A GENETIC ALGORITHM FOR BIOBJECTIVE MULTI-SKILL PROJECT SCHEDULING PROBLEM WITH HIERARCHICAL LEVELS OF SKILLS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ELİF GÜRBÜZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

SEPTEMBER 2010

Approval of the thesis:

A GENETIC ALGORITHM FOR BIOBJECTIVE MULTI-SKILL PROJECT SCHEDULING PROBLEM WITH HIERARCHICAL LEVELS OF SKILLS

submitted by **ELİF GÜRBÜZ** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan ÖZGEN _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Nur Evin ÖZDEMİREL _____
Head of Department, **Industrial Engineering**

Assoc. Prof. Dr. Canan SEPİL _____
Supervisor, **Industrial Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Meral AZİZOĞLU _____
Industrial Engineering Dept., METU

Assoc. Prof. Dr. Canan SEPİL _____
Industrial Engineering Dept., METU

Assist. Prof. Dr. Sedef MERAL _____
Industrial Engineering Dept., METU

Assoc. Prof. Dr. Rıfat SÖNMEZ _____
Civil Engineering Dept., METU

Aybeniz YİĞİT _____
Engineering Directorate, ASELSAN

Date: **28.09.2010**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Elif GÜRBÜZ

Signature :

ABSTRACT

A GENETIC ALGORITHM FOR BIOBJECTIVE MULTI-SKILL PROJECT SCHEDULING PROBLEM WITH HIERARCHICAL LEVELS OF SKILLS

Gürbüz, Elif

M.S., Department of Industrial Engineering

Supervisor: Assoc. Prof. Dr. Canan Sepil

September 2010, 80 pages

In Multi-Skill Project Scheduling Problem (MSPSP) with hierarchical levels of skills, there are more than one skill type and for each skill type there are levels corresponding to proficiencies in that skill. The purpose of the problem is to minimize or maximize an objective by assigning resources with different kinds of skills and skill levels to the project activities according to the activity requirements while satisfying the other problem dependent constraints. Although single-objective case of the problem has been studied by a few researchers, biobjective case has not been studied yet. In this study, two objectives, which are the makespan and the total skill wasted, are taken into account and while trying to minimize the makespan, minimizing the total skills wasted is aimed. By the second objective, overqualification for the jobs is tried to be minimized in order to prevent job dissatisfaction. The biobjective problem is solved using a Multiobjective Genetic Algorithm, NSGA-II. The results of the proposed algorithm are compared with the GAMS results for small-sized problems and with the random search for larger problem sizes.

Keywords: Multi-skill Project Scheduling Problem with Hierarchical Levels of Skills, Staff Motivation in Project Scheduling, Multiobjective Genetic Algorithms

ÖZ

İKİ AMAÇLI HİYERARŞİK BECERİ SEVİYELERİNE SAHİP ÇOK-BECERİLİ PROJE ÇİZELGELEME PROBLEMİ İÇİN GENETİK BİR ALGORİTMA

Gürbüz, Elif
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Doç.Dr. Canan Sepil

Eylül 2010, 80 sayfa

Hiyerarşik beceri seviyelerine sahip çok-becerili proje çizelgeleme problemlerinde birden fazla beceri çeşidi ve her beceri çeşidi için o becerideki yeterliliğe denk gelen seviyeler bulunmaktadır. Problemin amacı probleme bağlı diğer kısıtlar sağlanırken, aktivitelerin gereksinimlerine göre farklı beceri çeşitlerine ve beceri seviyelerine sahip kaynakların proje aktivitelerine atanmasıyla bir amacın en az seviyede veya en yüksek seviyede tutulmasıdır. Her ne kadar tek amaçlı konular çok az araştırmacı tarafından çalışılmış olsa da, iki amaçlı konu hiç çalışılmamıştır. Bu çalışmada iki amaç, proje süresi ve toplam beceri israfı, göz önüne alınarak proje süresi en az seviyede tutulmaya çalışılırken toplam beceri israfının da en az seviyede tutulması hedeflenmiştir. İkinci amaçla, iş tatminsizliğini engellemek için işler için fazla niteliklilik en az seviyede tutulmaya çalışılmıştır. İki amaçlı problem bir Çok Amaçlı Genetik Algoritma, NSGA-II, kullanılarak çözülmüştür. Önerilen algoritmanın sonuçları küçük boyutlardaki problemler için GAMS sonuçlarıyla, daha büyük boyutlu problemler için rastgele arama ile karşılaştırılmıştır.

Anahtar Kelimeler: Hiyerarşik Beceri Seviyelerine Sahip Çok-Becerili Proje Çizelgeleme Problemleri, Proje Çizelgelemede Çalışan Motivasyonu, Çok Amaçlı Genetik Algoritmalar

To My Unique Mom

ACKNOWLEDGEMENTS

I would like to thank my supervisor Assoc. Prof. Dr. Canan Sepil for her continuous support, guidance, advice and encouragement throughout this study.

I would like to express my deepest thanks to my parents, Aysel Durakoğlu and Doğan Durakoğlu. I am forever grateful to them for their understanding, endless patience, love and encouragement.

Finally, I would like to thank to my husband, Sarper Gürbüz, who shared the burden of the study with me. He constantly believed in and motivated me, so that I complete this study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

In this chapter the project scheduling problem and its different types are briefly mentioned. Then, an overview of the problem that is analyzed in this study is presented and our motivation in defining this specific problem is defined.

1.1 Problem Definition

Project scheduling, which involves the planning of project activities over time by taking some constraints into account in order to minimize or maximize some objective(s), is a crucial task in project management. For this reason, the project scheduling problem has been a popular research area over the years.

In a project, if the activities cannot be performed without using certain resources, and moreover, if the resources are limited, the problem is called Resource-Constrained Project Scheduling Problem (RCPSP). In the classical RCPSP, there are resource availability constraints in addition to precedence relationships between activities. The resources are assumed to be renewable which means that they are available with a constant amount in every period. Each activity in the project should be performed without preemption. The objective of the problem is to finish the project as early as possible (minimize makespan) by scheduling the activities considering the precedence relationships and assigning the required resources to these activities.

RCPSP is well studied in literature and there are many variants of the classical problem. Some of them can be listed as follows:

- activities can be performed in different modes (Multi-Mode Resource-Constrained Project Scheduling Problem)

- activities are allowed to be interrupted (preemption allowed),
- resources are nonrenewable or doubly-constrained,
- different objectives are defined,
- activities have release dates and deadlines.

We can refer the reader to Hartmann and Briskorn (2009) for a more detailed study on variants and extensions of the RCPSP.

In this study, we focus on an extension of Multi-Mode Resource-Constrained Project Scheduling Problem (MRCPSP) which is called Multi-Skill Project Scheduling Problem (MSPSP) with hierarchical levels of skills. In MSPSP with hierarchical levels of skills, the resources are defined as the work force, involving individuals with different skill levels of different skill types. In order to perform an activity, individuals with different skill types and skill levels should come together according to the activity requirements. In other words, activities can be scheduled in multiple modes where each mode corresponds to a subset of resources that meets the activity requirements. However, this problem cannot be solved by using the methods proposed for MRCPSP since there can be too many modes for activities.

MSPSP with hierarchical levels of skills is a difficult problem that is commonly faced in real life. For this reason, French Operational Society (ROADEF), which organizes a challenge on an industrial application every other year, selected this topic as a challenge in 2007. The problem was proposed by France Telecom. In the problem, each activity had a priority and the objective of the problem was to minimize the weighted makespan of each priority type. For the detailed description of the problem see the web page of the French Operational Society.

As in the most of the scheduling problems, by minimizing the weighted makespan France Telecom aimed to achieve quick responses to customers, while implicitly decreasing the costs. However, it did not take the staff motivation into account.

The common mistake made in scheduling problems is to behave workers as machines and do not consider their willingness to do their assigned jobs. France Telecom suffered from this mistake severely as many of its staff took their own lives. The suicides were broadly discussed on televisions and in newspapers. According to France 24 International News web page, there have been 24 suicides among France Telecom's staff in just over 18 months. An interesting comment made after the latest death by a local union leader Patrice Diochet took place in The Australian's web page. He was saying that the firm had failed to draw lessons from the recent suicides, there was no humanity, all they talked about was numbers and workers were treated like sausage meat.

Since workers' job satisfaction has usually not been considered in project scheduling, there should be many people suffering from the same problem all over the world. Therefore, the motivation behind this study is not only proposing a method for MSPSP with hierarchical levels of skills but also offering a schedule by which the staff would not be unfit for the job, which in fact would hopefully increase the motivation of the staff.

In this study a MSPSP with hierarchical levels of skills problem similar to Cordeau et al. (2008), which tied for second place in the ROADEF 2007, is examined. There are a set of tasks, which are labeled i=1,…,N and set of workers, which are labeled j=1,…,W. Each worker is able to do a number of skills. There are q skill domains. Moreover, according to the worker's level of proficiency in a skill domain, a level from 0 to p is assigned to him/her. Level 0 means that the worker has no skill in that domain, whereas level p means that he/she has the highest proficiency. Skills of each worker can be shown as a matrix ($v^j_{\alpha\beta}$) where rows indicate levels (α) and columns indicate skill domains (β). Lower part of the matrix corresponds to higher levels of proficiencies. As an example, when there are three skill domains and three skill levels, a worker with skill vector (2,0,1) can be shown as follows:

$$\begin{matrix} level1 \\ level2 \\ level3 \end{matrix}\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

In the first domain, worker's skill level is 2. Since he/she can perform an activity that requires either level-1 or level-2, both of these rows are filled. In other words, ability for a higher skill level carries over to the lower skill levels. In the second domain, the worker has no skill. Finally, in the last domain he/she has the minimum proficiency (i.e. level-1).

Similarly an activity's requirements can be shown as a matrix ($s^i_{\alpha\beta}$). An example is given below:

$$\begin{matrix} level1 \\ level2 \\ level3 \end{matrix}\begin{bmatrix} 2 & 2 & 1 \\ 2 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

As in the staff skill matrix, in this matrix requirement for a higher skill level carries over to the lower skill levels. In the first domain, two level-2 workers are needed. In the second domain, one level-2 worker and one level-1 worker are needed. In the last domain, only a level-3 worker is required for the activity.

Each activity *i* has a processing time. This duration is constant and does not change if overqualified staff is assigned to the activity. Moreover, there are precedence relationships between activities.

In order to perform an activity, a set of staff that has the skills in the required levels should be assigned to the activity. Although overqualified staff can be assigned to the activities, it is assumed that motivation of the staff decreases as he/she is assigned to a job which is underfit for him/her. In other words, wasted skills matrix for each activity is calculated by subtracting the skill requirement matrix ($s^i_{\alpha\beta}$) of the activity

from the sum of the assigned workers' skill matrices ($\sum_j v_{\alpha\beta}^j$). Then, the total wasted skill in the project is tried to be minimized in order to satisfy person-job fit and prevent job dissatisfaction.

The relation between overqualification and the job dissatisfaction has been broadly studied in the literature. Researches show that person-job fit and overqualification significantly affect job satisfaction and workers' motivation.

Perceived overqualification is defined by Johnson et al. (2002) as the extent to which a worker perceives that he/she (a) possesses surplus job qualifications or (b) has limited opportunities to acquire and use new job-related skills. It is mentioned that perceived overqualification is said to exist when individuals perceive that they possess education, experience, or skills that exceed normal job requirements. Johnson et al. investigated the outcomes of the overqualification in terms of job satisfaction, somatization (a chronic condition in which a person has physical symptoms that are caused by psychological problems, and no physical problem can be found) and organizational commitment. Data drawn from three different samples of employees were used and as expected, overqualification has found to have significant negative relationship with job dissatisfaction and positive relationship with somatization.

The levels of perceived overqualifications of Air Force Institute of Technology (USA) graduates were investigated in M.Sc. Thesis of Hoskins (2003) and the affects of overqualification on their job satisfaction, organizational commitment, and turnover were analyzed. Analysis indicated that the ones who perceive they were overqualified, experience lower levels of job satisfaction and organizational commitment.

Vieira (2005) investigated the relationship between overqualification and job dissatisfaction using the data gathered from the European Community Household Panel. Applicants filled a questionnaire which includes the following question: "Do

you feel that you have skills or qualifications to do a more demanding job than the one you now have?" The ones that responded positively to this question were considered to be overqualified. As for the job satisfaction, applicants were asked to describe their extent of satisfaction by a six-point scale. As a result of the statistical analysis, it was found that overall job satisfaction is adversely affected by perceived overqualification.

There are many other studies about job dissatisfaction which stems from overqualification. That is why in this study, motivation of the workers is considered in terms of their qualification for the jobs they are assigned to in a project.
To sum up, while trying to minimize the makespan of the project, the aim is to minimize the assignment of overqualified staff to the activities of the project simultaneously. These two objectives can be conflicting because while trying to assign the workers according to their skills, makespan will be larger than the assignment made without taking person-job fit into account.

Blazewicz et al. (1983) showed that the RCPSP is a generalization of the classical job shop scheduling problem which belongs to the NP hard optimization problems. Since our problem can be reduced to the RCPSP, it is also NP hard. Due to the complexity of the problem, one of the state-of-the-art metaheuristics in the class of multiobjective genetic algorithms is proposed in order to approximate the pareto-optimal set with a reasonable error.

The organization of the thesis is as follows: In Chapter 2 a mathematical model for the problem is given and the literature review on MSPSP with hierarchical levels of skills, RCPSP and multiobjective genetic algorithms are summarized. In Chapter 3, proposed algorithm for the problem is stated and its convergence and robustness are analyzed. In Chapter 4, the test problems to evaluate the algorithm are described and the computational results are given. Finally, in Chapter 5 the study is concluded.

CHAPTER 2

LITERATURE REVIEW

In this chapter, mathematical model for the biobjective MSPSP with hierarchical levels of skills is given. The literature on MSPSP with hierarchical levels of skills and multiobjective genetic algorithms (MOGA) are reviewed. Moreover, some definitions regarding multiobjective optimization are given.

2.1 Mathematical Model for Biobjective MSPSP with Hierarchical Levels of Skills

Consider a project with N activities which are labeled as i=1,…,N and W workers which are labeled as j=1,…,W. Each activity takes at least one day and k represents the days, k=1,…K. There exist precedence relations between some of the activities. $P_{i'}$ is the set of predecessors of activity i' and activity i' cannot be started before each of its predecessors $i \in P_{i'}$ is completed. The precedence relations can be represented by an activity on node (AoN) network which is assumed to be acyclic. In the network, i_0 is the dummy source node and i_n is the dummy sink node which represents starting and ending time of the project.

