_{n=1}^{N_j} \sum_{k=1}^{K_{jn}} C_{jnk} * X_{jnk} + C_o * EFT_J + \sum_{j=1}^{J} \sum_{l_j=1}^{L_j} W_{l_j} * d'_j * C_{l_j} \quad (4.9)$$

$$MinE(C_{\max}) = EFT_J \quad (4.10)$$

Multiobjective optimization with posteriori preference articulation is a developing topic in the OR literature. GAs constitute a popular solution procedure for this group of problems. As evidence of this popularity is that approximately 70% of the metaheuristic approaches suggested and published between 1991 and 2000 are GAs (Jones *et al.*, 2002). Since GA uses parallel search techniques and multiobjective optimization problems have several nondominated solutions, this problem class and the solution procedure make a perfect match. Because of this reason, in this thesis, GAs are used to solve the problem.

After solving the biobjective problem some of the solutions need further improvement for decreasing expected total cost while keeping the critical path fixed. For this reason, three improvement heuristics are proposed.

4.3.1. Genetic Algorithms Employed

4.3.1.1. The Chromosome Representation and the Management of the Genetic Algorithms Employed

In this study, direct representation is used for encoding a solution to the problem. Each gene corresponds to a risk in the chromosome and the number in the gene represents the state that will be chosen for this risk.

The number of risks in the problem determines the number of genes in the chromosome and an extra three bit portion is added to display the expected makespan, expected total cost and fitness values. The chromosome representation is depicted in Figure 4.2.

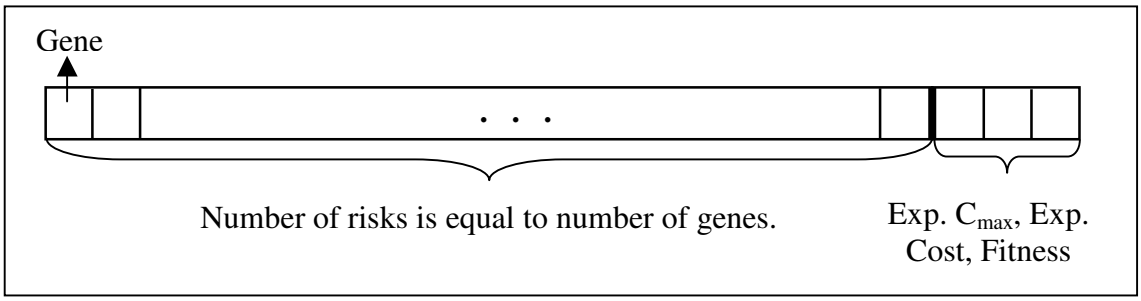

Figure 4-2 Chromosome representation

Decoding: The number in the gene represents the state that will be chosen for the risk.

Selection Mechanism: Random wheel selection is used. The fitness values of all individuals are summed up and the fitness of each individual is normalized by dividing to this total. Then a random number is chosen and this random number is used to find out which individual will be selected.

Crossover: One point crossover is used for the GA. A number is chosen between one and the number of risks. This is used as the cutting point, where the two chromosomes are cut. The parts that have been generated by cutting operation are crossed and two new individuals are generated.

Mutation: Bit mutation is used. The value on the randomly chosen gene of the chromosome is replaced with another value also randomly generated.

Generation Cycle: First generation is generated randomly. Then the later generations are generated based on this first generation.

While generating the later generations an operator is chosen with the specified probabilities (i.e. crossover is chosen with a probability of P_c, mutation is chosen with a probability of P_m and reproduction is chosen with a probability of $(1-(P_c+P_m))$). Then the chromosome(s) is (are) chosen according to the operator. Finally, the chosen operator is applied to the chosen chromosome(s).

Different than the traditional GA approach these operators are applied in a parallel fashion rather than the serial application of the crossover and mutation operators.

After defining how the GA routine works, multiobjective GA's can be defined. Before applying the improvement heuristics to the problem, two strategies based on the VEGA (Vector Evaluated Genetic Algorithm) and a new genetic algorithm are used. These algorithms are explained in greater detail in the following subsections.

4.3.1.2. VEGA Based Strategies

In the original VEGA (see section 3.2.5.1), the fitness function is calculated after dividing the population into subpopulations. Subpopulations contain equal number of individuals and these individuals are evaluated according to the objective of the corresponding subpopulation they are in. This evaluating frame has a drawback called speciation. This problem arises because this technique selects individuals who excel in one dimension without looking at other dimensions. The potential danger is that the procedure evolves without generating middling performance individuals. Middling refers to an individual with acceptable performance, perhaps above average in all objectives, but not outstanding when measured by any particular function. This problem is depicted in Figure 4.3. In the figure, filled circles are identified by VEGA; others represent the middling individuals, which could not be identified.

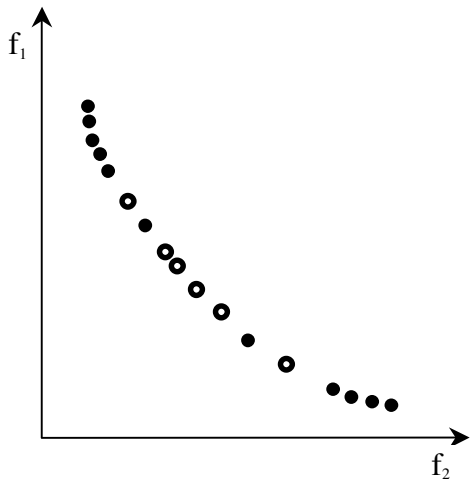

Figure 4-3 Middling individuals in VEGA

Middling individuals are the results of VEGA and this type of a solution is undesirable. To avoid the middling problem of VEGA two strategies are proposed here. These strategies bring a dynamic subpopulation sizing to the algorithm. Rather than dividing the population equally, they divide the population according to a parameter.

In these strategies, firstly, the problem is solved for each of the objectives. The values of these solutions constitute a basis for comparison. When the problem is solved with the objective of minimizing expected total project cost, a solution with a cost value C^* is obtained. When the problem is solved with the objective of minimizing expected project duration, a solution with a makespan value $C_{max}{}^*$ is obtained. These two values are called the ideal values of the objectives.

The two strategies based on VEGA are developed to avoid clusters on the endpoints of the Pareto front. The distance from ideal points of the objectives are measured for each generation and the next generation's subpopulation sizing is done according to these distances. The distance calculation method determines the strategy.

4.3.1.2.1. Strategy 1

In the first proposed strategy, the distance between the ideal value of the objective and generation's best individual for this objective ($C_{max}{}^b, C^b$) value is measured. The following formula represents the distances for the objectives, d_1 represents the distance for the expected total cost and d_2 represents the distance for the expected makespan.

$$d_1 = \frac{C^b - C^*}{C^*} \tag{4.11-a}$$

$$d_2 = \frac{C_{\max}{}^b - C_{\max}{}^*}{C_{\max}{}^*} \tag{4.11-b}$$

After calculating the distances, the normalized distances are Nd_1 and Nd_2 calculated as shown below.

$$Nd_1 = \frac{d_1}{d_1 + d_2} \tag{4.12-a}$$

$$Nd_2 = \frac{d_2}{d_1 + d_2} \tag{4.12-b}$$

Assume these values are calculated in generation t and the generation size for the genetic algorithm is N. Then for generation t+1, subpopulation sizes become Nd_1*N and Nd_2*N, respectively. So, in generation t+1, (Nd_1*N) individuals will be evaluated according to objective one; (Nd_2*N) individuals will be evaluated according to objective two.

4.3.1.2.2. Strategy 2

Second strategy is based on the same intuition with the first strategy, only differs in the distance measurement. Assume population size for GA is *N*. The following formulas show the difference.

$$d_1 = \sum_{i=1}^{N} \left(\frac{C_i - C^*}{C^*} \right) \tag{4.13-a}$$

$$d_2 = \sum_{i=1}^{N} \left(\frac{C_{\max(i)} - C_{\max}{}^*}{C_{\max}{}^*} \right) \tag{4.13-b}$$

4.3.1.3. Proposed Genetic Algorithm

This new algorithm is a mixture of NSGA-II and NPGA. For each generation generated, fitness is calculated by the help of two components. First component is similar to the Pareto domination tournament of NPGA. But in this algorithm we do not compare an individual with a group of individuals; rather we compare the individual with the entire population. The number of individuals dominated by the individual is represented by N_{dom}. The ratio of domination, R_{dom}, represents the first component.

$$R_{dom} = \frac{N_{dom} + 1}{N_{pop} + 1} \tag{4.14}$$

The second component called the nearest neighbourhood radius, *NNR*, is given below which is the division of nearest individual's distance ($d_{nearest}$) to the maximum distance in the generation (d_{maxgen}).

$$NNR = \frac{d_{nearest}}{d_{\max gen}} \tag{4.15}$$

The product of these two components becomes the fitness value.

$$Fitness = NNR * R_{dom} \tag{4.16}$$

R_{dom} is used to evaluate the fitness of an individual in nondomination sense. By the help of R_{dom}, in each generation individuals are compared with all individuals.

NNR behaves like a sharing function in this algorithm. Similar to but simpler than NSGA-II, *NNR* finds the nearest individual for all individuals and divides the distance between them to the maximum distance in the population. So, if an individual is closer to its nearest neighbour, it is assumed that, it is in a crowded region.