Duration of an activity is denoted as d_i . $s^i_{\alpha\beta}$ is the skill requirement matrix of an activity that shows the number of workers with a level of α in domain β required by activity i whereas $v^j_{\alpha\beta}$ is the skill matrix of a worker that shows whether or not a worker has skill of level α in domain β. If the worker has a skill then the corresponding entry of the $v^j_{\alpha\beta}$ matrix is 1, otherwise it is zero.

As for the decision variables, x_{jik} is a binary variable and it is equal to 1 if and only if worker *j* is assigned to activity *i* on day *k*. In order to assign the same workers during the execution of the activity, another binary variable y_{ji} is defined which is equal to 1 if and only if worker *j* is assigned to activity *i*. z_{ik} is the last binary variable in the model that is equal to 1 if activity *i* is completed on day *k*. $e^{i}_{\alpha\beta}$ is the only continuous variable that shows the number of extra skills level of α in skill domain β assigned to activity *i*.

The mathematical model for the problem is as follows:

$$\text{Minimize } \sum_{k} k * z_{nk} \tag{1}$$

$$\text{Minimize } \sum_{i}\sum_{\alpha}\sum_{\beta} e^{i}_{\alpha\beta} \tag{2}$$

subject to

$$\sum_{i} x_{jik} \le 1 \qquad \forall j \in W, \forall k \in K \tag{3}$$

$$x_{jik} \le y_{ji} \qquad \forall i \in N, \forall j \in W, \forall k \in K \tag{4}$$

$$\sum_{k}\sum_{j} x_{jik} = \sum_{j} y_{ji} * d_{i} \qquad \forall i \in N \tag{5}$$

$$\sum_{k} z_{ik} = 1 \qquad \forall i \in N \tag{6}$$

$$\sum_{k} x_{jik} \le \sum_{t=k}^{k+d_i-1} z_{it} \qquad \forall i \in N, \forall j \in W, \forall k \in K \tag{7}$$

$$\sum_{j} v^{j}_{\alpha\beta} y_{ji} - s^{i}_{\alpha\beta} = e^{i}_{\alpha\beta} \qquad \forall i \in N, \forall \alpha \in \{1,...,p\}, \forall \beta \in \{1,...,q\} \tag{8}$$

$$\sum_k k * z_{ik} \leq \sum_k (k - d_{i'}) * z_{i'k} \quad \forall i \in P_{i'} \tag{9}$$

$$x_{jik} \in \{0,1\} \quad \forall i \in N, \forall j \in W, \forall k \in K \tag{10}$$

$$y_{ji} \in \{0,1\} \quad \forall i \in N, \forall j \in W \tag{11}$$

$$z_{ik} \in \{0,1\} \quad \forall i \in N, \forall k \in K \tag{12}$$

$$e^i_{\alpha\beta} \geq 0 \quad \forall i \in N, \forall \alpha \in \{1,...,p\}, \forall \beta \in \{1,...,q\} \tag{13}$$

In the model, there are two conflicting objectives. The first one is minimizing the makespan whereas the second one is minimizing the total wasted skills. Inequality (3) ensures that each worker is assigned to at most one activity in a day. (4) defines the relationship between x_{jik} and y_{ji} variables. The worker is assigned to a job on a day if and only if he/she is assigned to that activity during the whole duration of it (i.e. in terms of y variable). Equality (5) ensures the number of days that a worker assigned to an activity is equal to the activity's duration. Equality (6) obliges each activity to be completed only once. By inequality (7), daily assignments made to the activities are constrained by the completion time of the activities. In other words, the day that a worker assigned to an activity should be within the starting and ending day of the activity. Equality (8) ensures the skill requirements of the activities are satisfied. Moreover, extra skills that are assigned to the activities (i.e. $e^i_{\alpha\beta}$ values) are defined by this equality. (9) enforces the precedence relationships between the activities. Finally, constraints (10)-(13) define the domains of the decision variables.

2.2 Literature Review on MSPSP with Hierarchical Levels of Skills

Although multi-skill project scheduling problem is well studied in the literature, there are very few studies related to hierarchical levels type of it. Classical MSPSP with hierarchical levels of skills, which minimizes makespan with non-preemptive

activities, is considered by Bellenguez and Neron (2005). They propose two lower bounds to evaluate a heuristic method or to be used in a Branch-and-Bound algorithm.

Hurkens (2007) and Cordeau et al. (2008) propose solutions to the France Telecom problem in which the activities are prioritized and the objective is to minimize the ending time of the last activity of each priority type. In this problem, technicians are grouped into teams and teams must stay together on a given day. The teams are kept fixed during a day because technicians are sent to intervention locations and it will be time consuming to come back to central point and mix the teams and also there is limited number of available cars. Activities have duration less than one day, so that a team can be assigned to more than one activity in a day. However, in our problem duration of an activity is an integer number and an activity takes at least one day as in the most of other project scheduling problems. So, there is no need to construct a team and workers are assigned to activities directly. Another difference is outsourcing of activities. In the paper, it is allowed as long as the total budget is not exceeded. Apart from these, the main difference of this problem from ours is that it has a single objective.

Valls et al. (2009) propose a solution to the Skilled Workforce Project Scheduling Problem (SWPSP). This problem is similar to our problem since workers are specialized in one or more knowledge area. Moreover, they are classified as senior, standard and junior which corresponds to skill levels in our problem. They have also more than one objective that deal with the criticality levels of activities, balanced worker loads and efficient assignment of workers. Efficiency increases as a more proficient worker is assigned to the job. Since the activities require only one worker, this is reasonable. However, in our problem activities are more complex and can require more than one worker from different skill domains that have different skill levels. So, our aim is to assign workers close to the needed properties. We assume that assigning an overqualified worker to an activity decreases his/her motivation. Another difference between two problems is the activity durations. In their problem durations of the activities change depending on the worker type. Assigning a senior

(junior) worker makes a reduction (increase) of the standard duration of the activity whereas the duration of activities are constant in our problem. Finally, activities are preemptive depending on the worker's timetables in their problem.

None of these methods considers the motivation of the staff while scheduling the activities. Since regarding the motivation of staff can be as a quality related issue, we can also review some approaches that consider quality issues in RCPSP.

Icmeli-Tukel and Rom (1997) propose a method in order to ensure quality in RCPSP. However, by quality they consider activities' expected rework times and their objective is to minimize sum of the rework times and rework costs.

Haouari and Al-Fawzan (2002) propose a bi-objective model for RCPSP which considers quality in project scheduling. One of the objectives is to minimize makespan while the other one is to maximize weighted sum of slacks of activities. By this way model is more robust and risk of finishing the project after deadline is minimized. This model also considers the quality in terms of activities not the workers directly.

There are other studies that aim to increase the quality in the project such as Vanhoucke (2006), Tareghian and Taheri (2007) and Li and Womer (2008). However, none of them considers the quality in terms of staff motivation.

The only study encountered that takes staff motivation into account in project scheduling is proposed by Oktay (2000). In this M.Sc. Thesis, a form is filled by workers and they write their motivation level for each activity. There are three levels: low, average and high. Then, durations of activities are calculated as fuzzy numbers which are a function of skill level and the motivation levels of the workers and difficulty level of activities. Since filling a form by each worker can take very long time for the projects with many activities, this may not be applicable in the real life problems. For that reason, in this study it is preferred to consider motivation in terms of extra skills and extra skill levels assigned to activities.

As for the solution approaches, firstly approaches proposed for the MSPSP with hierarchical levels of skills is investigated. Then, since there are not any other studies that deal with the problem stated here and MSPSP with hierarchical levels of skills is a generalization of the classical RCPSP for the single objective case, solution approaches used for the classical RCPSP are investigated.

Methods to solve multi-skill project scheduling problem is analyzed by Bellenguez-Morineau (2008) in his Ph.D. Thesis written in French. In the paper which is a summary of the thesis, three methods are mentioned to solve the problem: lower bounds, different heuristics and metaheuristics and finally a branch and bound procedure. Tabu search and two genetic algorithms are proposed as metaheuristics, it is mentioned that genetic algorithms are dominated by the tabu search. However, since paper is just a summary, there is no information about the details of the algorithms.

Hurkens (2007), the winner of the ROADEF 2007 proposed an algorithm based on finding a lower bound by allowing preemption and by estimating the minimum number of workers needed for a job. Then, solution is constructed by the relaxed schedule. Teams that include one worker are built and these teams are assigned to the activities by a matching problem. As a result of the matching problem, a number of teams can be assigned to the activities to meet the skill requirements and outsourcing activities are determined.

Cordeau et al. (2008) proposed an adaptive large neighborhood search (ALNS) for the problem stated in France Telecom. Firstly teams are constructed by a construction heuristic and activities are assigned to each team. Then solution is improved by destroying and repairing it. The results are promising so that the team is tied for the second place in the competition.

As stated by Blazewicz et al. (1983), Resource Constrained Project Scheduling Problem is a generalization of the job-shop scheduling problem and thus is a NP-hard problem. Therefore, although exact algorithms are proposed for the problem, these

methods are not applicable to large sized and highly resource constrained problems. Thus, several heuristic approaches are developed for the problem.

Kolisch and Hartmann (1999) summarize the heuristic algorithms used for the RCPSP. They categorize the heuristics into three classes: priority rule based heuristics, metaheuristic approaches and other heuristics such as truncated branch and bound methods and disjunctive arc methods. The first heuristic methods applied to RCPSP are based on priority rules. In this type of heuristics, a serial or parallel schedule generation scheme and one or more priority rules (e.g. minimum latest finish time and most total successors) are applied to construct one or more schedules (i.e. single pass vs. multi pass methods). Kolisch (1996) gives detailed description of parallel and serial scheduling method for RCPSP and compare their performances.

Metaheuristics approaches that are mentioned by Kolisch and Hartmann (1999) are Simulated Annealing (SA), Tabu Search (TS) and Genetic Algorithms (GA). Since the representation of solution is very important in metaheuristics, the paper summarizes five representations reported in the literature of RCPSP. These are:

1. **Activity List Representation:** A precedence feasible representation of the activities is used and each activity must have a higher index than each of its predecessors. In the solution vector, activity indexes are written. The serial schedule generation scheme can be applied in order to convert the activity list to a feasible schedule. Although parallel generation scheme cannot be used directly, it can be applied with a modification.

2. **Random Key Representation:** Each activity is assigned a number between 0 and 1 and a solution is represented by a vector that shows the assigned numbers. Random keys are used as priority values. Both serial and parallel generation schemes can be used as a decoding procedure. Among the activities whose all predecessors are scheduled, the one with the highest priority value is scheduled by taking the resource constraints into account.

3. **Priority Rule Representation:** For each activity, a priority rule is determined and a solution is represented by a vector that shows the priority rule used for each activity. Some of the priority rules used in the literature are as follows:

    - Greatest Rank Positional Weight (GRPW)
    - Minimum Latest Finish Time (LFT)
    - Minimum Latest Start Time (LST)
    - Minimum Slack (MSLK) etc.

    Again, both serial and parallel generation schemes can be used as a decoding procedure. After selecting the activity, it is scheduled according to the defined priority rule.

4. **Shift Vector Representation:** A nonnegative integer is assigned to an activity and each solution is represented by a vector that shows the assigned numbers to the activities. Start time of an activity is determined by adding the assigned number to the earliest start time of the activity. Since the resource constraints are not taken into account, schedules resulting from this representation can be infeasible. Violation of resource constraints is penalized in order to satisfy feasibility.

5. **Schedule Scheme Representation:** In order to represent a solution four distinct relations are defined. These are conjunctions, disjunctions, parallelity relations and flexibility relations. By satisfying related relations, a schedule scheme is constructed. Then, a heuristic is used as a decoding procedure.

In addition to these five representations, Toklu (2002) proposes a genetic algorithm by using a representation that shows the start times of activities. Since genetic operators result in infeasible solutions, constraint violations are penalized in the proposed approach.

After the metaheuristic approaches, Kolisch and Hartmann (1999) summarize other heuristics that are used in the RCPSP literature. Then, they compare the performances of available priority rule based heuristics, metaheuristics and other heuristics. They conclude that the best heuristics as metaheuristics which use activity list representations.

Moreover, Hartmann (1998) investigates the performances of solution representations in RCPSP by using a genetic algorithm. In his paper, firstly he proposes a genetic algorithm that uses activity list representation. Then, he employs his basic genetic algorithm scheme to the random key representation and priority rule representation. After that, he compares the performance of these three algorithms. He states that the algorithm with the activity list representation outperforms other two algorithms.

Alcaraz and Maroto (2001) propose a robust genetic algorithm for RCPSP by improving the activity list representation. They add a gene, which is called forward-backward gene (f/b gene), at the end of the genotype. While applying a serial generation scheme, f/b gene determines whether the schedule is constructed by forward scheduling or backward scheduling. They compare their algorithm with the -algorithms from the literature and conclude that their algorithm outperforms the best algorithms proposed.

Hartmann and Kolisch (2000) investigate the performance of proposed RCPSP heuristics in detail and they focus on the building blocks (i.e. schedule generation schemes, priority rules, schedule representations, operators and search strategies) and the way these building blocks are applied to the heuristics. Then, they update their survey by adding new approaches in 2006.

Hartmann and Kolisch (2006) give very detailed information about the heuristics proposed in RCPSP. They group the approaches into priority rule-based X-pass methods, classical metaheuristics, non-standard metaheuristics and other heuristics. In addition to the metaheuristics mentioned by Hartmann and Kolisch (2000), which

are SA, TS and GA, first application of ant systems to the RCPSP is stated. In the class of non-standard metaheuristics, local search-oriented approaches and population based approaches are mentioned. As for the other heuristics, an important trend forward-backward improvement method is given. Although there are many different approaches in the literature, the most popular algorithms are said to be genetic algorithms and tabu search. At the end of the survey, all the algorithms are compared by using the same test sets and the same stopping criterion. In order to determine the best approaches, the concept of dominance is used. Again, metaheuristics are found to be the best performing methods. Another result of the comparison is that all the nondominated heuristics use serial schedule generation scheme.

Up to now, solution approaches applied to single objective RCPSP are mentioned. Although RCPSP is a multiobjective problem by its nature, methods are usually developed for the single objective case. For this reason, there is very scarce information about the multiobjective approaches.