By multiplying R_{dom} and *NNR* fitness of an individual in the nondomination sense and the sharing concept are combined.

In the proposed GA, elitism is employed as well. During the evolution process, the nondominated individuals in each generation are carried to the next generation and the other individuals are formed by using the mutation and crossover operators.

4.3.2. Heuristics to Improve the GA Results

The GA result to the problem gives a makespan and a cost value. The expected cost values may not be satisfactory if it tells us to invest on reducing risk in noncritical activities.

Assume we have project consisting of seven activities with the following network (Figure 4.4). For one solution, the activities on thick lined arcs (1-2-5-7) are on the critical path.

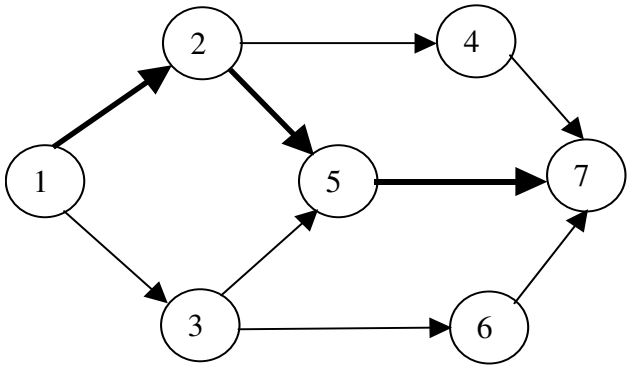

Figure 4-4 Example project network (AON)

For such a solution, the aim of the heuristic is to avoid investing more money than needed to the non-critical activities to reduce their risks, while not changing the risk structure of the activities on the critical path and hence, the makespan.

In this problem, we specify a limit on expected project duration and fix modes of some activities on the critical path that have been chosen by GA. Then we try to minimize the expected cost while preserving the critical path. This problem is a special case of discrete time/cost trade-off problem, which is shown to be NP hard by De *et al.* (1997). In their paper, they have shown that under a due date constraint multi-mode project scheduling problem with the cost minimization as the objective, is an NP hard problem. This problem is similar to the problem that we are trying to solve.

First we need to transform our problem to the multi-mode project scheduling problem. This process can be done by choosing every possible combination of states to different modes. Assume we have an activity with two risks having two states each. For such an activity we can find (2*2) four modes. Then for increasing the solvability of the problem we can also perform a domination search along the modes and we can eliminate the dominated modes. Table 4.2 demonstrates the mode generation and nondominated mode selection.

Table 4-2 Mode generation and nondominated mode selection

Mode No	State chosen for Risk 1	State chosen for Risk 2	Cost	Duration	Domination Statue
1	1	1	5443.20	38.88	Dominated (by mode 2&4)
2	1	2	5034.60	33.39	Nondominated
3	2	1	5587.00	31.05	Dominated (by mode 4)
4	2	2	5178.40	25.56	Nondominated

For the modes of activities the duration column represents the expected durations of activities when these states are chosen. The cost for an activity represents the sum of expected labor cost and the risk reducing costs, which constitute a local trade-off with the expected duration. After identifying the nondominated modes we have a discrete time/cost trade-off problem, whose critical activities have only one mode. Since the multi mode project scheduling problem under a due date with the cost minimization as the objective is NP hard, we can say that our problem is also NP hard. Different than the discrete time/cost trade-off problem in our problem the critical activities have only one mode.

Exact solution approaches to this problem are given by Demeulemeester *et al*. (1996). These solution approaches seemed computationally very costly so that heuristics are tried to be generated. The following sections describe the various heuristics proposed to solve our problem.

4.3.2.1. An Improvement Heuristic Based on Continuous Cost vs Duration Model

Continuous form of project crashing problems has been widely studied. A large number of methods are proposed. Among others Fulkerson (1961) used network flows to generate the project cost curve, Siemens (1971) generated a heuristic by defining effective cost slopes for activities, Goyal (1975) improved Siemens' approach and

Robinson (1975) used dynamic programming to solve the problem. Nowadays these problems can be solved by computers optimally and very quickly.

In this continuous cost vs. duration model based (CCDM) improvement heuristic, first, all the nondominated modes of the problem are identified and they are scattered on a graph as given below. After identifying all the modes and scattering we try to fit a linear curve to these modes. By doing so we can transform our problem to a continuous project crashing problem, which is easier to solve.

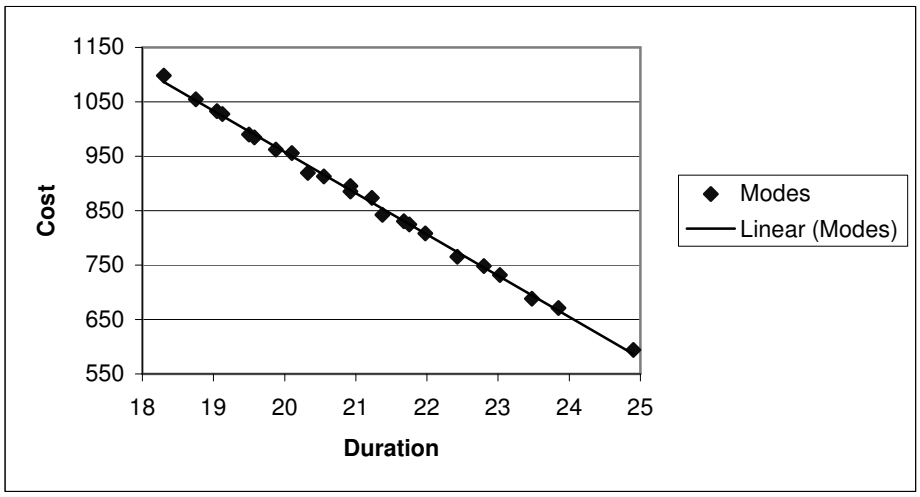

Figure 4-5 Example activity graph.

The continuous form of the problem is represented by the following mathematical model if all the activities of the project can be approximated by a line segment as seen in Figure 4.5.

Notation:

{J}: Set of activities $j=1,\ldots,J$;

{P_j}: Set of immediate predecessors of activity j;

{K}: Set of critical activities k (subset of J);

{U}: Set of non-critical activities u (subset of J);

EST_j: Earliest starting time of activity j;

EFTj: Earliest finishing time of activity j;

p_k: Duration of the activities on the critical path;

s_u: Slope of the curve for noncritical activities (note that slope is negative);

t_u: Duration of the activities on the noncritical path;

a_u: The endpoint of curve, smallest duration;

b_u: The endpoint of curve, largest duration;

CPL: Critical path length;

Model:

$$Min \sum_{u \in U} (d'_u - a_u) * s_u \tag{4.17}$$

subject to:

$$EST_1 = 0 \tag{4.18}$$

$$EST_j = \max\{EFT_i \mathrm{I} i \in P_j\} \qquad j = 2,...,J \tag{4.19}$$

$$EFT_j = EST_j + d'_j \qquad j = 1,...,J \tag{4.20}$$

$$d'_k = p_k \qquad for\, k \in K \tag{4.21}$$

$$EFT_J \leq CPL \tag{4.22}$$

$$a_u \leq d'_u \leq b_u \qquad for\, u \in U \tag{4.23}$$

This model is valid for the situations where the modes can be represented by a single line segment. Most of the time, this line segment would not be adequate to represent all the modes of an activity accurately. For such cases, the modes of the activity is tried to be represented by a piecewise linear function. For projects containing such activities another model is needed. The following model is used for the situations where the cost function of the activities are represented by piecewise linear functions (see Figure 4-6).

For piecewise linear function generation, first a continuous curve is fitted on the nondominated modes (gray line in Figure 4-6). Then the continuous curve is approximated by three connected line segments (black line segments on Figure 4-6). This approximation of the continuous curve by a piecewise linear function is based on one of the methods proposed by Wei and Wang (2003). In this method, authors propose to use tangents to the continuous curve. The first line is drawn tangent to the curve at the beginning point and the last line is drawn tangent at the ending point. The other lines are drawn tangent at the points between the beginning and ending point, which are equally away from other tangent points. The intersection points of the tangent lines constitute the beginning and ending points of segments. As the number of segments increases the precision of the piecewise linear approximation increases.