Hapke et al. (1998) propose a pareto simulated annealing approach (PSA), which is multiobjective version of SA, in order to solve a multiobjective multi-mode resource constrained project scheduling problem with renewable, non-renewable and doubly constrained resources. The objectives are based on time and cost criteria. After representative sample of approximately non-dominated schedules, they apply an interactive procedure over the sample. In the paper, no results are reported about the proposed PSA.

Viana and Sousa (2000) consider multiobjective versions of both simulated annealing and tabu search in order to solve a RCPSP by minimizing makespan, weighted lateness and the violation of resource constraints. Since the authors cannot find the benchmarks for the multiobjective case, they modify the instances produced by PROGEN, which is a problem generator developed by Kolisch and Sprecher (1997). They conclude that, in general, multiobjective tabu search gives better results.

As mentioned, Haouari and Al-Fawzan (2002) propose a bi-objective model for RCPSP that minimizes makespan while maximizing weighted sum of slacks of the activities. They apply a multiobjective tabu search algorithm with an activity list representation. They also used PROGEN in order to evaluate the performance of several variants of their algorithm and compare the performance of single objective case with the existing algorithms.

Abbasi et al. (2006) propose a simulated annealing algorithm for a multiobjective RCPSP in which the objectives are minimizing makespan and maximizing robustness. In the paper, an evaluation function that is a linear combination of makespan and sum of floating time is used. Initial and best solutions that are found by the proposed algorithm are reported. However, the results are not compared with any of the existing algorithms.

We do not encounter a paper that uses the multiobjective genetic algorithm (MOGA) to solve the multiobjective RCPSP in the literature. However, since genetic algorithms are widely used in the single objective RCPSPs and give promising results, it is decided to use MOGA while solving our problem.

2.3 Definitions for Multiobjective Optimization

In classical optimization problems, there is a single objective function and the aim is to find a solution that optimizes the objective function value. However, most of the real life problems have more than one objective that should be taken into account simultaneously.

Most of the time these objectives are conflicting so that in order to have a better solution in terms of one objective, one should accept a worse solution in terms of the other objective. Therefore, typically there are many solutions that do not outperform each other in all objectives. For this reason, among these solutions the best solution differs from one person to another, in other words it depends on the decision maker.

A multiobjective optimization problem can be simply stated as follows:

Maximize or Minimize $\{ f_1(x), f_2(x), ..., f_m(x) \}$

Subject to $x \in X$

where f_i are objective functions, x is the solution vector and X is the feasible solution space.

Following definitions should be given in order to further investigate multiobjective optimization problems (Deb, 2001):

Domination: A solution s_1 is said to dominate s_2 if and only if

1. s_1 is no worse than s_2 in all objectives
2. s_1 is strictly better than s_2 in at least one objective

Nondominated solutions: Among a set of solutions *P*, the nondominated set of solutions *P'* are those that are not dominated by any member of the set *P*.

Pareto-optimal solutions: The set of all nondominated solutions of entire feasible solution space is called pareto-optimal set or efficient frontier.

For a biobjective min-min problem, which is solvable in the continuous domain, efficient frontier is demonstrated by a bold line in Figure 1.

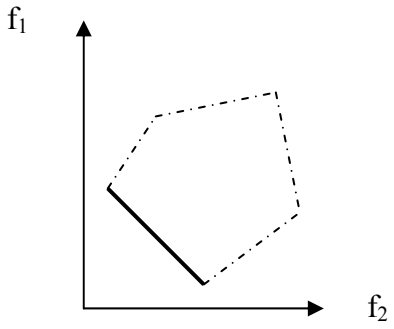

Figure 1 *Efficient frontier of a biobjective problem*

It should be mentioned that there exist pareto-optimal solutions if the objectives are conflicting to each other. If they are not conflicting to each other, there will be only one solution which is optimal for all objectives.

2.4 Methods for Handling Multiobjective Optimization Problems

There are three types of methods for handling multiobjective optimization problems depending on the timing of the decision maker's selection.

1. **A priori methods:** In this method, preference information from the decision maker is taken prior to optimization. For this reason, multiobjective optimization problem turns into single objective prior to optimization. However, it has some disadvantages like difficulty to have sufficient information from the decision maker and non-convexity of problems. Weighted sum method and lexicographic approaches are some examples to this method.

2. **Interactive methods:** This method uses the preference information during the optimization process. During optimization process, some solutions are presented to decision maker and the preference information is asked. Using the obtained information, solution space is reduced. However, it is

burdensome for the decision maker to be involved in the optimization process.

3. **Posteriori methods:** In this method, there is no preference information about the objectives. Pareto-optimal candidate solutions are found and presented to decision maker. Then, the decision maker chooses among the offered solutions. The main disadvantage in this method is the difficulty to converge to pareto-optimal frontier as the number of objectives increase.

As Deb (2001, p.162) states, posteriori methods give an overall perspective of other possible optimal solutions that the underlying multiobjective optimization problem offers before decision maker chooses a solution. Moreover, since our problem has only two objectives, the main disadvantage of posteriori methods is not valid for this study. Therefore, posteriori method is preferred while solving the biobjective MSPSP with hierarchical levels of skills.

2.5 Multiobjective Genetic Algorithm (MOGA) Literature

Genetic algorithms (GAs) are one of the most popular algorithms used as an approximation tool for optimization problems. The primary reasons for their success are their broad applicability, ease of use and global perspective (Goldberg, 1989). They were first proposed by Holland in 1975. They are based on genetic processes of biological organisms. In the nature, the fittest individuals in the population survive and have children. So, they transfer their genetic properties to the next generation. GAs use this idea and are broadly used for difficult problems when there are no specialized techniques.

GAs are initiated by creating an initial population of individuals. Each individual in the population corresponds to a solution to the problem. Initial population can be generated randomly or by using a heuristic method. Each individual is assigned a fitness score such as objective function value. Then, highly fit individuals are crossed together with a crossover probability to have new solutions (offspring). Offspring are

subject to mutation with a mutation probability. Among the parents and/or offspring, best individuals are selected depending on their fitness score. These individuals form the next generation, in other words parent population is updated. These steps are repeated until converging to a good solution.

Some decisions should be made before starting GAs. These are solution representation, fitness function, population size, generation of initial population, parent selection for reproduction, crossover operator, crossover probability, mutation operator, mutation probability, selecting the individuals for the next generation and finally number of generations to be performed.

Since the population of solutions is processed in each iteration and the result is the nondominated solutions in GAs, they are suitable for multiobjective optimization problems, in which the aim is to find all pareto-optimal solutions. There are two desirable features for MOGAs, which are convergence to pareto-optimal set and diversity. Diversity is needed in order to generate a uniformly distributed range of solutions. Without these two features, MOGAs cannot be able to find the entire pareto-optimal set. In order to satisfy these two features many different algorithms are proposed. Deb (2001, p.161-274) classifies these algorithms into two groups: Non-Elitist MOGAs and Elitist MOGAs. Before mentioning the proposed algorithms in these classes, it would be better to define the elitism.

Elitism: In GAs, elitism is satisfied by giving the opportunity to the fittest individuals in the population to be directly carried over to the next generation. After crossing over two parents and mutating the offspring, elitism can be achieved by comparing each offspring with its parents. After comparison, best two solutions are selected. Another way of ensuring elitism is to combine parent and offspring populations at the end of each generation and select the fittest individuals among them up to the population size. By this way, when a promising solution is found, it is kept until a solution better than this solution is found.

Non-Elitist MOGAs: These algorithms do not use elite-preserving operators. Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1985), Vector Optimized Evolution Strategy (VOES) (Kursawe, 1990), Weight-Based Genetic Algorithm (WBGA) (Hajela et al., 1993), Random Weighted Genetic Algorithm (RWGA) (Murata and Ishibuchi, 1995), Multiple Objective Genetic Algorithm (MOGA) (Foncesa and Fleming, 1993) , Nondominated Sorting Genetic Algorithm (NSGA) (Srinivas and Deb, 1994) and Niched-Pareto Genetic Algorithm (NPGA) (Horn et al., 1994) are some of the algorithms that are proposed in this class. These algorithms are easy to understand and implement. Even though they do not include an elite-preserving operator, some of these algorithms have good solutions in their original studies.

Elitist MOGAs: In this class, algorithms have elite-preserving operators, which differ from one algorithm to another. So, fittest solutions do not deteriorate in these algorithms. Rudolph's Elitist Multiobjective Evolutionary Algorithm (REMEA) (Rudolph, 2001), Elitist Nondominated Sorting Genetic Algorithm (NSGA-II) (Deb et al., 2000), Distance-Based Pareto Genetic Algorithm (DPGA) (Osyczka and Kundu, 1995), Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele, 1998) and Pareto-Archived Evolution Strategy (PAES) (Knowles and Corne, 2000) are some of the algorithms in this class. Presence of an elite-preserving operator makes a multiobjective genetic algorithm better converge to the pareto-optimal set.

We can refer the reader to Deb (2001) for a detailed description of these heuristics and their advantages and disadvantages.

In this study NSGA-II, one of the elitist MOGAs, is used to propose a solution to the biobjective MSPSP with hierarchical levels of skills. NSGA-II is proposed by Deb et al. (2000) as a MOGA. As Deb (2001) mentions, NSGA-II has fast convergence property and the spread of solutions is very good depending on its diversity metric. In NSGA-II, nondominated sorting is used in which the solution in the first nondominated front has the highest fitness score and the solutions in the same front have the same fitness score.

While forming fronts, each individual in the population is compared with every other individual. For each individual, number of solutions that dominate the individual and solutions that is dominated by this individual are found. Individuals that are not dominated by any member of the population (i.e., if number of solutions that dominate the individual is zero) are put in the first front. Then, individuals who are dominated by the individuals in the first front are checked and number of solutions that dominate these individuals is decreased by one. The ones that remain nondominated are put in the second front. This process is continued until all the individuals are put into fronts. At the end of the process, none of the individuals in a front dominate each other and a front that has a smaller index dominates all the fronts having larger indexes.

In order to maintain diversity, NSGA-II uses "crowded comparison". Crowding distance of an individual in a front is found by calculating the average distance between neighboring individuals. Given two individuals, the one with the lower rank is preferred. If two individuals have the same rank, then the one with the higher crowding distance, in other words the one located in a less crowded region is preferred. This ensures diversity in NSGA-II.

The steps in NSGA-II can be summarized as follows:

1. Creating initial population randomly
2. Sorting population into nondomination levels and assigning crowding distance
3. Generating child population by using binary tournament selection and problem dependent crossover and mutation operators
4. Combining parent and child population and sorting them based on nondomination
5. Where N is the population size, selecting best N solutions from 2N solutions according to solutions' ranks and crowding distances
6. Repeating steps 2, 3, 4 and 5 until the stopping criterion is satisfied.

Elitism is maintained in the 4^{th} step of the algorithm, where the parent and the child population are merged together. Therefore, selection is made from 2N individuals and the fittest individuals do not deteriorate.

CHAPTER 3

SOLUTION APPROACH

In this chapter the proposed algorithm for biobjective MSPSP with hierarchical levels of skills is stated. The performance metric in order to fine tune the algorithm is explained and the preliminary results are presented.

3.1 NSGA-II Employed

3.1.1 Solution Representation

As mentioned in Section 2.2, solution representation is very important in metaheuristics and Kolisch and Hartmann (1999) reported the best heuristics as metaheuristics which use activity list representations in RCPSP. For this reason, activity list representation is modified by adding the worker information for our problem. An example solution representation to a problem containing 30 non-dummy activities and 12 workers is given in Figure 2. First and the last activities are dummy activities. Activities on the top are in a precedence feasible order.

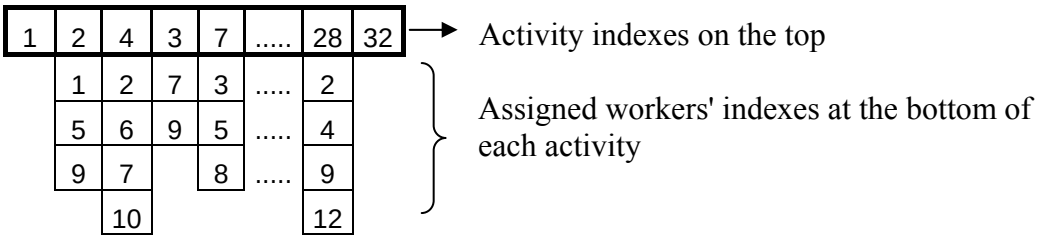

Figure 2 *An example solution representation*

The activities are scheduled in the order that is prescribed by the sequence. In order to calculate the first objective function value, which is the makespan, firstly dummy source activity is finished at time 0. Then for activity i, the following steps are applied:

1. Early start time ($ES(i)$) of activity i is found as the maximum of the finish times of its predecessors.
2. Early finish time $(EF(i))$ of activity i is calculated by adding duration of the activity to its early start time.
3. All the activities that are scheduled before activity i are checked (except its predecessors). If there is an intersection time between execution of any of the past activities and $[ES(i),EF(i)]$ times of activity i, it is checked whether there is a common worker or not. If there is a common worker, then $ES(i)$ is shifted to the finish time of the past activity that has a common worker. If there is no common worker or there is no intersection time $ES(i)$ is set as the start time of i.

At the end, finish time of the last activity gives us the makespan of the problem.

The second objective function value, which is total skill wasted, is simply calculated as in equation (8) in the mathematical model. For each level in each skill domain, skill requirement of an activity is subtracted from total skills of the assigned workers. Then, total skill wasted is calculated by summing wastes for all skill domains and skill levels for all of the activities.

3.1.2 Fitness Function

Based on the objective function values, nondominated sorting is done and fitness scores are assigned by using rank numbers. A solution in the first rank has the highest fitness score. For the solutions in the same rank, crowding distance is calculated and a solution with a larger crowding distance value has higher fitness score than a solution with a less crowding distance value.

3.1.3 Initial Population

Initial population is generated randomly in terms of both activities and workers. After the dummy source activity, activities are selected randomly from the decision set, which shows all the eligible activities (i.e. activities whose all predecessors are already scheduled), one by one until all the activities are selected.

As for the workers, for each activity worker sets are constructed by workers who have a skill that the activity needs. Then, for each activity, workers are selected randomly from the worker set until all the skill requirements of the activity are covered. After assignment, all workers are checked one by one whether he/she can be discarded or not. This is so important that after adding a worker to cover a skill, another worker may become redundant.

3.1.4 Selecting the Parents

Parents to be crossed over are selected based on tournament selection. In other words, from two individuals, the one with better fitness score is selected as the first parent and from another two individuals, again the better individual in terms of fitness score is selected as the second parent.

3.1.5 Generating New Members

One-point and two-point crossover operators are considered to generate child population. They are similar to the crossover technique described by Hartmann (1998) for RCPSP. In our case, assigned workers exist in the solution; however, it does not make a big difference, since the most important point is to keep feasibility in terms of activity sequences. Both of the operators take precedence relations into account. Therefore, the resulting offspring are also precedence feasible.

For one-point crossover operator, a number integer q is chosen randomly between 2 and the number of activities minus 2 (N-2) since the first and the last activities are dummy activities. Then, for the first offspring first q activities are taken from the

mother directly and the remaining activities (q+1, ….,N) are taken from the father. However, for the remaining activities, the activities that have already been taken from the mother are not considered again. By this way, relative positions in the parents' activity sequences are preserved and the precedence constraints are satisfied. We can refer the reader to Hartmann (1998) for the theorem that shows by this technique, precedence assumptions are fulfilled. The second offspring is formed similarly such that this time first q activities are taken from the father directly and the remaining activities are taken from the mother. For both of the offspring, while taking activities from parents, assigned workers to each activity are also carried over.

For the two-point crossover operator, the only difference is that this time there are two cutting points and two random numbers, $q_1 < q_2$, should be chosen. For the first offspring, first q_1 activities are taken from the mother, the positions q_1+1,…., q_2 are taken from the father and the remaining positions q_2+1,…,N are again taken from the mother. The second offspring is formed analogously, taking the first and the third part from the father and the second part from the mother. Again, while taking activities from parents, the workers assigned to each activity are also carried over.

3.1.6 Mutation

For our problem, mutation operator should both create activity sequences that could not have been produced by crossover operator and make new worker assignments that the parents do not have in order to let the algorithm search for the unvisited areas. For this purpose, several mutation operators are proposed for the algorithm and some of them are eliminated since they are not promising.

Firstly, an operator that finds the activity with the maximum wasted skill and changes the workers assigned to that activity is tried. Another alternative is selecting an activity randomly and changing its workers. However, these operators are eliminated in the preliminary tests, since changing the workers of only one activity is not sufficient when small mutation probabilities are used. For this reason, mutation operator is applied to all positions and both the activity sequence and the worker assignments for each activity are considered.

Two mutation operators are proposed in terms of worker assignments. For an activity, first operator selects a random worker and changes it with another random worker from the worker set. If the skill requirements of the activities are not satisfied, another worker is selected randomly and added. This process is repeated until all the skill requirements are satisfied. At the end, it is checked whether or not there is(are) worker(s) that can be discarded. This operator is also eliminated in preliminary tests since trying to preserve some of the workers is not so advantageous that it is decided to change all the assigned workers of the activity.

Therefore, for all activities with a mutation probability all the worker assignments are abolished and new workers are assigned randomly as in initializing the population in section 3.1.3. Again, possibility of discarding workers is checked and if it is possible, some of them are discarded. As for the activity sequence, with the same mutation probability each activity is tried to be replaced with the next activity if the precedence assumptions are not violated.

As a result, by the help of crossover and mutation operators, every solution to the problem is reachable.

3.1.7 Replacement

As mentioned before, as in all NSGA-II, parent and children populations are merged and individuals up to the population size are selected by taking firstly rank number and then crowding distance into account.

The pseudo code for the proposed NSGA-II is given in Appendix A.

3.2 Performance Metric Used

Before setting parameters for the proposed algorithm, first a performance metric should be defined in order to compare the determined parameters. As mentioned before, there are two desirable features for MOGAs, which are convergence to

pareto-optimal set and diversity. Therefore, performance metrics should be selected so that these two features can be measured.

Deb (2001, p.306-324) classifies performance metrics used for multiobjective optimization problems in the literature into three groups. The first type metrics measure only closeness to the pareto-optimal set. Error Ratio, Set Coverage Metric, Generational Distance and Maximum Pareto-Optimal Front Error are metrics in this class. Second type metrics measure only the diversity. Spacing, Spread and Chi-Square-Like Deviation Measure are commonly used metrics in this class. On the other hand, third type metrics evaluate two features in a combined sense. Hypervolume, Attainment Surface Based Statistical Method, Weighted Metric and Non-Dominated Evaluation Metric are metrics that evaluate both closeness and diversity. We can refer the reader to Deb (2001, p.306-324) for a detailed description of each performance metric.

Among these metrics, a metric from the third class is selected in order to evaluate both goals of the multiobjective optimization at the same time. Since Hypervolume Ratio (HVR), which is a variant of Hypervolume (HV), is easy to understand and calculate, it will be used to evaluate the performance of our algorithm during our study.

HV metric calculates the union of the volumes each nondominated point generates with respect to a reference point. The reference point can be found by using the worst objective function values. HV enclosed by a given set of nondominated points for a biobjective problem where both objectives are to be minimized is shown in Figure 3.

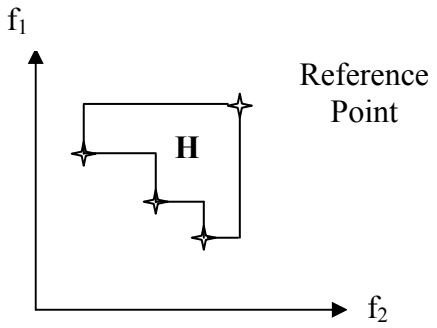

Figure 3 *HV enclosed by a given set of nondominated points*

As can be understood, an algorithm with a higher HV is preferred. An important point that should be taken into account is that if the objective function values are not in the same order of magnitude, this metric will favor solutions that converged in the objective function value which has higher order of magnitude. In order to eliminate this problem, HVR, which shows the relative distance between two set of nondominated solutions, can be used. HVR can be calculated as follows:

$$HVR = \frac{HV_{Approximation}}{HV_{Optimal}} \tag{14}$$

In our study, since there are scaling problems between makespan and total skills wasted in some of the test problems, HVR will be used while both fine tuning our algorithm and comparing the results of our algorithm with pareto-optimal set of solutions.

During fine tuning the algorithm since pareto-optimal set is not known, HVR is simply calculated by dividing H, which is represented in Figure 3, to A which is the total area bounded by the origin and the reference point as shown in Figure 4.

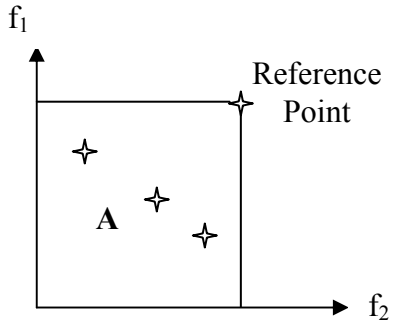

Figure 4 *Area bounded by origin and the reference point*

In our study, in order to find the worst objective function value in terms of makespan, all of the activity durations are summed. As for the worst objective function value in terms of total skills wasted, all of the workers that own at least one skill type that an activity needs are assigned to that activity and the total skills wasted is calculated by this way.

3.3 Parameter Setting

3.3.1 Population Size

Although large population sizes are desirable, the algorithm will require very long time in each generation when the population size is large. On the other hand, although population size determines the size of the pareto-optimal set in NSGA-II, since we will deal with the problems that contain 15, 30, 60 and 90 non-dummy activities in this study, there would not be so many nondominated points. So, in our problem, a very large population size is not needed to show all non-dominated points. Therefore, it is decided to use a population size of 100.

3.3.2 Crossover Probability

The original crossover probability of 0.8 proposed by the Deb et al. (2000) is used in the study.

3.3.3 Mutation Probability

Different mutation probabilities are proposed in the literature. Instead of selecting one of them, both 0.05 and 0.1 are used during fine tuning the algorithm.

3.3.4 Generation Number Limit

Different generation number limits should be used for different problem sizes since convergence to the pareto-optimal set becomes difficult and the probability of being stuck at local optima increases as the problem size increases. For this reason, after various experiments, generation number limit of 50 is used in the problems containing 15 non-dummy activities, limit of 100 for 30 non-dummy activities, limit of 150 for 60 non-dummy activities and 200 is used for the problems containing 90 non-dummy activities.

3.4 Tuning the Algorithm

While tuning the algorithm, the problem sets containing 15 and 30 non-dummy activities are used. For all problems, activity on node (AoN) representation is used. For the problems with 30 activities, precedence relationships and activity durations are taken directly from a project scheduling problem library, PSPLIB (Kolisch and Sprecher, 1996). Since the problem sizes start from 30 activities in the PSPLIB, for the problems with 15 activities, precedence relationships of 30 activity problems from PSPLIB are updated and activity durations are taken directly for the first 15 activities. Then, for each skill domain and level, skill requirements of the activities and skills that the workers have are added to the problems. While doing this, both complex activities, which require skills from different types and levels, and simple activities, which require skills from only one type, are added to each problem. Similarly, the workers that own complex skills and the workers with simple skills are both taken into account.

For each problem size, two kinds of problems are generated in terms of constraint tightness. Constraint tightness is changed by varying three parameters, which are

network complexity, number of skill types and skill ratio. These parameters are calculated as follows:

$$Network\ \ Complexity = \frac{total\ \ number\ \ of\ \ arcs}{total\ \ number\ \ of\ \ nodes} \tag{15}$$

While calculating network complexity, total number of nodes includes both non-dummy and dummy activities.

$$Number\ \ of\ \ skill\ \ types = number\ \ of\ \ skill\ \ domains * number\ \ of\ \ skill\ \ levels \tag{16}$$

$$Skill\ \ Ratio = \frac{\sum_{skill\ \ type} \frac{total\ \ skills\ \ available}{total\ \ skills\ \ needed}}{number\ \ of\ \ skill\ \ types} \tag{17}$$

While calculating the skill ratio of a problem, for each skill level in a skill domain, total skills available is divided by total skills needed by all activities.

Parameters that are used in each test problem are given in Table 1. As can be seen, while Problem-1 and Problem-3 are relaxed in terms of constraint tightness, Problem-2 and Problem-4 are tight.

Table 1 *Parameters for test problems*

	P1	**P2**	**P3**	**P4**
# of Activities	17	17	32	32
# of Workers	6	6	12	12
Network Complexity	1.2	1.5	1.5	2.1
# of Skill Types	4	9	4	9
Skill Ratio	0.5	0.3	0.5	0.3

3 different problems are generated for each type, so experiments are done for 12 problems. PSPLIB file numbers for these problems are given in Appendix B. 10 replications are performed for each problem and parameter setting. Therefore, for 2 kinds of crossover operator and 2 kinds of mutation probability total of 480 runs are made. The algorithm is coded by Matlab 7.0 and all the computational analysis are done on an Intel Core Duo, 1.86 GHz computer with 2.90 MB memory.

For each problem type, average HVR and average elapsed time in seconds are calculated for 30 runs (10 replications for 3 problems) and given in Table 2. CO stands for crossover operator and MP stands for mutation probability. As can be seen, for each problem type, the results do not differ much from one combination to another.

Table 2 *Full factorial design*

Parameter Combination		**Problem Category**							
		17 Activities, 6 Workers				**32 Activities, 12 Workers**			
		Relaxed (P1)		**Tight (P2)**		**Relaxed (P3)**		**Tight (P4)**	
CO	**MP**	**HVR (Avr.)**	**Time (Sec.)**	**HVR (Avr.)**	**Time (Sec.)**	**HVR (Avr.)**	**Time (Sec.)**	**HVR (Avr.)**	**Time (Sec.)**
One point	**0.05**	0.6284	65.53	0.3735	71.40	0.7234	445.56	0.6553	488.74
	0.1	0.6287	65.65	0.3810	71.21	0.7231	445.65	0.6561	489.42
Two point	**0.05**	0.6283	66.28	0.3766	70.91	0.7235	446.18	0.6564	487.02
	0.1	0.6287	66.46	0.3844	71.21	0.7231	447.19	0.6558	488.39

Two-way interactions between the factors (i.e. problem type, crossover type and mutation probability) are analyzed using Minitab. According to analysis of variance table, it is found that interactions between factors are not significant. This can be understood visually from the interaction plots for HVR and time, which are given in Figure 5 and Figure 6 respectively.

From Figure 5, it can be seen that HVR is the highest for P3 and lowest for P2. HVR for P3 is higher than P4 since P3 is relaxed and P4 is tight. Similarly, HVR for P1 is higher than P2. P3 and P4 are on the top since as the number of workers increases, HVR increases. P3 and P4 contain 12 workers whereas P1 and P2 contain 6 workers and having higher number of workers results in solutions that assign appropriate workers to the activities so solutions that have low skill waste and high HVR values.

As for the interactions, from the first row it can be seen that HVR values for each problem type form a straight line for two types of crossover operator and for two types of mutation probability. First plot shows that there is no interaction between problem types and crossover operator. Similarly, second plot shows that there is no interaction between problem types and mutation probability. Moreover, when we look at the second row, it is obvious that there is also no interaction between crossover operator and mutation probability since the black and red lines are coinciding. However when carefully looked, when the mutation probability is 0.05, it can be seen that two-point crossover gives better HVR values as the red line is slightly above the black one on the left part of the graph.