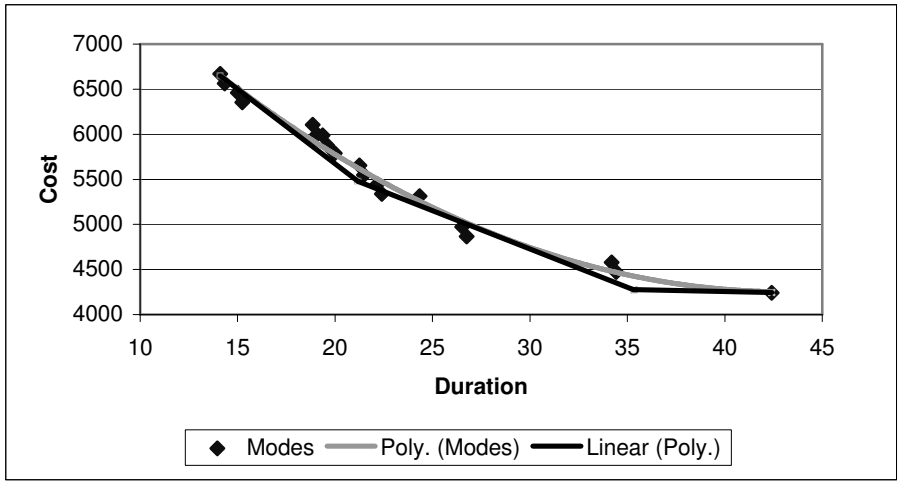

Figure 4-6 Example of piecewise linear curve fitting on an activity

Notation:

{J}: Set of activities $j=1,\ldots,J$;

$\{P_j\}$: Set of immediate predecessors of activity j;

{K}: Set of critical activities k (subset of J);

{U}: Set of noncritical activities u (subset of J);

EST_j: Earliest starting time of activity j;

EFT_j: Earliest finishing time of activity j;

p_k: Duration of the critical activity k;

s_u^m: Slope of the m^{th} segment of curve for noncritical activities (note increasing negative slope as m increases);

d'_u: Duration of the noncritical activity u;

a_u^m: The endpoint of m^{th} segment of curve, smallest duration of segment;

b_u^m: The endpoint of m^{th} segment of curve, largest duration of segment;

M_u: The number of segments of the cost curve for activity u;

X_u^m: Duration on segment m of noncritical activity u;

CPL: Critical path length;

Model:

$$Min \sum_{u \in U} \sum_{m=1}^{M_u} X_u^m * s_u^m \tag{4.24}$$

subject to:

$$EST_1 = 0 \tag{4.25}$$

$$EST_j = \max\{EFT_i \mid i \in P_j\} \qquad j = 2,\ldots,J \tag{4.26}$$

$$EFT_j = EST_j + d'_j \qquad j = 1,...,J \tag{4.27}$$

$$d'_k = p_k \qquad \text{for } k \in K \tag{4.28}$$

$$0 \le X_u^m \le (b_u^m - a_u^m) \qquad \text{for } u \in U;\ m = 1,...,M_u \tag{4.29}$$

$$d'_u = a_u^1 + \sum_{m=1}^{M_u} X_u^m \qquad \text{for } u \in U \tag{4.30}$$

$$EFT_J \le CPL \tag{4.31}$$

The appropriate model is solved by using GAMS©. The durations for noncritical activities are taken from the GAMS© solution and then the modes are found. To find the modes of noncritical activities, for each noncritical activity, if the duration found corresponds to a nondominated mode's duration, then we assign this mode to that activity. Otherwise, we find the nondominated mode with the closest but smaller duration and assign this mode to that activity. Then, for each noncritical activity, we calculate the slacks and the earning per duration value that will result if the activity is performed at its next higher duration mode. Starting with the highest earning per duration activity, we expand the activities without violating the slacks. These operations (the operations after finding the mode from appropriate model solution – starting point assignment) are repeated until no slack exists or there is no further mode to expand to.

The earning per duration ratio is the ratio of the expected cost decrease to expected duration increase between the respective nondominated modes of the activity. For the activity shown in Table 4-2, this value is (from mode 4 to mode 2) 18.37 (=(5178.4-5034.6)/(33.39-25.56)).

Figure 4.7 explains this improvement heuristic more explicitly.

Step 1 - For each noncritical activity,
Determine the nondominated modes.
Fit a continuous curve to these nondominated modes.
Determine a piecewise linear underestimator for the continuous curve.

Step 2 - Using the appropriate model, solve a minimum cost problem keeping the project duration fixed.

Step 3 - For each noncritical activity, if the duration found corresponds to a nondominated mode's duration, then assign this mode to that activity. Otherwise, find the nondominated mode with the closest but smaller duration and assign this mode to that activity (starting point assignment).

Step 4 - For each noncritical activity, calculate the slack and the earning per duration value that will result if the activity is performed at its next higher duration mode.

Step 5 - Starting with the highest earning per duration activity, expand the activity without violating the slacks.

Step 6 - If there are other activities whose slacks are appropriate for expansion, go to step 4; else stop.

Figure 4-7 CCDM improvement heuristic procedure

4.3.2.2. An Improvement Heuristic Based on GA Results

In this GA results based (GAB) improvement heuristic, rather than finding the starting points for the noncritical activities with a continuous model, the GA results are taken as the starting points. The GA may result with the dominated modes for noncritical activities. In these situations, the nondominated mode with a lower duration is found and these are taken as starting points for the heuristic. From this point on, this heuristic is the same as the continuous model based heuristic. First, the earning per duration values and slacks are calculated. Then beginning from the highest earning per duration value, we expand the activity durations. The operations after the starting point assignment are repeated until no slack exists or there is no further mode to expand to.

A step by step procedure for the GAB improvement heuristic is given in Figure 4.8.

Step 1 - For each noncritical activity,
Determine the mode in which GA results.
If the mode is nondominated assign it as the starting point for this activity,
else if the mode is dominated, find the nondominated mode with a lower but closest duration value.

Step 2 - For each noncritical activity, calculate the slack and the earning per duration value that will result if the activity is performed at its next higher duration mode.

Step 3 - Starting with the highest earning per duration activity, expand the activity without violating the slacks.

Step 4 - If there are other activities, whose slacks are appropriate for expansion, go to step 2; else stop.

Figure 4-8 GAB improvement heuristic procedure

4.3.2.3. *From Start* Improvement Heuristic

In this *from start* (FS) improvement heuristic, the starting point assignment is done from scratch. Every noncritical activity is assigned the nondominated mode with the lowest duration as the starting mode. From this point on this heuristic is the same as the others. First, the earning per duration values and slacks are calculated. Then beginning from the highest earning per duration value we expand the activity durations. The operations after the starting point assignment are repeated until no slack exists or there is no further mode to expand to.

The procedure for FS improvement heuristic is given in Figure 4-9.

Step 1 - For each noncritical activity,

Find the nondominated mode with the lowest duration and assign this mode as starting point for this activity.

Step 2 - For each noncritical activity, calculate the slack and the earning per duration value that will result if the activity is performed at its next higher duration mode.

Step 3 - Starting with the highest earning per duration activity, expand the activity without violating the slacks.

Step 4 - If there are other activities, whose slacks are appropriate for expansion, go to step 2; else stop.

Figure 4-9 FS improvement heuristic procedure

5. TESTING AND COMPUTATIONAL STUDY

Parameter setting for GAs has been a difficult issue in most of the GA implementations. For problems, which have a single objective, it is easier to compare the results of different parameters. But in multiobjective optimization problems, as it is mentioned in the Section 3.2.6, it is difficult to compare the results of different algorithms or the different sets of parameters for the same algorithm. Since the results are needed to be compared, first a multiobjective performance metric is defined. Then the determined parameter sets are compared according to this metric. Finally, as the computational study, the algorithms and improvement heuristics are examined on a set of problems.

5.1. Performance Metric

As the performance metric, "extreme hyperarea ratio (EHR)" is developed based on the idea of hyperarea ratio. In hyperarea ratio, the hyperarea resulting from the use of the algorithm is divided to the hyperarea of the true Pareto front. This metric is a subjective but good measure to compare the results of the problems whose true Pareto fronts are known. But if the problem's true Pareto front is not known, it is impossible to use this metric.

For the problems used in this thesis the true Pareto fronts are not known, so another metric is needed to be developed. This metric is the ratio of the hyperarea of the front (Figure 5.1(a)) to the area bounded by the origin and maximum points of the two objective (Figure 5.1(b)).

As it is depicted on Figure 5.1, the EHR becomes as follows.

$$EHR = \frac{H}{A} \tag{5.1}$$

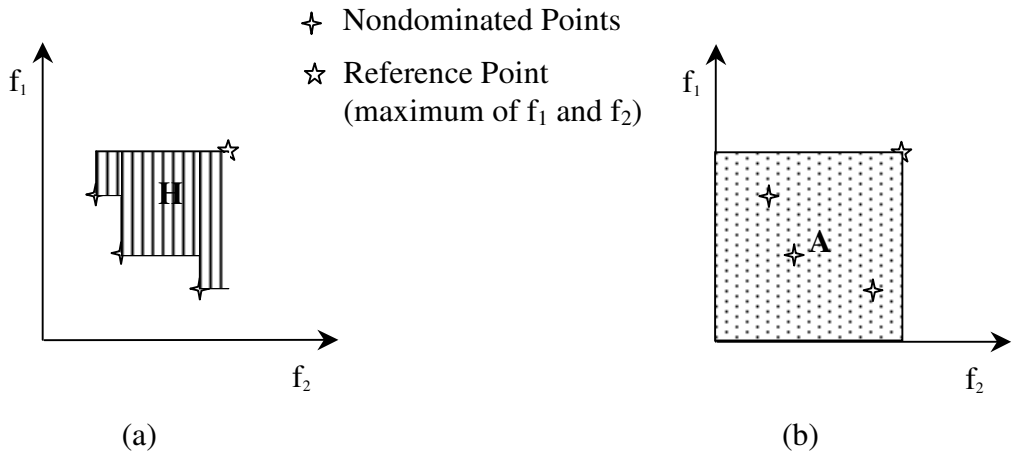

Figure 5-1(a) Hyperarea of the front, (b) maximum area bounded by origin and maximum points.