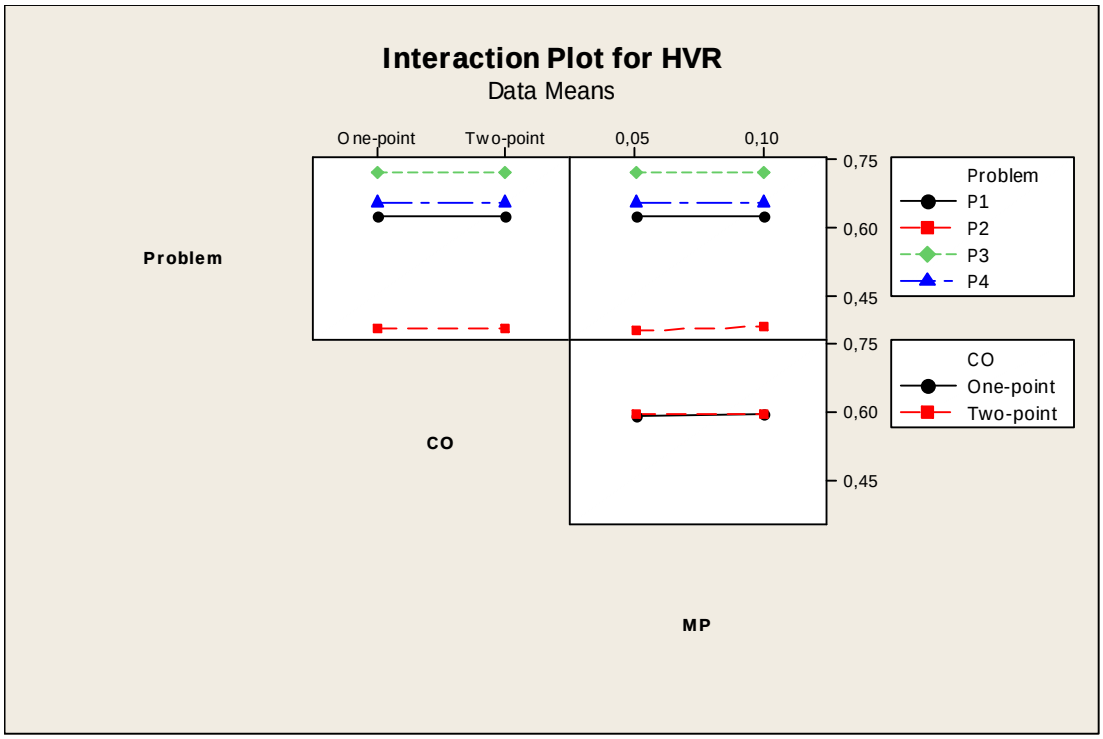

Figure 5 *Interaction plot for HVR*

As can be seen from Figure 6, as expected elapsed time is the highest for the problem that contains 32 activities with tight constraints. It is the lowest for two small-sized problems, P1 and P2. Time needed for P2 is higher than P1 in a very small amount since P2 is tighter than P1. Again, there is no interaction between problem types and crossover operator or mutation probability in terms of elapsed time. Also there is no interaction between crossover operator and mutation probability. For a problem type, time requirements are the same for all combinations of crossover operator and mutation probability.

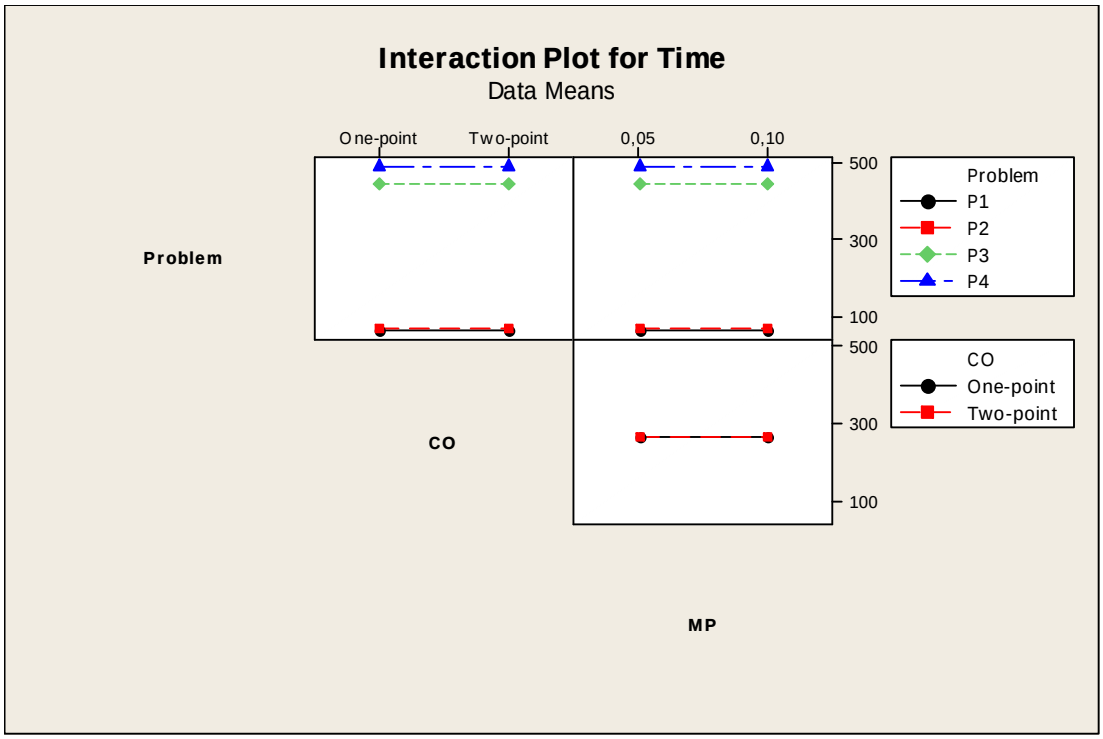

Figure 6 *Interaction plot for Time*

As a result, although HVR values and elapsed time do not differ much from one combination to another, firstly combination with one-point crossover operator and 0.05 mutation probability is eliminated since when the mutation probability is 0.05, two-point crossover operator gives better HVR values. Then, the combination including two-point crossover operator and mutation probability of 0.1 is selected since when the average HVR values for 4 problems is calculated, result is slightly higher with this combination although this small difference cannot be seen from the interaction plots.

3.4 Convergence of the HVR Values

After selecting the appropriate parameter settings for the proposed algorithm, convergence of the HVR values are analyzed. An example graph for the HVR values versus generation number for each problem type is plotted and given in Figures 7, 8, 9 and 10 for P1, P2, P3 and P4, respectively.

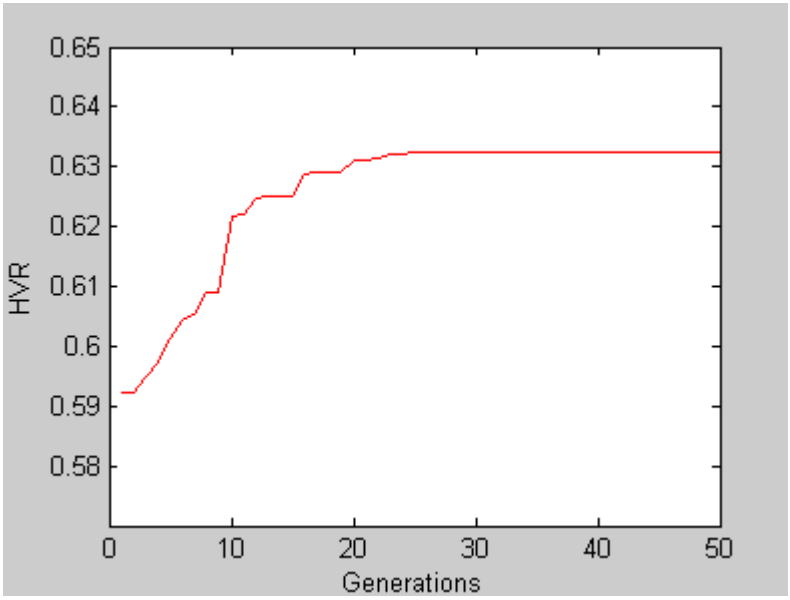

Figure 7 *Convergence graph of HVR for P1*

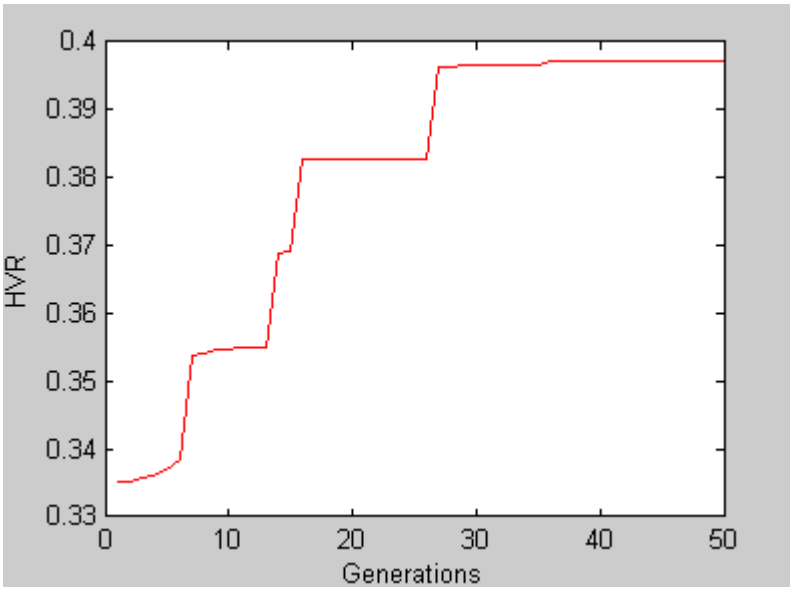

Figure 8 *Convergence graph of HVR for P2*

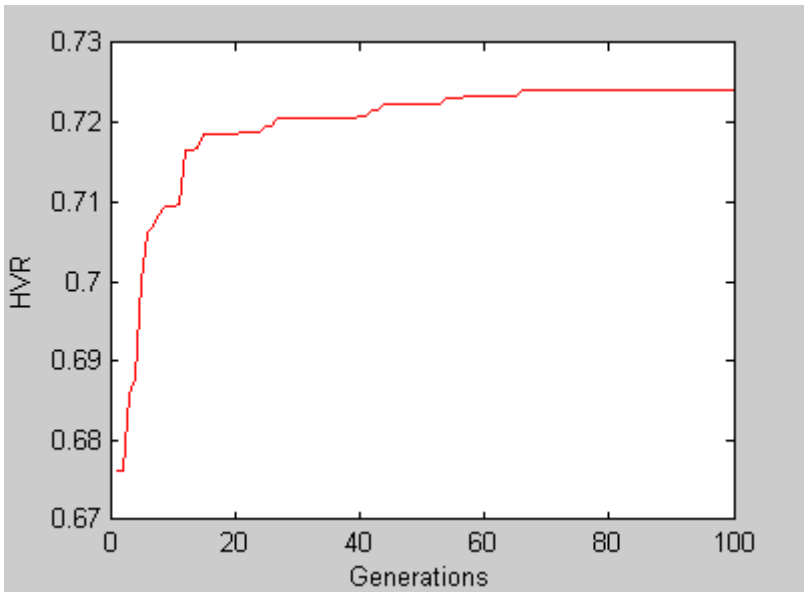

Figure 9 *Convergence graph of HVR for P3*

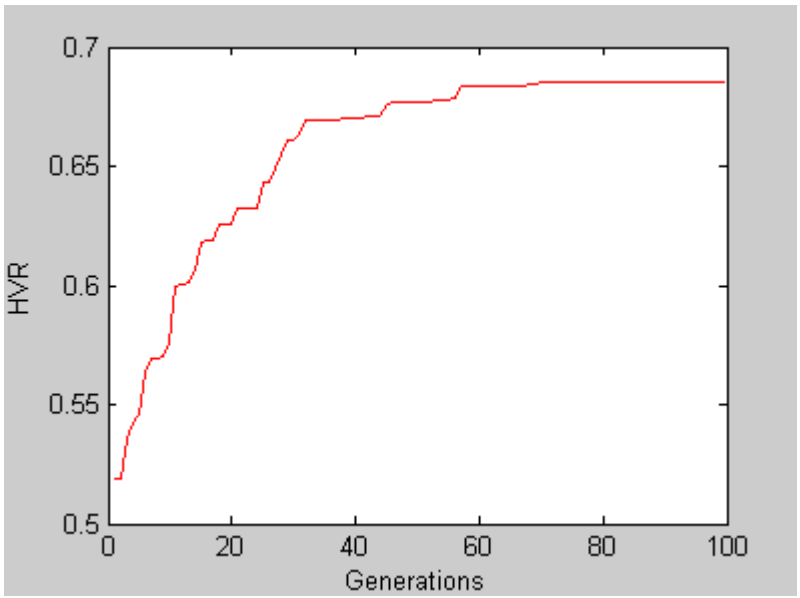

Figure 10 *Convergence graph of HVR for P4*

As can be seen, since NSGA-II is an elitist algorithm, HVR values are non-decreasing. From the graphs, it is understood that generation number limit of 50 for the problems with 17 activities and generation number limit of 100 for the problems with 32 activities are enough since HVR increases at the beginning of the algorithm and it forms a straight line towards the generation limit.

3.4 Robustness of the Proposed Algorithm

Proposed heuristic for biobjective MSPSP with hierarchical levels of skills gives a solution based on the initial population which is generated randomly. Therefore, in order to measure its performance 10 replications are conducted. If the result of these 10 replications proves that the correlation between initial condition and the result is weak, then the algorithm is said to be robust and it is adequate to use the results of a single run.

In order to understand the robustness of the proposed algorithm, results of 10 replications for the all problems are analyzed. In addition to minimum, average and maximum values, standard deviations of HVR and elapsed time are calculated and given in Table 3. As mentioned before, 3 different problems are generated for each problem type and these are shown by adding the problem number at the end of P1, P2, P3 and P4. Then, it is checked whether the standard deviations of the performance measures are high when compared to the average values for each problem.

Table 3 *Experiment on robustness of the proposed algorithm*

		Min	Avr	Max	Stdev	1% of Avr
P1_1	**HVR**	0,6319	0,6322	0,6324	0,0002	0,0063
	Time (Sec.)	65,41	65,95	66,97	0,51	0,66
P1_2	**HVR**	0,6629	0,6630	0,6630	0,0000	0,0066
	Time (Sec.)	64,97	65,41	66,08	0,39	0,65
P1_3	**HVR**	0,5904	0,5910	0,5912	0,0002	0,0059
	Time (Sec.)	67,17	68,00	68,66	0,40	0,68
P2_1	**HVR**	0,2901	0,3324	0,3590	**0,0213**	0,0033
	Time (Sec.)	70,55	71,43	72,27	0,47	0,71
P2_2	**HVR**	0,3829	0,4027	0,4250	**0,0118**	0,0040
	Time (Sec.)	70,97	71,17	71,49	0,18	0,71
P2_3	**HVR**	0,4084	0,4180	0,4190	0,0034	0,0042
	Time (Sec.)	70,28	71,02	71,99	0,47	0,71
P3_1	**HVR**	0,7545	0,7561	0,7582	0,0011	0,0076
	Time (Sec.)	441,34	443,90	448,52	2,16	4,44
P3_2	**HVR**	0,7241	0,7246	0,7251	0,0004	0,0072
	Time (Sec.)	446,05	451,95	457,30	2,88	4,52
P3_3	**HVR**	0,6883	0,6886	0,6890	0,0003	0,0069
	Time (Sec.)	442,50	445,71	448,81	1,90	4,46
P4_1	**HVR**	0,6437	0,6511	0,6540	0,0036	0,0065
	Time (Sec.)	484,05	488,27	492,34	2,91	4,88
P4_2	**HVR**	0,6267	0,6286	0,6300	0,0008	0,0063
	Time (Sec.)	491,45	492,64	494,39	1,03	4,93
P4_3	**HVR**	0,6745	0,6879	0,7021	**0,0093**	0,0069
	Time (Sec.)	482,47	484,26	486,38	1,12	4,84

As can be seen from the table, standard deviation of the results are always less than 1% of the average results in terms of both HVR and elapsed time except from three cases which are written in bold. Among these three cases, the maximum deviation is nearly 7% of the average HVR in P2_1. This means that pareto-optimal sets found are very close to each other and the time needed to find them does not change so much from one simulation to another. For this reason, results of a single run will be used while testing the performance of the proposed algorithm.

CHAPTER 4

COMPUTATIONAL RESULTS

In this chapter, in order to measure the performance of the proposed algorithm, its results are compared with the approximation of the pareto-optimal set for small-sized problems and with random search for large-sized problems.

4.1 Selection of Test Problems

While fine tuning the algorithm, the problem sets containing 15 and 30 non-dummy activities were used and for each problem size, two kinds of problems were generated in terms of constraint tightness. Parameters used for each type were given in Table 1. For these 4 problem types, 3 different problems were generated, so experiments were done on 12 problems. In this chapter in addition to these 12 problems, 8 different problems are generated. Therefore, proposed algorithm is evaluated based on total of 20 problems.

For new problems, problem sets containing 60 and 90 non-dummy activities were used. Again, precedence relationships and activity durations are taken directly from PSPLIB. Then, for each skill domain and level, skill requirements of the activities and skills that the workers have are added to the problems. As mentioned before, while constructing the activity requirements and worker skills, both complex and simple activities and the workers that own complex skills and simple skills are taken into account.

For each problem size, again two kinds of problems are generated in terms of constraint tightness which depends on network complexity, number of skill types and skill ratio. Parameters used for new test problems are given in Table 4.

As can be seen while P5 and P7 are relaxed in terms of constraint tightness, P6 and P8 are tight. For each problem type, 2 different problems are generated. Therefore, total of 8 problems are obtained. PSPLIB file number for each problem is given in Appendix B.

Table 4 *Parameters used for new test problems*

	P5	**P6**	**P7**	**P8**
# of Activities	62	62	92	92
# of Workers	24	24	36	36
Network Complexity	1.5	2.1	1.5	2.1
# of Skill Types	4	9	4	9
Skill Ratio	0.5	0.3	0.5	0.3

4.2 Comparison with the Approximation of the Pareto-optimal Set

Since the problem stated here has not been studied before, benchmark results are not available in the literature. Thus, the approximation of the pareto-optimal set is found by using a variation of ε-constraint method.

ε-constraint method is based on minimizing one objective function and considering the other objectives as constraints bound by some allowable levels ε_i. By this way, the model becomes a single objective model. In our problem, the second objective function value, which is total skill wasted, can be written as the original objective function and the first objective function value, which is the makespan, can be written as a constraint. Since the maximum makespan of the problems can be taken as the sum of the activity durations, left hand side of the makespan can be decreased starting from the maximum value in order to find different solutions to the problem.

However, by this way the model can give weakly nondominated but dominated solutions. In other words, for the same total waste value, the model can give a solution with a higher makespan which satisfies the makespan constraint while there exists a solution with a lower makespan. In order to avoid weakly nondominated but dominated solutions, a variation of ε-constraint method is used and makespan is multiplied by a small value and added to the objective function. By this way, since the makespan is added as an objective, the model will give the solutions with the highest makespan possible and this will result in nondominated solutions.

Therefore, the mathematical model of the problem is updated by changing the objective function as in (18) and adding the constraint (19) while keeping all the other constraints the same. Since the resulting values of the two objectives differ at most in one order of magnitude for all of the test problems, using 0.001 as a coefficient of the makespan is sufficient to find all nondominated solutions.

$$\text{Minimize } \sum_i \sum_\alpha \sum_\beta e_{\alpha\beta}^i + 0.001 * \sum_k k * z_{nk} \tag{18}$$

$$\sum_k k * z_{nk} \leq \varepsilon_i \tag{19}$$

This model is solved for 6 problems including 17 activities by using the mathematical programming software GAMS. For all problems, maximum makespan is taken as the starting ε_i value and when a solution obtained, for the next solution resulting makespan is decreased by 1 and added as a constraint. This method is repeated until an infeasible solution is obtained. By this way, solutions given in Appendix C are obtained for P1_1, P1_2, P1_3, P2_1, P2_2 and P2_3. Among all solutions, nondominated ones are written in bold.

As can be seen, while the relaxed problem (P1) can be solved in maximum one hour, tight problem (P2) needs days to find all nondominated solutions. For this reason, only problems including 17 activities are evaluated by this method and other

problems, which include more activities, are evaluated by comparing them with a random search.

Minimum and maximum hypervolumes (HV) for the proposed algorithm found during fine tuning (among 10 runs) are compared to the hypervolumes found by GAMS. The solutions that have minimum and maximum hypervolume for each problem are taken and hypervolume, elapsed time, makespan, and waste values of the proposed NSGA-II are reported in Appendix D.

The resulting values for hypervolume (HV) and hypervolume ratio (HVR) are summarized in Table 5.

Table 5 *Comparison with the approximation of the pareto-optimal set*

	NSGA-II		**GAMS**		
	Min HV	**Max HV**	**HV**	**Min HVR**	**Max HVR**
P1_1	8,651	8,657	8,657	1.00	1.00
P1_2	11,295	11,297	11,297	1.00	1.00
P1_3	8,833	8,844	8,845	1.00	1.00
P2_1	7,937	9,823	9,824	0.81	1.00
P2_2	5,896	6,545	6,536	0.90	1.00
P2_3	8,436	8,655	8,655	0.97	1.00

As can be seen, all maximum HVR values are approximately 1. As for the minimum results, while the problems with loose constraints have HVR values of approximately 1, tight problems can have less HVR values. Unsurprisingly, the maximum deviation occurs for P2_1 and P2_2 which are the two problems that were reported as the most deviated ones in Table 3 out of 12 problems. For the other test problems, except from P4_3, deviation was always less than 1% of the average HVR, which implies if it was possible to compare the results of the other problems with GAMS results, there would be slight difference between minimum and maximum HVR values.

To sum up, HVR values are close to 1 for minimum hypervolume results and always approximately 1 for maximum hypervolume results, which prove a very good approximation of the proposed algorithm. Moreover, when compared to GAMS results, the proposed algorithm requires very short time, which is at most 72 seconds for these 6 problems.

In addition to giving the resulting values of nondominated solutions of both algorithms, pareto fronts of them are compared visually for both minimum and maximum HVR values in Appendix E.

For the tight problems (P2), some solutions found by the proposed algorithm dominate the solutions found by GAMS. This is because GAMS has a relative optimality tolerance of 0.1 by default. This means that, the solver stops when it finds a feasible solution within 10% of the global optimum. For this reason, when the problem is tight, computational time increases and this may result in solutions that are not global optimum.

4.3 Comparison with the Random Search

Since it will take very long durations to find the solutions to the problems that contain 32, 62 and 92 activities by GAMS, in order to evaluate the proposed algorithm for these problems, the results are compared with the random search. Before doing that, in order to understand the performance of the random search, small-sized problems are also solved by random search and their results are compared with the GAMS results.

In the random search, solutions are generated randomly as in the initial population step of the proposed NSGA-II. Activities are selected randomly from the eligible activities one by one in order to satisfy precedence relationships and workers are selected randomly from the worker set until all the skill requirements of the activity are covered. After assignment, all workers are checked one by one whether he/she can be discarded or not.

After each problem is solved by the proposed algorithm, the random search is performed until elapsed time is equal to the time needed to perform the proposed algorithm. At the end of the random search, nondominated sorting is performed and the solutions that are nondominated are reported.

Hypervolumes found by random search are compared to the hypervolumes found by GAMS for the problems containing 17 activities and the ratios are given in Table 6.

Table 6 *Comparison of random search with GAMS*

	RANDOM SEARCH	**GAMS**	
	HV	**HV**	**HVR**
P1_1	7,911	8,657	0.91
P1_2	10,406	11,297	0.92
P1_3	8,107	8,845	0.92
P2_1	5,359	9,824	0.55
P2_2	4,343	6,536	0.66
P2_3	8,239	8,655	0.95

As can be seen, although HVR values are small for the tight problems (except from P2_3), they are close to 1 for the relaxed problems and since there are no benchmark results available in the literature, proposed algorithm is compared with random search for large-sized problems .

For each problem, nondominated solutions found by the proposed algorithm and the random search are compared in Appendix F, by giving time, HVR, makespan and total skill wasted values. Since pareto-optimal set is not known, HVR is again calculated by dividing the area generated by the algorithm to the total area bounded by the origin and the reference point. HVR results for each problem are summarized in Table 7.

Percentage1 shows the difference between HVR values divided by the HVR value found by the proposed algorithm and Percentage2 shows the difference between HVR values divided by the HVR values found by the random search.

Table 7 *Comparison with the random search*

	NSGA-II HVR	**Random HVR**	**Percentage1**	**Percentage2**
P3_1	0.7565	0.6751	10.76%	12.06%
P3_2	0.7249	0.6880	5.09%	5.36%
P3_3	0.6890	0.6508	5.54%	5.87%
P4_1	0.6531	0.4664	**28.59%**	**40.05%**
P4_2	0.6288	0.5142	**18.23%**	**22.29%**
P4_3	0.6799	0.4906	**27.84%**	**38.58%**
P5_1	0.7554	0.7266	3.80%	3.96%
P5_2	0.8023	0.7608	5.17%	5.45%
P6_1	0.7256	0.6196	**14.60%**	**17.10%**
P6_2	0.6966	0.5790	**16.88%**	**20.31%**
P7_1	0.8655	0.8289	4.23%	4.42%
P7_2	0.8064	0.7755	3.82%	3.98%
P8_1	0.7942	0.6870	**13.49%**	**15.59%**
P8_2	0.7561	0.6394	**15.43%**	**18.24%**

As can be seen from the table, proposed algorithm is always better than the random search. Moreover, the difference between two algorithms becomes significant when the problems are tight as can be understood from the percentages written in bold. When the resulting makespan and waste values are analyzed from Appendix F, it can be understood that especially total skills wasted are very high when compared to the values found by NSGA-II. This shows that proposed algorithm is good at satisfying person-job fit which ensures prevention of overqualification.

CHAPTER 5

CONCLUSION

In the literature, the project scheduling problem has attracted many researchers and there are many different kinds of studies that are documented in this area. In this study, an extension of Multi-Mode Resource-Constrained Project Scheduling Problem which is called Multi-Skill Project Scheduling Problem (MSPSP) with hierarchical levels of skills is focused on.

Although most of the project scheduling problems deals with the makespan, cost, risk or other activity based objective, in this study in addition to the makespan objective, a motivation based objective is taken into account since human factor is very important in order to complete a project successfully. Therefore, while trying to minimize the makespan of the project, at the same time minimizing total wasted skills is aimed. By this way, it is tried to minimize the assignment of overqualified staff to the activities and satisfy person-job fit in order to prevent job dissatisfaction, which affects staff motivation directly.

Biobjective MSPSP with hierarchical levels of skills has not been studied in the literature before. In order to solve the problem, a well-known Multiobjective Genetic Algorithm, NSGA-II is used. Four different parameter combinations are proposed for the heuristic and one of them is selected after analyzing full factorial design of the results in terms of both hypervolume ratio and elapsed time. Test problems are generated for the problems including 17, 32, 62 and 92 activities.

Results of the proposed algorithm are compared with the GAMS results found by using a variation of ε-constraint method for the problems including 17 activities. Since it takes a very long time to find all nondominated solutions by using GAMS, for larger problem sizes, results of the proposed algorithm are compared with the random search.

While comparing with GAMS, HVR values are near to 1 for minimum hypervolume results and always approximately 1 for maximum hypervolume results, which prove a very good approximation of the proposed algorithm. Moreover, the proposed algorithm requires very short time compared to GAMS.

As for the larger problem sizes, the proposed algorithm is always better than the random search. Its performance is much better when the problem is tighter. When the results of the two algorithms are analyzed, it is seen that the proposed NSGA-II is very good at minimizing skill waste, which directly affects staff motivation.

In conclusion, proposed NSGA-II is shown to be an effective and robust heuristic for the biobjective MSPSP with hierarchical levels of skills. As future search directions, performance of the heuristic can be observed for larger problem sizes and/or for different problem types such as containing scarce workers at specific skill types. Moreover, large-sized problems can be compared with other heuristics in addition to the random search and the ways to decrease the computational time can be analyzed. As for the problem environment, it can be assumed that the activity durations change depending on the skill types and levels of workers. Furthermore, how motivation of the worker changes depending on the other workers assigned to the same activity can be analyzed since this can also affect the activity durations.

REFERENCES

Abbasi B., Shadrokh S. and Arkat J., 2006. Bi-Objective Resource-Constrained Project Scheduling with Robustness and Makespan Criteria. *Applied Mathematics and Computation 180: 146–152.*

Alcaraz J. and Maroto C., 2001. A Robust Genetic Algorithm for Resource Allocation in Project Scheduling. *Annals of Operations Research 102: 83–109.*

Bellenguez O. and Neron E., 2005. Lower Bounds for the Multi-Skill Project Scheduling Problem with Hierarchical Levels of Skills. *Lecture Notes in Computer Science, 3616: 229-243.*

Bellenguez-Morineau O., 2008. Methods to Solve Multi-Skill Project Scheduling Problem. *4OR*, 6: 85-88.

Blazewicz J., Lenstra J.K. and Rinnooy Kan A.H.G, 1983. Scheduling Projects to Resource Constraints: Classification and Complexity. *Discrete Applied Mathematics, 5: 11-24.*

Cordeau J.F., Laporte G., Pasin F. and Ropke S., 2009. Scheduling Technicians and Tasks in a Telecommunications Company. *Journal of Scheduling.*

Deb K., Agrawal S., Pratap S. and Meyarivan T., 2000. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. Kanpur Genetic Algorithms Laboratory (KanGal), Indian Institute of Technology, Kanpur, INDIA.