This ratio is used to determine the parameters of the GAs and to compare the different algorithms.

5.2. Parameter Setting

For determining GA parameters, first a bound is determined for the number individuals that will be generated. This is defined as the point, after which nondominated solutions will be found seldomly. A large number of experiments are done on a project with twenty-five activities. After these experiments a conservative bound of fifty thousand is determined.

After determining the total number of individuals that will be generated, some probabilities for crossover and mutation are determined. Also to determine the effect of population size and the number of generations these values are also tested in parameter setting experiments. For the determined values a set which contains sixty experiments is considered for parameter setting tests. These experiment parameters are given in the Appendix-B in Table B.1.

These experiments are done for a set of fifteen problems which contains five problems of projects with fifteen activities, five problems of projects with twenty-five activities and five problems of projects with thirty-five activities.

After solving the problems with three algorithms (VEGA Strategy 1, Strategy 2 and proposed GA) the statistical testing is done on the EHR values.

For statistically testing the significance Systat© is used. As the method to determine the significance one-way ANOVA is used. In one-way ANOVA the experiments are taken as factors and the corresponding EHR values for problems are taken as dependent variables. This test determines whether the means of the experiments are different at a statistically significant level. Based on the results of the tests the parameters of GAs are chosen.

For VEGA strategies, after applying ANOVA, the results of experiments did not differ statistically. So, the best average valued set is chosen for further experimentation.

For the proposed GA the results differ at a confidence level of 5%. This shows that results are different but the different samples are identified on pairwise comparison. Since it is not possible to compare all sixty experiment results, the best average valued set is chosen for the proposed GA.

The chosen parameters for the algorithms are given in Table 5.1.

Table 5-1 Parameters chosen for GAs

	VEGA Strategy 1	VEGA Strategy 2	Proposed GA
Probability of Crossover	0.30	0.75	0.15
Probability of Mutation	0.60	0.15	0.75
Generation Size	100	100	250
Population Size	500	500	200

Although a conservative bound of fifty thousand evaluations has been determined for the chromosomes to be generated, the parameter setting experiments showed that as the size of the search space increases further exploration is needed. After determining this need, an increase for the number of chromosomes to be generated has been applied. Table 5.2 shows the population size and generation size for the problem groups and for the different algorithms. As can be observed in Table 5.2, as the number of activities increases, so does the number of evaluations performed in each GA. But the number of evaluations given in Table 5.2 does not represent the real number of evaluations for the proposed GA. For the proposed GA, the number of evaluations is less than the number of evaluations given in Table 5.2 because the proposed GA uses elitism. The individuals that are nondominated in a generation are carried into the next generation, so the number of individuals generated by the operators is less than the population size for

each generation. Depending on the problem the number of individuals evaluated is nearly 20 percent less than the number given in Table 5.2.

Table 5-2 Population size and generation sizes for different problem groups and for different algorithms

Number of Activities in Problem	VEGA Strategy 1		VEGA Strategy 2		Proposed GA	
	Pop. Size	Gen. Size	Pop. Size	Gen. Size	Pop. Size	Gen. Size
15	500	100	500	100	200	250
25	500	150	500	150	200	375
35	500	200	500	200	200	500

5.3. Comparison of GAs

The GAs that are used to solve the problem are tested on 60 problems. These 60 problems include equal number of problems consisting of fifteen, twenty-five and thirty-five activities.

For comparing GAs, EHR is used. EHR is calculated for each of the problems. EHR may be used for comparing the algorithms but it does not evaluate the performance of the algorithm with the true Pareto front. Since an exact evaluation for the algorithm is needed an approximation of the true Pareto front is found and proposed GA is compared with it.

5.3.1. Comparison with the Approximation of the True Pareto Front

The approximation of the true Pareto front is done by using the mathematical programming software GAMS©. For approximating the true Pareto front, beginning from the maximum makespan the makespan objective is added to the model as a constraint. Since makespan is added as a constraint to the model, the model becomes a single objective model.

First, maximum makespan is taken as the constraint and the model is solved for the objective of cost minimization. Then, the result of the solved model is taken and the makespan is decreased by 0.01 from the result level and added as constraint again. This procedure is repeated until the minimum makespan reached.

The approximation of the true Pareto front founded by the GAMS© and the results gained from the 60 experiments of the proposed GA runs are plotted on the graphs. These figures showed that proposed GA is comparable to the GAMS© solution procedure. The following three figures (Figures 5.2 - 5.4) show the results that are obtained from these comparisons.

Figure 5.2 shows that the GA results are comparable to approximation of the true Pareto front.

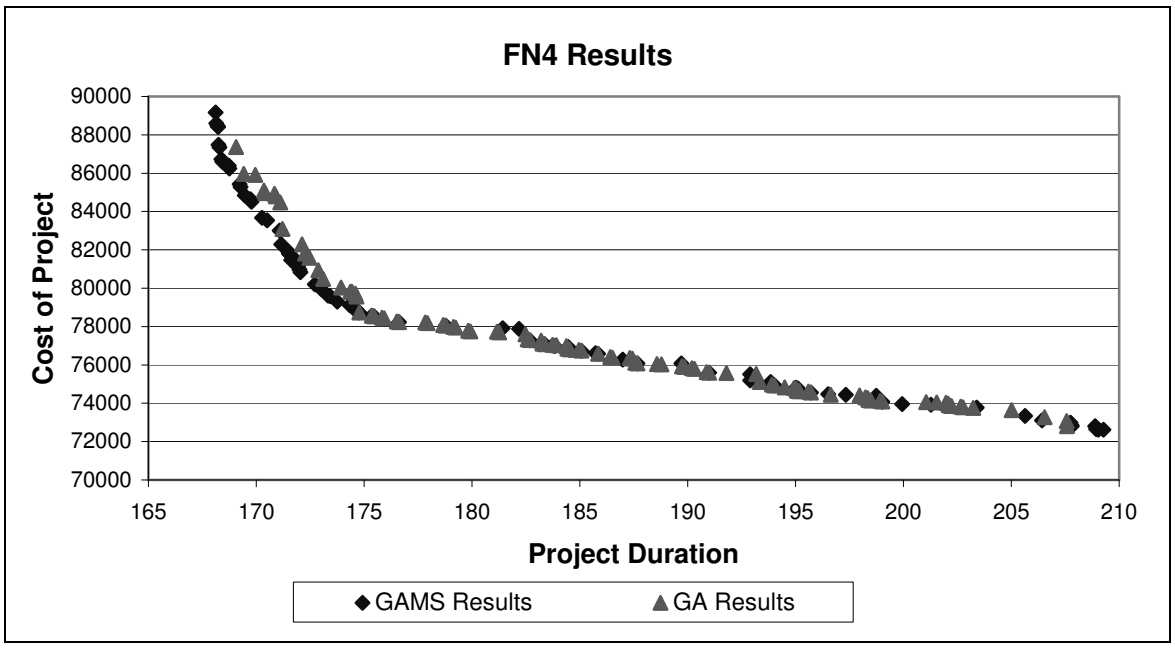

Figure 5-2 Comparison of proposed GA results with approximation of true Pareto front

In Figure 5.3, the GA results perform better than GAMS© results. This seems impossible but since GAMS© has some tolerances to stop the search for some problems this may be possible. Also decreasing the makespan constraint by 0.01 may lead to skip some solutions in between.

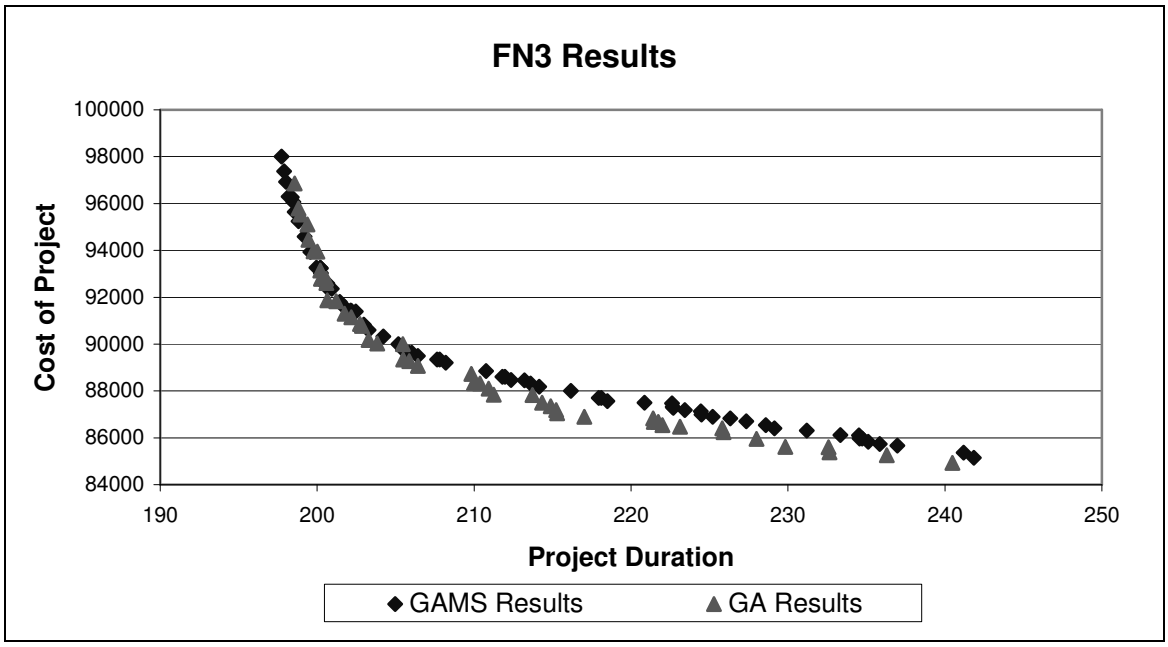

Figure 5-3 Comparison of proposed GA results with approximation of true Pareto front