Deb K., 2001. Multiobjective Optimization Using Evolutionary Algorithms. *John Wiley & Sons Ltd.*

Foncesa C. M. and Fleming P. J., 1993. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms: 416-423.*

French Operational Society, http://www.g-scop.inpg.fr/ChallengeROADEF2007, Last Accessed Date: 03 January 2010.

France 24, http://www.france24.com/en/20090928-suicide-france-telecom-unions-stress-restructuring-france, Last Accessed Date: 03 January 2010.

Goldberg D. E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. *MA: Addison-Wesley.*

Hajela P., Lee E. and Lin C. Y., 1993. Genetic Algorithms in Structural Topology Optimization. *Proceedings of the NATO Advanced Research Workshop on Topology Design of Structures: 117-133.*

Haouari M. and Al-Fawzan M., 2002. A Bi-objective Model for Maximizing the Quality in Project Scheduling. *DIMACS Technical Report, 2002-14.*

Hapke M., Jaszkiewicz A. and Slowinski R, 1998. Interactive Analysis of Multiple-Criteria Project Scheduling Problems. *European Journal of Operational Research 107: 315-324.*

Hartmann S., 1998. A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling. *Naval Research Logistics 45: 733–750.*

Hartmann S. and Kolisch R., 2000. Experimental Evaluation of State-of-the-Art Heuristics for the Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research 127: 394–407.*

Hartmann S. and Kolisch R., 2006. Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update. *European Journal of Operational Research 174: 23-37.*

Hartmann S. and Briskorn D., 2008. A Survey of Deterministic Modeling Approaches for Project Scheduling under Resource Constraints. *Hamburg School of Business Administration Working Paper 2 (2008).*

Hartmann S. and Briskorn D., 2009. A Survey of Variants and Extensions of the Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research, In Press, 2009.*

Horn J., Nafploitis N. and Goldberg D., 1994. A Niched Pareto Genetic Algorithm for Multi-Objective Optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation: 82-87.*

Hoskins T. E., 2003. The Effects of Perceived Overqualification On Job Satisfaction, Organizational Commitment, and Turnover: A Study of AFIT Graduates, Unpublished M.Sc. Thesis. Department of the Air Force, Air University, Ohio, USA.

Hurkens C.A.J., 2007. Incorporating the Strength of MIP Modeling in Schedule Construction. In ROADEF 2007, the 8th Congres de la Societe Francaise de Recherche Operationnelle et d'Aide a la Decision, Grenoble, France.

Icmeli-Tukel O. and Rom W.O., 1997. Ensuring Quality in Resource Constrained Project Scheduling. *European Journal of Operational Research, 103: 483-496.*

Johnson W.R., Morrow P.C. and Johnson G.J., 2002. An Evaluation of a Perceived Overqualification Scale Across Work Settings. *The Journal of Psychology, 136(4): 425-441.*

Knowles J. D. and Corne D. W., 2000. Approximating the Non-Dominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation Journal 8(2): 149-172.*

Kolisch R., 1996. Serial and Parallel Resource-Constrained Project Scheduling Methods Revisited: Theory and Computation. *European Journal of Operational Research, 90: 320–333.*

Kolisch R. and Sprecher A., 1996. PSPLIB – A Project Scheduling Problem Library. *European Journal of Operational Research 96: 205-216.*

Kursawe F., 1990. A Variant of Evolution Strategies for Vector Optimization. *Parallel Problem Solving from Nature I (PPSN-I): 193-197.*

Li H. and Womer K., 2008. Modeling the Supply Chain Configuration Problem with Resource Constraints. *International Journal of Project Management, 26: 646-654.*

Murata T. and Ishibuchi H., 1995. MOGA: Multi-objective Genetic Algorithms. *Proceedings of the Second IEEE International Conference on Evolutionary Computation: 289-294*

Oktay M., 2000. Fuzzy Project Scheduling with Motivation Considerations, Unpublished M.Sc. Thesis. Department of Industrial Engineering, Middle East Technical University, Ankara, Turkey.

Osyczka A. and Kundu S., 1995. A New Method to Solve Generalized Multicriteria Optimization Problems Using the Simple Genetic Algorithm. *Structural Optimization 10(2): 94-99.*

Rudolph G., 2001. Evolutionary Search under Partially Ordered Fitness Sets. *Proceedings of the International Symposium on Information Science Innovations in Engineering of Natural and Artificial Intelligent Systems (ISI 2001): 818-822.*

Schaffer J. D., 1985. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. *Proceedings of the First International Conference on Genetic Algorithms: 93-100.*

Srinivas N. and Deb K., 1994. Multi-objective Function Optimization Using Non-Dominated Sorting Genetic Algorithms. *Evolutionary Computation Journal 2(3): 221-248.*

Tareghian H.R. and Taheri S.H., 2007. A Solution Procedure for the Discrete Time, Cost and Quality Tradeoff Problem Using Electromagnetic Scatter Search. *Applied Mathematics and Computation, 190: 1136-1145.*

The Australian, http://www.theaustralian.com.au/news/breaking-news/another-death-in-france-telecom-suicide-spiral/story-fn3dxity-1225780658273, Last Accessed Date: 03 January 2010.

Toklu Y.C., 2002. Application of Genetic Algorithms to Construction Scheduling with or without Resource Constraints. *Canadian Journal of Civil Engineering 29: 421–429.*

Valls V., Perez A. and Quintanilla S., 2009. Skilled Workforce Scheduling in Service Centers. *European Journal of Operational Research, 193: 791-804.*

Vanhoucke M., 2006. Scheduling an R&D Project with Quality-Dependent Time Slots. *Lecture Notes in Computer Science, 3982: 621-630.*

Viana A. and J. P. de Sousa, 2000. Using Metaheuristics in Multiobjective Resource Constrained Project Scheduling. *European Journal of Operational Research 120: 359-374.*

Vieira J.A.C., (2005). Skill mismatches and job satisfaction. *Economics Letters, 89: 39–47.*

Zitzler E. and Thiele L., 1998. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Zürich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH).

APPENDIX A

PSEUDO CODE FOR THE PROPOSED NSGA-II

Define

- P(i): Parent population in i^{th} generation (size = pop)
- C(i): Children population in i^{th} generation (size = pop)
- M(i): Mixed population in i^{th} generation (size = 2 * pop)

Initialize

- Generate P(1) by
    - Selecting activities randomly among the eligible ones
    - Assigning workers randomly among the ones that owns a skill that the activity needs until all the skill requirements of the activity is satisfied
        - After worker assignment, all workers are checked whether they can be discarded or not. If possible, discard the worker(s)
- Evaluate P(1)
    - Calculate makespan
        - Find early start and early finish of activity(j) by
            - ES(j) = max{finish time of predecessors of activity(j)}
            - EF(j) = ES(j) + duration(j)
        - For all the activities scheduled before activity(j) (except from its predecessors), check whether ES(j) and EF(j) are between past_activity(k)'s start and finish times
            - If S(k)<=ES(j)<F(k) or S(k)<EF(j)<=F(k), check whether there is a common worker
                - If there is a common worker, set S(j)= F(k)

        - If there is no common worker for all the past_activity(k), set S(j)= ES(j)
      - If EF(j)<=S(k) or ES(j)>=F(k) or for all the past_activity(k), set S(j)= ES(j)
    - Calculate total wasted skills
- Assign rank and crowding distance
  - Assign rank
  - Assign crowding distance
    - Assign infinity to extreme solutions
    - For other solutions, crowding distance is equal to summation of the following equation for makespan and total wasted skills:

      $$\frac{\text{difference of two neighboring solutions in objective i}}{\text{difference of the extreme solutions in objective i}}$$
- Return P(1) as P(2)

For (i=2 ; i<=generation number ; i++)

- **Crossover**
  - Randomize P(i) as POPULATION1
  - Randomize P(i) as POPULATION2
  - For (j=1; j<population_size; j=j+4)
    - Choose individual(j) and individual(j+1) from POPULATION1
      - Tournament select parent1 among individual(j) and individual(j+1)
        - If individual(j) dominates individual(j+1), select individual(j)
        - If individual(j+1) dominates individual(j), select individual(j+1)
        - If they are nondominated

        - If crowding_distance(j)> crowding_distance(j+1), select individual(j)
        - If crowding_distance(j)< crowding_distance(j+1), select individual(j+1)
        - If crowding_distance(j)= crowding_distance(j+1), select one of the individuals randomly
- Choose individual(j+2) and individual(j+3) from POPULATION1
    - Tournament select parent2 among individual(j+2) and individual(j+3)
- Choose individual(j) and individual(j+1) from POPULATION2
    - Tournament select parent3 among individual(j) and individual(j+1)
- Choose individual(j+2) and individual(j+3) from POPULATION2
    - Tournament select parent4 among individual(j+2) and individual(j+3)
- Perform crossover
    - For parent1 and parent2
        - With crossover probability perform one-point or two-point crossover
        - Otherwise keep parent1 and parent2 as children
- Perform crossover
    - For parent3 and parent4
        - With crossover probability perform one-point or two-point crossover
        - Otherwise keep parent3 and parent4 as children

- Return child population C(i)

- **Mutation**
  - For all activities of a child perform mutation with mutation probability
    - If the precedence relationships allow, replace the activity with the activity in the next position
    - Change all the workers assigned to the activity by selecting workers randomly among the ones that owns a skill that the activity needs until all the skill requirements of the activity is satisfied
      - After worker assignment, all workers are checked whether they can be discarded or not. If possible, discard the worker(s)
  - Return mutated C(i)

- **Evaluate child population C(i)**

- **Merge P(i) and C(i) into mixed population M(i)**

- **Perform nondominated sorting**
  - Divide M(i) into fronts until having a population of size pop and assign crowding distance to all fronts
    - If the last front's size is larger than the (pop-previous fronts)
      - Fill population with all individuals from the previous fronts and take the individuals from the last front in nonincreasing order of crowding distance
    - Otherwise fill the population with all the individuals
  - Return P(i+1)

APPENDIX B

FILE NUMBERS FROM PSPLIB

Table 8 *File numbers from PSPLIB*

Problem Number	**File Number**
P1_1	j302_1
P1_2	j302_2
P1_3	j302_3
P2_1	j3048_1
P2_2	j3048_2
P2_3	j3048_3
P3_1	j301_1
P3_2	j301_2
P3_3	j301_3
P4_1	j3033_1
P4_2	j3033_2
P4_3	j3033_3
P5_1	j601_1
P5_2	j601_2
P6_1	j6033_1
P6_2	j6033_2
P7_1	j901_1
P7_2	j901_2
P8_1	j9033_1
P8_2	j9033_2

APPENDIX C

GAMS RESULTS FOR P1 AND P2

Table 9 *GAMS results for P1_1*

Makespan Constraint <=	Makespan	Total Waste	Time (Hrs)
74	73	8	00:00:36
72	**32**	**8**	00:01:59
31	**30**	**9**	00:00:49
29	29	10	00:01:00
28	**28**	**10**	00:00:17
27	**26**	**11**	00:00:05
25	**25**	**12**	00:00:02
24	INFEASIBLE		00:00:01
Total Elapsed Time			00:04:49
Hypervolume			8.657

Table 10 *GAMS results for P1_2*

Makespan Constraint <=	Makespan	Total Waste	Time (Hrs)
80	80	7	00:01:14
79	67	7	00:01:22
66	50	7	00:02:41
49	**45**	**7**	00:02:10
44	**35**	**8**	00:09:24
34	**28**	**9**	00:07:12
27	**25**	**10**	00:00:17
24	INFEASIBLE		00:00:01
Total Elapsed Time			0:24:20
Hypervolume			11.297

Table 11 *GAMS results for P1_3*

Makespan Constraint <=	**Makespan**	**Total Waste**	**Time (Hrs)**
88	80	9	00:01:39
79	50	9	00:01:45
49	45	9	00:00:28
44	42	9	00:00:19
41	**40**	**9**	00:00:09
39	39	10	00:00:05
38	38	10	00:00:20
37	**36**	**10**	00:00:17
35	35	11	00:00:43
34	34	11	00:00:24
33	**33**	**11**	00:00:25
32	INFEASIBLE		00:56:30
Total Elapsed Time			1:03:04
Hypervolume			8.845

Table 12 *GAMS results for P2_1*

Makespan Constraint <=	**Makespan**	**Total Waste**	**Time (Hrs)**
80	79	68	00:09:11
78	75	69	00:06:37
74	73	68	00:08:39
72	71	68	00:03:50
70	68	68	00:04:25
67	67	68	00:03:12
66	65	68	00:02:23
64	63	70	00:01:32
62	**62**	**68**	00:03:02
61	61	69	00:13:13
60	54	71	00:12:50
53	53	74	00:10:24
52	**51**	**69**	02:01:44
50	50	73	03:56:57
49	49	72	03:57:15
48	**48**	**71**	02:01:55
47	47	76	15:05:44
46	46	76	24:22:22
45	45	73	09:33:22
44	**44**	**73**	33:50:32
43	INFEASIBLE		93:17:21
Total Elapsed Time			165:26:30
Hypervolume			9.824

Table 13 *GAMS results for P2_2*

Makespan Constraint <=	Makespan	Total Waste	Time (Hrs)
59	59	42	00:00:22
58	55	42	00:01:20
54	54	42	00:04:01
53	42	42	00:00:49
41	41	42	00:01:31
40	40	42	00:00:39
39	38	42	00:00:10
37	**37**	**42**	00:07:54
36	36	46	00:02:34
35	35	46	00:17:54
34	34	45	00:16:37
33	33	45	00:03:59
32	**32**	**45**	00:23:44
31	31	48	00:49:36
30	**30**	**48**	00:14:00
29	**29**	**49**	00:00:16
28	INFEASIBLE		34:23:19
Total Elapsed Time			36:48:45
Hypervolume			6.536

Table 14 *GAMS results for P2_3*

Makespan Constraint <=	Makespan	Total Waste	Time (Hrs)
81	68	33	00:00:57
67	57	33	00:00:20
56	54	33	00:00:28
53	53	33	00:00:11
52	52	33	00:00:10
51	47	33	00:00:33
46	**43**	**33**	00:00:31
42	**42**	**36**	00:04:33
41	INFEASIBLE		54:13:31
Total Elapsed Time			54:21:14
Hypervolume			8.655