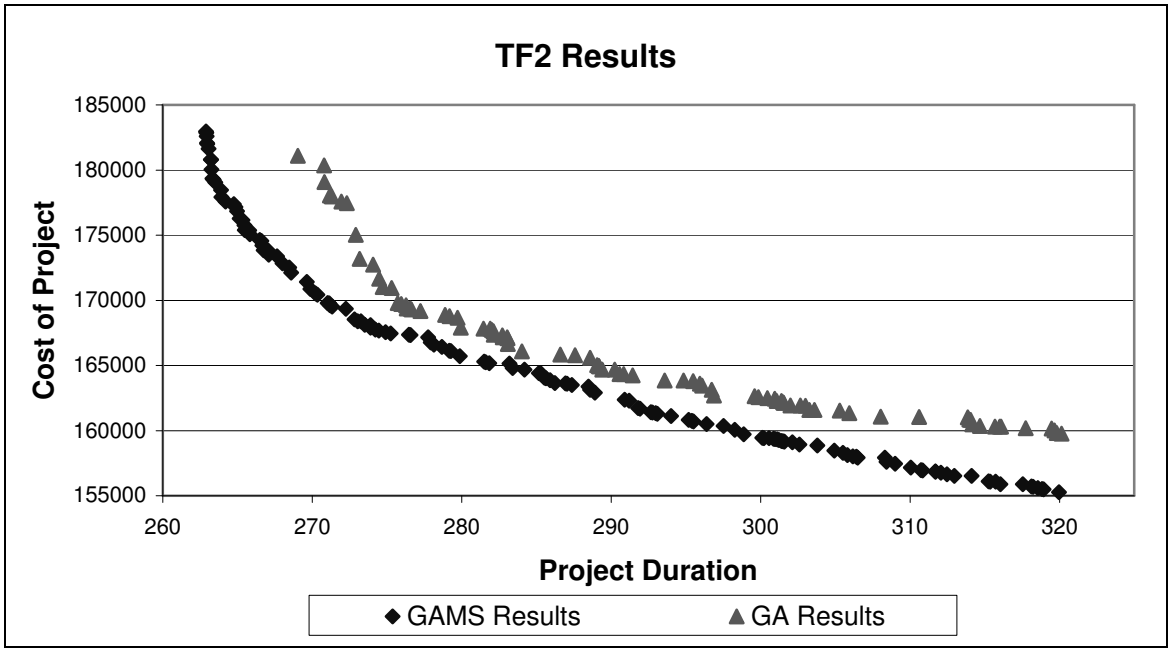

Figure 5-4 Comparison of proposed GA results with approximation of true Pareto front

Figure 5.4 shows that the approximation of the true Pareto front may also dominate the GA results. The problem results shown in this graph belongs to a problem with thirty-five activities. On the other hand, Figures 5.2 and 5.3 are the results of problems with fifteen activities. This may lead to the conclusion that as the size of the problem increases the need for exploration in GA increases.

The EHR values for the approximation of the true Pareto front are also calculated. The average EHR value for the true Pareto front approximation is 0.2573. This value is very close to the average EHR value of proposed GA, which is 0.2404. When the EHR values are investigated one by one for all the problems, most of the time EHR value of the approximation is too close to the proposed GA's EHR value. These values are given in Table A.2 in the Appendix-B.

When the EHR values of the approximation of the true Pareto front and GAs are statistically analysed, it is observed that they differ in a statistically significant way. In Table 5.3, percent deviations of GAs from the approximation of the true Pareto front are given.

The values in Table 5.3 show that VEGA Strategies 1 and 2 lead to similar results when compared to the approximation of true Pareto front. The results of the proposed GA are significantly better than the VEGA strategies and comparable to the approximation of true Pareto front.

Table 5-3 Percent deviations of the GAs from the approximation of true Pareto front

Problem Type	VEGA Strategy 1	VEGA Strategy 2	Proposed GA
Overall	24.26	23.67	6.44
15 Activities	18.50	19.05	5.62
25 Activities	23.88	21.82	4.74
35 Activities	30.40	30.14	8.97

5.3.2. Pairwise Comparison of GAs

GA results are compared according to the runs that are made on 60 problems with the selected parameters. For each problem-GA pair the EHR values are calculated and these values are used to compare the algorithms. EHR values for each problem and the algorithm is given in Table B.2 in the Appendix-B. Average EHR values for the problem classes are given in Table 5.4.

Table 5-4 EHR values for problem classes

Problem Type	VEGA Strategy 1	VEGA Strategy 2	Proposed GA
Overall	0.1951	0.1964	0.2404
15 Activities	0.2177	0.2161	0.2522
25 Activities	0.1918	0.1968	0.2395
35 Activities	0.1757	0.1763	0.2295

The ANOVA is applied to the GA results, which shows that these three algorithms differ at zero confidence level. This result leads us to make pairwise comparisons.

5.3.2.1. Comparison of VEGA Strategies

For comparing VEGA strategies the EHR values of 60 problems are used. These values are used to decide whether the strategies differ in a statistically significant way. The results show that the difference of these two strategies is not statistically significant. The mean values for these strategies are also too close to differentiate. The mean EHR values for Strategy 1 and Strategy 2 are 0.1951 and 0.1964 respectively.

5.3.2.2. Comparison of VEGA Strategy 1 with the Proposed GA

When the VEGA Strategy 1 is compared with the proposed GA, the results show that proposed GA is better than the VEGA Strategy 1. The applied hypothesis test is also a proof of this statement. These two algorithms differ at zero confidence level. Their mean EHR values are 0.1951 and 0.2404 for VEGA Strategy 1 and proposed GA respectively.

5.3.2.3. Comparison of VEGA Strategy 2 with the Proposed GA

The applied hypothesis test showed that, up to zero confidence level these two algorithms have different results. The mean EHR values are 0.1964 and 0.2404 for

VEGA Strategy 2 and proposed GA respectively. So we can conclude that proposed GA performs better than VEGA Strategy 2.

5.4. Comparison of the Improvement Heuristics

The improvement heuristics are implemented at the end of the proposed algorithm and the nondominated solutions found by the algorithm are further improved by these heuristics. The results for the improvement heuristics are compared according to two criteria: Average value of the improvement (expected cost decrease) and the ratio of the number of improved solutions to the number of nondominated solutions found.

Average value of improvement (AI) is the average of expected cost decrease from the nondominated GA solution level to improvement heuristic result level expressed in percentage.

$$AI = (\sum_{i=1}^{l} \frac{C_i^{GA} - C_i^{H}}{C_i^{GA}}) * 100 / l \qquad (5.1)$$

where l is the number of Pareto optimal solutions, C_i^{GA} is the expected cost of i^{th} Pareto optimal solution in which GA resulted, C_i^{H} is the expected cost after the improvement heuristic is applied to the Pareto solution i.

5.4.1. FS Improvement Heuristic Results

The early computational results showed that FS improvement heuristic almost surely does not improve the quality of the results. The solutions of the thirty problems are analyzed and FS improvement heuristic has not improve any of the solutions. Based on these unsatisfactory results, further investigation of this heuristic is terminated at this point.

The results for CCDM improvement heuristic and GAB improvement heuristic are given in Table B.3 in the Appendix-B.

5.4.2. CCDM Improvement Heuristic Results

This improvement heuristic has improved most of the solutions. On the other hand, since an exact linear underestimator to the modes of the problem has not been used, some of the expected cost values increased instead of decreasing. Table 5.5 shows the overall results and the results according to problem classes.

Table 5-5 Result summary of CCDM improvement heuristic

Problem Type	Number of Nondominated Solutions	Average Improvement (%)	Number of Solutions Improved	Ratio of Improved Solutions (%)
Overall	2546	0.50	1948	76.51
15 Activities	717	0.16	422	58.86
25 Activities	848	0.63	693	81.72
35 Activities	981	0.71	833	84.91

The results show that as the problem size increases the performance of the improvement heuristic gets better. This may be the result of the deteriorating performance of GA as the problem size increases. Since GA can not explore the search space adequately, there remains more area for the improvement heuristic.

5.4.3. GAB Improvement Heuristic Results

GAB improvement heuristic has improved more solutions than the CCDM improvement heuristic but the quality of the results differ slightly. Table 5.6 represents the performance of the GAB improvement heuristic.

Table 5-6 Result summary of GAB improvement heuristic

Problem Type	Number of Nondominated Solutions	Average Improvement (%)	Number of Solutions Improved	Ratio of Improved Solutions (%)
Overall	2546	0.27	2039	80.09
15 Activities	717	0.19	421	58.72
25 Activities	848	0.32	723	85.26
35 Activities	981	0.30	895	91.23

This improvement heuristic has not decreased the expected cost of the solutions as much as the CCDM improvement heuristic but the number of solutions improved are higher for this heuristic, and hence the ratio of improved solutions.

The average improvement values of the GAB improvement heuristic are not as high as the CCDM improvement heuristic. This may be the result of the starting points of the two improvement heuristics. Since GAB improvement heuristic starts from the GA result, which may be a local optimum, the GAB improvement heuristic might not be able to move away from this local optimum.

It might be conjectured that CCDM improvement heuristic may be more effective if a more precise piecewise linear underestimator is used. But as the precision of the estimator increases, the effort to generate the underestimator and to solve the continuous model will increase. This leads to a trade-off to be resolved between the computational cost and decreased cost by the CCDM improvement heuristic.

When considering the improvement results of both heuristics these results may seem unsatisfactory and thus the improvement heuristics may seem unnecessary. The improvement heuristics serve the purpose of finding the cost savings associated with some noncritical activities in the final solution and thus avoiding a trivially inferior solution. The risks associated with some noncritical activities might have been reduced at a cost. But expected duration of these activities might be further increased without affecting the critical path length. Thus savings can be realized by reducing the states of the associated risks and allowing for longer expected activity duration.

5.5. Computational Times of the Study

The computational times of the GAs, CCDM improvement heuristic and true Pareto front approximation (TPFA) are given in Table 5.7. These values are the average of five problems' computational times from each problem class. The computational times of GAB improvement heuristic cannot be given since they are very small and thus cannot be measured accurately.

Table 5-7 Computational times of the study in milliseconds

Problem Type	Vega Strategy 1	Vega Strategy 2	Proposed GA	TPFA	CCDM
15 Activities	1768	3807	6339	167075	17588
25 Activities	4472	7658	10094	443936	23385
35 Activities	8907	12964	16291	849036	27976

As it is clearly seen from the Table 5.7 the computational times of the TPFA are very high compared to GAs. TPFA is computationally costly in these relatively small problems. For big problems it may be very costly to generate the TPFA because of excessive computational time and limitations of mathematical programming softwares.

When GAs are compared, it is clear that VEGA strategies have smaller computational times compared to the proposed GA. As it is seen from the Table 5.7 as the problem size increases, the difference between computational times of VEGA strategies and the proposed GA decreases.

6. CONCLUSION AND FUTURE RESEARCH DIRECTIONS

6.1. Conclusion

According to limited computational study made on three different sets of problems, the results give the opportunity of giving some conclusions on the performance of the algorithms and improvement heuristics.

The GAs that are used in solving the problem of project scheduling under risk can not be compared with the true Pareto front, since true Pareto front is not known. Thus results are compared with the approximation of the true Pareto front and the results of GAs are compared with each other.

The graphs given for the approximation of true Pareto front and the GA results show that these two are comparable. Even for some cases, GA results in better solutions because of the early termination of GAMS© due to the tolerance employed on the objective function value obtained. These results show that the proposed GA results are comparable to the approximation of the true Pareto front.

When VEGA strategies are compared with each other the results show that these two algorithms did not differ statistically. The hypothesis test applied to the results of these algorithms show that their performances do not differ. Also the means of the results for these two algorithms are too close to differentiate.

The proposed GA seems better than the two VEGA strategies. When pairwise comparisons of VEGA strategies are made with the proposed GA, the hypothesis test results show that the proposed GA performs better than the two VEGA strategies. The mean values and the robust difference between EHR values of these algorithms prove that the proposed GA is better.

When the improvement heuristics are tried to be compared, it is difficult to come up with a performance criterion for comparison. There are three improvement heuristics proposed, one of which has been eliminated because of the unsatisfactory results.

FS improvement heuristic is eliminated from computational study after the unsatisfactory results that are taken from the early computational studies.

The CCDM improvement heuristic and GAB improvement heuristic are compared according to two criteria. One of them is the average improvement for the nondominated solutions and the other is the ratio of solutions improved, i.e., with their expected costs decreased. When these two criteria are taken together it is impossible to compare the results. When the performance criteria are taken one by one, it then becomes possible to compare. The CCDM improvement heuristic is better when the results are compared according to the average improvement. When the heuristics are compared according to the ratio of solutions improved, the GAB improvement heuristic seems to perform better.

6.2. Future Research Directions

Two possible research directions are proposed here. One of them is related with the problem formulation and the other is related with the solution approach.

6.2.1. Solution Approach Related Future Research

There may be different approaches other than GAs for solving multiobjective optimization problems. Those methods will be the other metaheuristic methods which need posteriori preference articulation. Also the methods with priori preference articulation and the methods with progressive preference articulation may be used, in case, real problem data and decision maker preference data are available.

The other future work topic may be the use of GAs for comparison purposes, for which promising results have been reported. One such algorithm is NSGA-II. Comparing the proposed algorithm with NSGA-II is another future research direction.

For the problem solved, another solution approach may be using the modes instead of risks to select. For this approach first the nondominated modes will be identified then the search may be done along these modes. This approach may also decrease the size of search space and increase the quality of the results.

6.2.2. Problem Formulation Related Future Research

For problem formulation related future research there are two possible directions. The first is related with the content of problem. The second direction is related with an extension of the formulation.

The problem formulation may be extended by using dependent risks instead of independent risks. The risks may be replaced with the risks that affect other risks. This formulation will be more realistic to represent the real life situations.

Adding resource constraints to the formulation is another topic that will make the problem formulation more realistic.

The impacts and probability of occurrences are assumed to be discrete in the problem formulation. However, these may be formulated using continuous functional forms.

REFERENCES

1. Bagchi, T,P., *Multiobjective Scheduling by Genetic Algorithms.* Kluwer Academic Publishers, Boston, 1999.
2. Coello Coello, C.A., An Updated Survey of GA-Based Multiobjective Optimization Techniques. ACM Computing Surveys, Vol: 32-2, 109-143, 2000.
3. Coello Coello, C.A.; Christiansen, A.D., MOSES : A Multiobjective Optimization Tool for Engineering Design. *Engineering Optimization,* Vol: 31, 337-368, 1999.
4. Coello Coello, C.A.; Van Veldhuizen, D.A.; Lamont, G.B., *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York, 2002.
5. De, P.E.; Dunne, J.; Ghosh, J.B.; Wells, C.E., Complexity of the Discrete Time-Cost Tradeoff Problem for Project Networks. *Operations Research,* Vol: 45-2, 302-306, 1997.
6. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T., A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation,* Vol: 6-2, 182-197, 2002.
7. Demeulemeester, E.L., Herroelen, W.S., Elmaghraby, S.E., Optimal Procedures for the Discrete Time/Cost Tradeoff Problem in Project Networks. *European Journal of Operational Research,* Vol: 88, 50-68, 1996.
8. Ehrgott, M., *Multicriteria Optimization*. Springer Verlag, Berlin, 2000.
9. Elmaghraby, S.E., Activity Nets: A Guided Tour Through Some Recent Developments. *European Journal of Operational Research,* Vol: 82, 383-408, 1995.
10. Fonseca, C.M.; Fleming, P.J.; An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation,* Vol: 3-1, 1-16, 1995.
11. Fulkerson, D.R., A Network Flow Computation for Project Cost Curves. *Management Science,* Vol:7, 167-178, 1961.
12. Goyal, S.K., A Note on "A Simple CPM Time/Cost Tradeoff Algorithm". *Management Science,* Vol: 21-6, 718-722, 1975.
13. Hartmann, S., *Project Scheduling Under Limited Resources,* Springer Verlag, Berlin, 1999.
14. Horn, J.; Nafpliotis, N.; Goldberg, D.E., A Niched Pareto Genetic Algorithm for Multiobjective Optimization, in *Proceedings of the First IEEE Conference on Evolutionary Computation,* edited by Michalewicz, Z., IEEE Press, Piscataway NJ, 1994.

15. Jones, D.F.; Mirrazavi, S.K.; Tamiz, M., Multiobjective Metaheuristics: An Overview of the Current State-of-the-Art. *European Journal of Operational Research,* Vol: 137-1, 1-9, 2002.

16. Knowles, J.; Corne, D., On Metrics for Comparing Non-Dominated Sets. In *Congress on Evolutionary Computation (CEC'2002)*, Vol: 1, 711-716, IEEE Service Center, Piscataway, New Jersey, 2002.

17. Kolisch, R.; Padman, R., An Integrated Survey of Deterministic Project Scheduling. *Omega,* Vol: 29, 249-272, 2001.

18. Périaux, J.; Sefriou, M.; Mantel, B., GA Multiple Objective Optimization Strategies for Electromagnetic Backscattering, in *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science,* edited by Quagliarella *et al.*, John Wiley & Sons, Chichester, 1998.

19. Quagriella, D.; Vicini, A., Coupling Genetic Algorithms and Gradient Based Optimization Techniques, in *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science,* edited by Quagliarella *et al.*, John Wiley & Sons, Chichester, 1998.

20. Robinson, D.R., A Dynamic Programming Solution to Cost-Time Tradeoff for CPM. *Management Science,* Vol: 22-2, 158-166, 1975.

21. Romero, C., *Handbook of Critical Issues in Goal Programming,* Pergamon Press, Oxford, 1991.

22. Siemens, N., A Simple Time/Cost Tradeoff Algorithm. *Management Science,* Vol: 17-6, 354-363, 1971.

23. Ulusoy, G., Proje Planlamada Kaynak Kısıtlı Çizelgeleme, in *Yöneylem Araştırması Halim Doğrusöz'e Armağan,* edited by Erkip, N., Köksalan M.; ODTÜ Basım İşliği, Ankara, 2002.

24. Van Veldhuizen, D.A.; Lamont, G.B., Multiobjective Evolutionary Algorithms: Analyzing The State-Of-The-Art. *Evolutionary Computation,* Vol: 8-2, 125-147, 2000a.

25. Wei, C.C.; Wang, C.M.F., Efficient Approaches of Linearization in Project Compression. *Computers & Industrial Engineering,* Vol: 44, 695-706, 2003.

26. Zitzler, E.; Thiele, L., Multiobjective Evolutionary Algorithms: A Comparative Case Study and Strength Pareto Approach, *IEEE Transactions on Evolutionary Computation,* Vol: 3-4, 257-271, 1999.

27. Zitzler, E.; Laumanns, M.; Thiele, L., SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Zurich, Switzerland, 2001.

REFERENCES NOT CITED

1. Brooke, A.; Kendrick, D.; Meeraus, A.; Raman, R.; GAMS© A User's Guide. GAMS Development Corporation, Washington, 1998.
2. Chang, C.K.; Christensen, M.J.; Zhang, T., Genetic Algorithms for Project Management. *Annals of Software Engineering,* Vol: 11, 107-139, 2001.
3. Elmaghraby, S.E., On Criticality and Sensitivity in Activity Networks. *European Journal of Operational Research,* Vol: 127, 220-238, 2000.
4. Gardiner, P.D.; Stewart, K., Revisiting the Golden Triangle of Cost, Time and Quality: The Role of NPV in Project Control, Success and Failure. *International Journal of Project Management,* Vol:18, 251-256, 2000.
5. Goldberg, D.E.; *Genetic Algorithms in Search, Optimization & Machine Learning,* Addison Wesley Longman Inc., Reading, Massachusetts, 1989.
6. http://www.lania.mx/~ccoello (November, 2002)
7. http://gal4.ge.uiuc.edu/pubarch.html (November, 2002)
8. http://gal4.ge.uiuc.edu/technreprts.html (November, 2002)
9. Klein, R., *Scheduling of Resource Constrained Projects.* Kluwer Academic Publishers, New York, 1999.
10. Laumanns, M.; Thiele, L.; Deb, K.; Zitzler, E., Combining Convergence and Diversity in Evolutionary Multiobjective Optimization, *Evolutionary Computation,* Vol: 10-3, 1-21, 2002.
11. Phillips, S., Project Management Duration/Resource Tradeoff Analysis: An Application of the Cut Search Approach. *Journal of the Operational Research Society,* Vol: 47, 697-701, 1996.
12. Vanhoucke, M.; *Exact Algorithms for Various Types of Project Scheduling Problems.* PhD Dissertation, Katholieke Universiteit Leuven, Leuven, 2000.
13. Van Veldhuizen, D.A.; Lamont, G.B., On Measuring Multiobjective Evolutionary Algorithm Performance. In *2000 Congress on Evolutionary Computation*, vol.1, pp. 204-211, IEEE Service Center, Piscataway, New Jersey, 2000b.
14. Zitzler, E.; Thiele, L.; Deb, K., Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation,* Vol: 8-2, 173-195, 2000.

APPENDIX - A

```
Initialize population
Evaluate objective values
Assign rank based on pareto dominance
Compute niche count
Assigned linearly scaled fitness
For i=1 to G
    Selection via stochatic universal sampling
    Single point crossover
    Mutation
    Evaluate objective values
    Assign rank based on pareto dominance
    Compute niche count
    Assign linearly scaled fitness
    Assign shared fitness
End loop
```

Figure A-1 MOGA pseudocode

```
Initialize population
Evaluate objective values
Assign rank based on pareto dominance in each wave
Compute niche count
Assigned shared fitness
For i=1 to G
    Selection via stochatic universal sampling
    Single point crossover
    Mutation
    Evaluate objective values
    Assign rank based on pareto dominance in each wave
    Compute niche count
    Assign shared fitness
End loop
```

Figure A-2 NSGA pseudocode

```
Initialize population
    Generate random population – size M
    Evaluate objective values
    Assign rank based on pareto dominance – "sort"
    Generate child population
        Binary tournament selection
        Recombination and mutation
For i=1 to G
    With parent and child population
        Assign rank based on pareto dominance – "sort"
        Generate Sets of nondominated fronts
        Loop (inside) by adding solutions to next generation starting from the
        "first" front until M individuals found
        Determine crowding distance between points on each front
    Select points (elitist) on the lower front (with lower rank) and are outside a
    crowding distance
    Create next generation
        Binary tournament selection
        Recombination and mutation
    Increment generation index
End loop
```

Figure A-3 NSGA-II pseudocode

```
Initialize population
Evaluate objective values
For i=1 to G
    Specialized binary tournament selection
        Only candidate 1 dominated : Select candidate 2
        Only candidate 2 dominated : Select candidate 1
        Both candidates dominated or both not dominated :
            Perform specialized fitness sharing
                Return candidate with lower niche count
    Single point crossover
    Mutation
    Evaluate objective values
End loop
```

Figure A-4 NPGA pseudocode

```
Initialize population
Evaluate objective values
For i=1 to G
    Specialized binary tournament selection
    Using degree of domination as rank
    Only candidate 1 dominated : Select candidate 2
    Only candidate 2 dominated : Select candidate 1
    Both candidates dominated or both not dominated
        Perform specialized fitness sharing
            Return candidate with lower niche count
    Single point crossover
    Mutation
    Evaluate objective values
End loop
```

Figure A-5 NPGA-II pseudocode

```
Initialize population
Create empty external set E
For i=1 to G
    Copy nondominated members of P to E
    Remove elemets from E which are covered by any other member of E
    Prune E (using clustering) when the maximum capacity of E has been exceeded
    Compute fitness of each individual in P and in E
    Use binary tournament selection with replacement to select individuals from P+E
    (multiset union) until the mating pool is full
    Apply crossover and mutation
End loop
```

Figure A-6 SPEA pseudocode

```
Initialize population
Create empty external set E
For i=1 to G
    Compute fitness of each individual in P and E
    Copy all nondominated individuals in P and E to E
    Use the truncation ooperator to remove elements from E when the capacity of the
    file has been extended
    If the capacity of E has not been exceeded then use dominated individuals in P to
    fill E
    Perform binary tournament selection with replacement to fill the mating pool
    Apply crossover and mutation to the mating pool
End loop
```

Figure A-7 SPEA-II pseudocode

```
Repeat
Initialize single population parent p & add to archive                (line 2)
Mutate p to produce child c and evaluate fitness
    If p dominates c discard c
    Else if c dominates p
        Replace p with c, and add c to archive
    Else if (if c is dominated by any member of the archive)
        Discard c
    Else apply test (p, c, archive) to determine which becomes the new
    current solution and whether to add c to the archive
Until a termination criterion is true, return to line 2.
```

Figure A-8 PAES pseudocode

```
Generate a random (internal) population PI
Evaluate each member of PI
Initialize the external population PE to the empty set
While termination criterion has not been met
    Incorporate nondominated vecors from PI into PE
    Delete the current contents of PI
    Repeat
        With probability  Pc, select two parents from PE
        Produce a single child with crossover
        Mutate the child created in previous step
        With probability (1-Pc), select one parent
        Mutate the selected parent to produce a child
    Until the population PI is filled
End while
Return the members of PE as the result
```

Figure A-9 PESA pseudocode

APPENDIX - B

Table B-1 Experiment parameters used in parameter setting tests

Experiment No	Population Size	Number of Generations	Probability of Mutation	Probability of Crossover
1	100	500	0.15	0.15
2	100	500	0.30	0.15
3	100	500	0.45	0.15
4	100	500	0.60	0.15
5	100	500	0.75	0.15
6	100	500	0.15	0.30
7	100	500	0.30	0.30
8	100	500	0.45	0.30
9	100	500	0.60	0.30
10	100	500	0.15	0.45
11	100	500	0.30	0.45
12	100	500	0.45	0.45
13	100	500	0.15	0.60
14	100	500	0.30	0.60
15	100	500	0.15	0.75
16	200	250	0.15	0.15
17	200	250	0.30	0.15
18	200	250	0.45	0.15
19	200	250	0.60	0.15
20	200	250	0.75	0.15
21	200	250	0.15	0.30
22	200	250	0.30	0.30
23	200	250	0.45	0.30
24	200	250	0.60	0.30
25	200	250	0.15	0.45
26	200	250	0.30	0.45
27	200	250	0.45	0.45
28	200	250	0.15	0.60
29	200	250	0.30	0.60
30	200	250	0.15	0.75
31	250	200	0.15	0.15
32	250	200	0.30	0.15
33	250	200	0.45	0.15
34	250	200	0.60	0.15
35	250	200	0.75	0.15

Table B-1 Experiment parameters used in parameter setting tests (cont'd)

Experiment No	Population Size	Number of Generations	Probability of Mutation	Probability of Crossover
36	250	200	0.15	0.30
37	250	200	0.30	0.30
38	250	200	0.45	0.30
39	250	200	0.60	0.30
40	250	200	0.15	0.45
41	250	200	0.30	0.45
42	250	200	0.45	0.45
43	250	200	0.15	0.60
44	250	200	0.30	0.60
45	250	200	0.15	0.75
46	500	100	0.15	0.15
47	500	100	0.30	0.15
48	500	100	0.45	0.15
49	500	100	0.60	0.15
50	500	100	0.75	0.15
51	500	100	0.15	0.30
52	500	100	0.30	0.30
53	500	100	0.45	0.30
54	500	100	0.60	0.30
55	500	100	0.15	0.45
56	500	100	0.30	0.45
57	500	100	0.45	0.45
58	500	100	0.15	0.60
59	500	100	0.30	0.60
60	500	100	0.15	0.75

Table B-2 EHR values according to problem and algorithm, true Pareto front approximation (TPFA)

Problem	VEGA Strategy 1	VEGA Strategy 2	Proposed GA	TPFA
FN1	0.2579	0.2474	0.2844	0.3022
FN2	0.2449	0.2487	0.2854	0.3082
FN3	0.2270	0.2227	0.2663	0.2842
FN4	0.2463	0.2381	0.2812	0.2878
FN5	0.2161	0.2186	0.2426	0.2519
FN6	0.2063	0.2050	0.2277	0.2293
FN7	0.1836	0.1982	0.2522	0.2605
FN8	0.1980	0.1945	0.2195	0.2278
FN9	0.1995	0.1939	0.2281	0.2333
FN10	0.1943	0.1995	0.2339	0.2387
FN11	0.1843	0.1846	0.2054	0.2125
FN12	0.2241	0.2178	0.2566	0.2768
FN13	0.2175	0.2057	0.2770	0.2868
FN14	0.1791	0.1733	0.2274	0.2671
FN15	0.2243	0.2283	0.2741	0.2933
FN16	0.2572	0.2566	0.2839	0.2898
FN17	0.2431	0.2454	0.2738	0.2832
FN18	0.1888	0.1991	0.2227	0.2421
FN19	0.2608	0.2563	0.2791	0.2957
FN20	0.2014	0.1880	0.2224	0.2805
EB1	0.1554	0.1595	0.2014	0.2027
EB6	0.1909	0.2016	0.2425	0.2600
EB8	0.2153	0.2169	0.2691	0.2857
EB10	0.1707	0.1752	0.2121	0.2235
EB11	0.1951	0.2024	0.2551	0.2720
EB12	0.1924	0.2036	0.2416	0.2493
EB13	0.1976	0.2046	0.2443	0.2642
EB14	0.1688	0.1780	0.2132	0.2244
EB18	0.2090	0.2155	0.2578	0.2652
EB20	0.1903	0.1927	0.2385	0.2430
EB21	0.2092	0.2200	0.2724	0.2726
EB22	0.1739	0.1809	0.2199	0.2224
EB23	0.1663	0.1707	0.2021	0.2055
EB26	0.2175	0.2173	0.2690	0.2873
EB28	0.1831	0.1825	0.2306	0.2516
EB30	0.2682	0.2650	0.2978	0.3348
EB33	0.1912	0.1934	0.2315	0.2497
EB35	0.1891	0.2015	0.2467	0.2599
EB39	0.1881	0.1772	0.2275	0.2344
EB40	0.1633	0.1774	0.2172	0.2316
TF1	0.1657	0.1598	0.2146	0.2307
TF2	0.1836	0.1732	0.2305	0.2475
TF3	0.1581	0.1573	0.2028	0.2359
TF4	0.1785	0.1714	0.2268	0.2475

Table B-2 EHR values according to problem and algorithm, true Pareto front approximation (cont'd)

Problem	VEGA Strategy 1	VEGA Strategy 2	Proposed GA	TPFA
TF5	0.1995	0.1938	0.2562	0.2760
TF6	0.2111	0.2061	0.2705	0.2961
TF7	0.1709	0.1817	0.2263	0.2409
TF8	0.1899	0.1920	0.2369	0.2619
TF9	0.1589	0.1662	0.2273	0.2376
TF10	0.1858	0.1834	0.2432	0.2647
TF11	0.1715	0.1850	0.2444	0.2671
TF12	0.1902	0.1908	0.1911	0.2602
TF13	0.1807	0.1747	0.2396	0.2617
TF14	0.1919	0.1820	0.2536	0.2685
TF15	0.1783	0.1825	0.2318	0.2779
TF16	0.1586	0.1698	0.2214	0.2427
TF17	0.1399	0.1410	0.1930	0.2029
TF18	0.1678	0.1657	0.2154	0.2323
TF19	0.1489	0.1540	0.2116	0.2229
TF20	0.1839	0.1950	0.2522	0.2729

Table B-3 Results of heuristics according to the problem

(AI stands for average improvement, NSI stands for number of solutions improved and RIS stands for ratio of solutions improved.)

Problem	Number of Nondominated Solutions	CCDM Improvement Heuristic			GAB Improvement Heuristic		
		AI (%)	NSI	RIS (%)	AI (%)	NSI	RIS (%)
FN1	38	0.68	36	94.74	0.44	29	76.32
FN2	38	0.10	24	63.16	0.01	18	47.37
FN3	37	0.28	37	100.00	0.18	35	94.59
FN4	45	-0.85	1	2.22	0.03	40	88.89
FN5	40	0.00	19	47.50	0.00	6	15.00
FN6	31	0.11	14	45.16	0.08	10	32.26
FN7	22	-0.20	6	27.27	0.29	21	95.45
FN8	39	0.14	22	56.41	0.03	11	28.21
FN9	33	0.39	18	54.55	0.34	20	60.61
FN10	39	0.33	25	64.10	0.20	21	53.85
FN11	35	-0.14	6	17.14	0.02	14	40.00
FN12	45	0.05	14	31.11	0.19	20	44.44
FN13	42	0.43	37	88.10	0.42	37	88.10
FN14	10	0.41	10	100.00	0.38	9	90.00
FN15	38	0.52	34	89.47	0.52	34	89.47
FN16	42	0.39	28	66.67	0.14	23	54.76
FN17	30	0.25	21	70.00	0.09	25	83.33
FN18	33	0.02	13	39.39	0.17	15	45.45
FN19	44	0.17	33	75.00	0.08	20	45.45
FN20	36	0.13	24	66.67	0.12	13	36.11
EB1	45	1.18	45	100.00	0.26	45	100.00
EB6	32	1.07	32	100.00	0.34	24	75.00
EB8	35	0.87	35	100.00	0.41	35	100.00
EB10	34	-0.41	12	35.29	0.31	31	91.18
EB11	46	1.12	45	97.83	0.48	36	78.26
EB12	35	0.61	35	100.00	0.31	35	100.00
EB13	46	0.70	40	86.96	0.38	33	71.74
EB14	40	0.29	24	60.00	0.25	40	100.00
EB18	59	0.47	50	84.75	0.38	47	79.66
EB20	45	0.53	30	66.67	0.20	40	88.89
EB21	49	-0.20	20	40.82	0.25	13	26.53
EB22	44	1.10	39	88.64	0.23	32	72.73
EB23	47	0.92	41	87.23	0.47	47	100.00
EB26	52	0.27	46	88.46	0.18	52	100.00
EB28	26	0.43	21	80.77	0.15	16	61.54
EB30	49	0.61	43	87.76	0.33	49	100.00
EB33	32	0.66	28	87.50	0.16	30	93.75
EB35	41	0.87	38	92.68	0.47	38	92.68
EB39	49	0.16	27	55.10	0.45	49	100.00
EB40	42	1.44	42	100.00	0.28	31	73.81

Table B-3 Results of heuristics according to the problem (cont'd)

Problem	Number of Nondominated Solutions	CCDM Improvement Heuristic			GAB Improvement Heuristic		
		AI (%)	NSI	RIS (%)	AI (%)	NSI	RIS (%)
TF1	48	1.12	44	91.67	0.44	48	100.00
TF2	61	1.32	61	100.00	0.13	60	98.36
TF3	44	1.08	44	100.00	0.31	44	100.00
TF4	46	0.48	34	73.91	0.33	46	100.00
TF5	51	0.67	50	98.04	0.24	50	98.04
TF6	37	0.97	35	94.59	0.24	35	94.59
TF7	51	0.18	22	43.14	0.01	14	27.45
TF8	43	0.79	43	100.00	0.28	43	100.00
TF9	43	0.13	22	51.16	0.25	43	100.00
TF10	47	0.03	20	42.55	0.34	47	100.00
TF11	38	1.52	36	94.74	0.51	38	100.00
TF12	63	0.34	46	73.02	0.02	44	69.84
TF13	55	0.45	55	100.00	0.14	30	54.55
TF14	47	0.40	44	93.62	0.49	47	100.00
TF15	54	0.43	46	85.19	0.23	53	98.15
TF16	49	0.97	42	85.71	0.40	49	100.00
TF17	42	0.94	36	85.71	0.42	42	100.00
TF18	56	0.76	55	98.21	0.42	56	100.00
TF19	44	0.71	36	81.82	0.42	44	100.00
TF20	62	0.88	62	100.00	0.36	62	100.00