APPENDIX D

NSGA-II RESULTS FOR P1 AND P2

Table 15 *NSGA-II results for P1_1*

Min HV		**Max HV**	
HV	8.651	**HV**	8.657
Time (Sec.)	65,64	**Time (Sec.)**	66,08
Makespan	**Waste**	**Makespan**	**Waste**
36	8	32	8
32	9	30	9
28	10	28	10
26	11	26	11
25	12	25	12

Table 16 *NSGA-II results for P1_2*

Min HV		**Max HV**	
HV	11.295	**HV**	11.297
Time (Sec.)	65,13	**Time (Sec.)**	65,20
Makespan	**Waste**	**Makespan**	**Waste**
45	7	45	7
37	8	35	8
28	9	28	9
25	10	25	10

Table 17 *NSGA-II results for P1_3*

Min HV		**Max HV**	
HV	8.833	**HV**	8.844
Time (Sec.)	68,05	**Time (Sec.)**	68,66
Makespan	**Waste**	**Makespan**	**Waste**
49	9	40	9
39	10	37	10
33	11	33	11

Table 18 *NSGA-II results for P2_1*

Min HV		**Max HV**	
HV	7.937	**HV**	9.823
Time (Sec.)	71,50	**Time (Sec.)**	71,81
Makespan	**Waste**	**Makespan**	**Waste**
58	68	58	68
53	69	53	69
51	70	52	70
		50	71
		47	72
		45	73
		44	74

Table 19 *NSGA-II results for P2_2*

Min HV		**Max HV**	
HV	5.896	**HV**	6.545
Time (Sec.)	71,49	**Time (Sec.)**	71,39
Makespan	**Waste**	**Makespan**	**Waste**
37	42	37	42
36	43	36	43
34	45	34	44
32	47	31	45
		30	46
		29	49

Table 20 *NSGA-II results for P2_3*

Min HV		**Max HV**	
HV	8.436	**HV**	8.655
Time (Sec.)	71,99	**Time (Sec.)**	71,42
Makespan	**Waste**	**Makespan**	**Waste**
43	33	43	33
		42	36

APPENDIX E

COMPARISON OF PARETO FRONTS

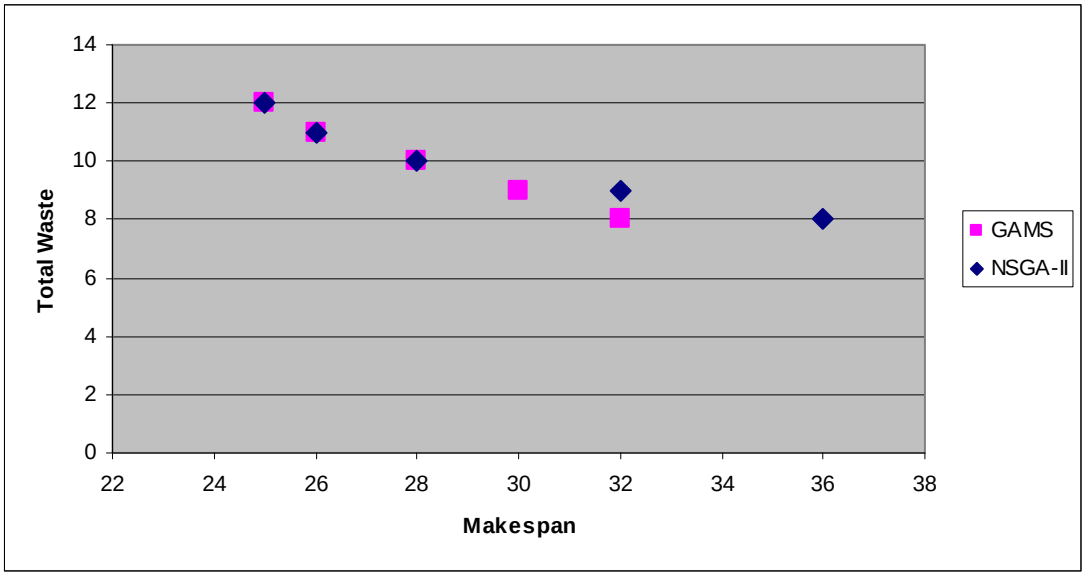

Figure 11 *Comparison of P1_1 for minimum HVR values*

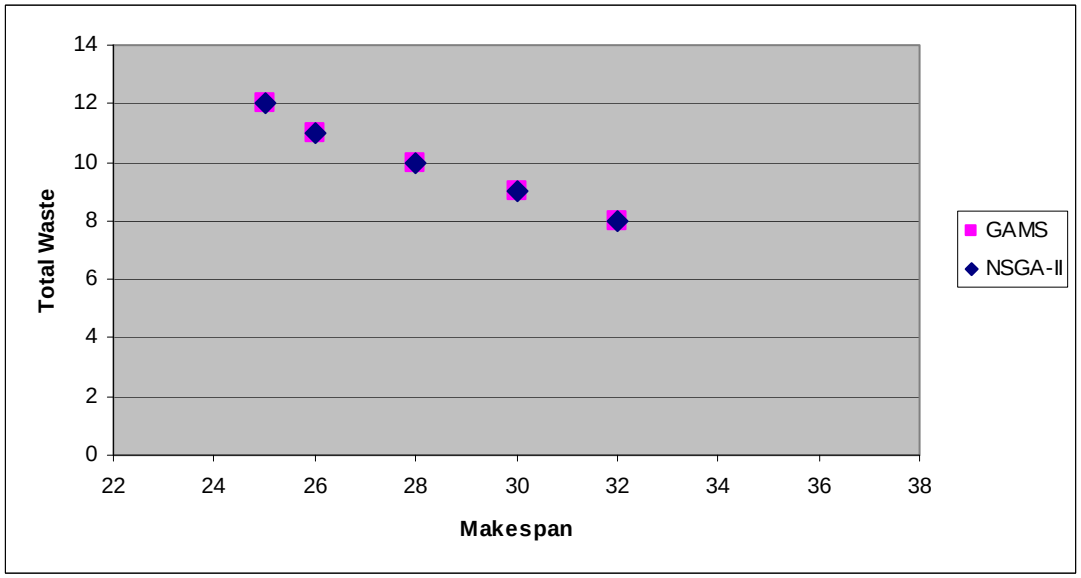

Figure 12 *Comparison of P1_1 for maximum HVR values*

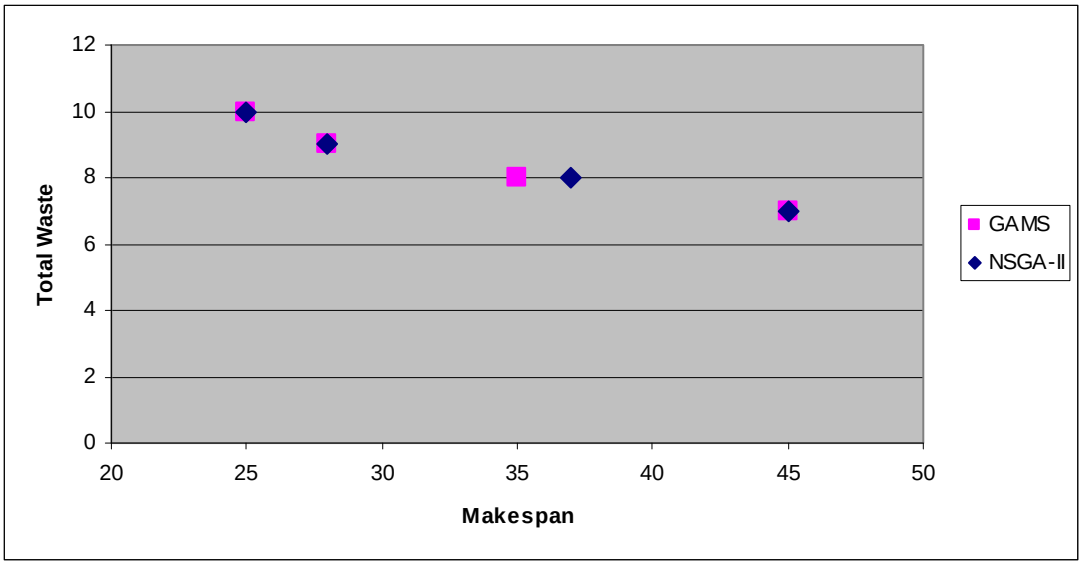

Figure 13 *Comparison of P1_2 for minimum HVR values*

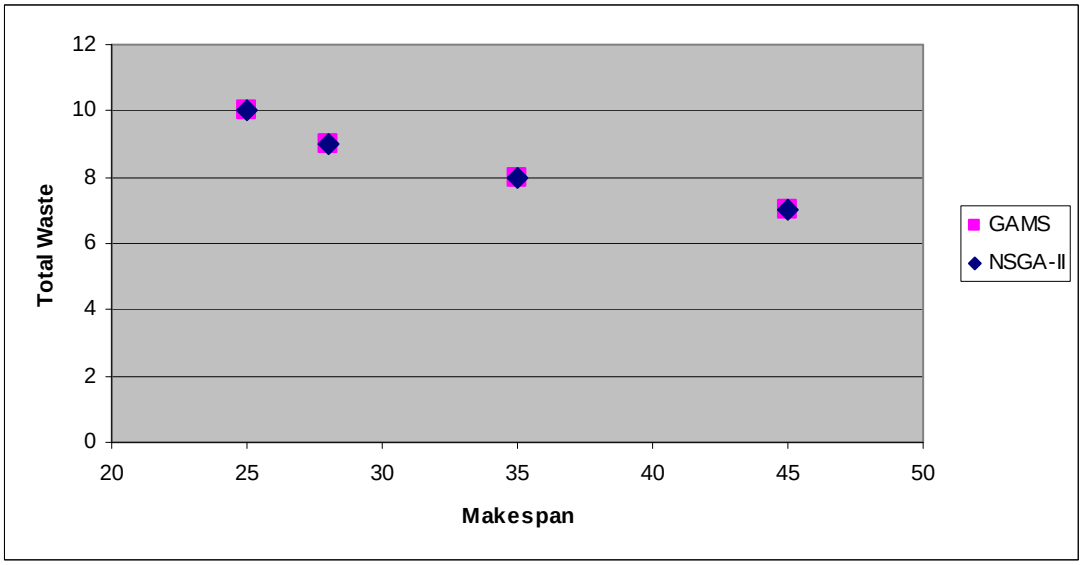

Figure 14 *Comparison of P1_2 for maximum HVR values*

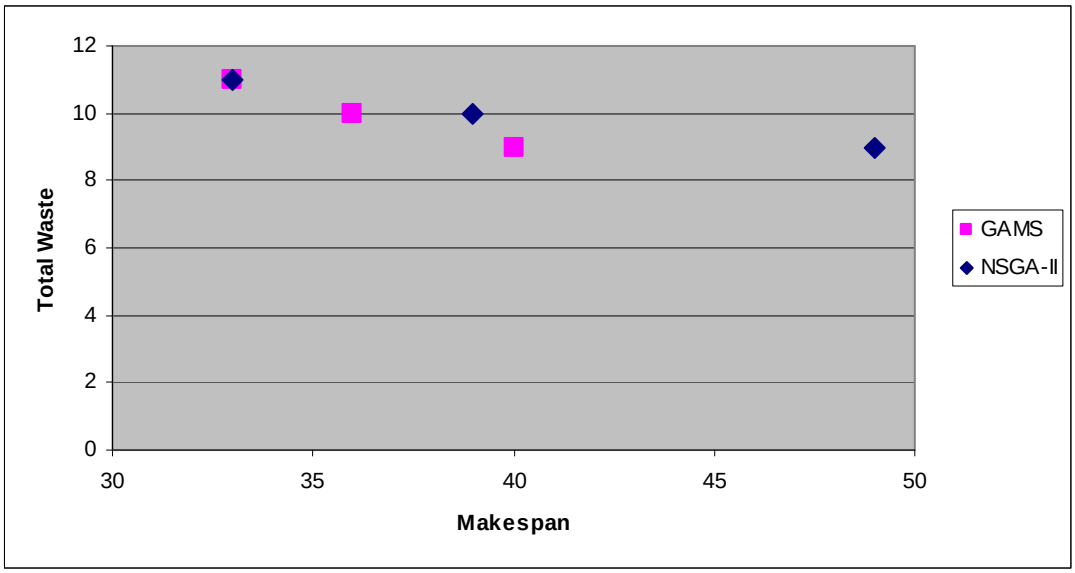

Figure 15 *Comparison of P1_3 for minimum HVR values*

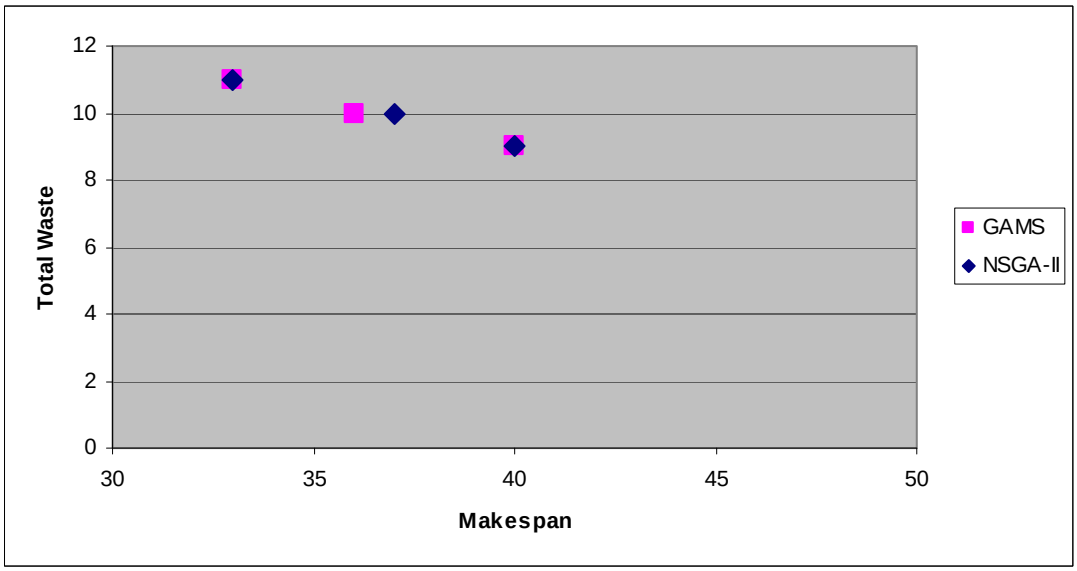

Figure 16 *Comparison of P1_3 for maximum HVR values*

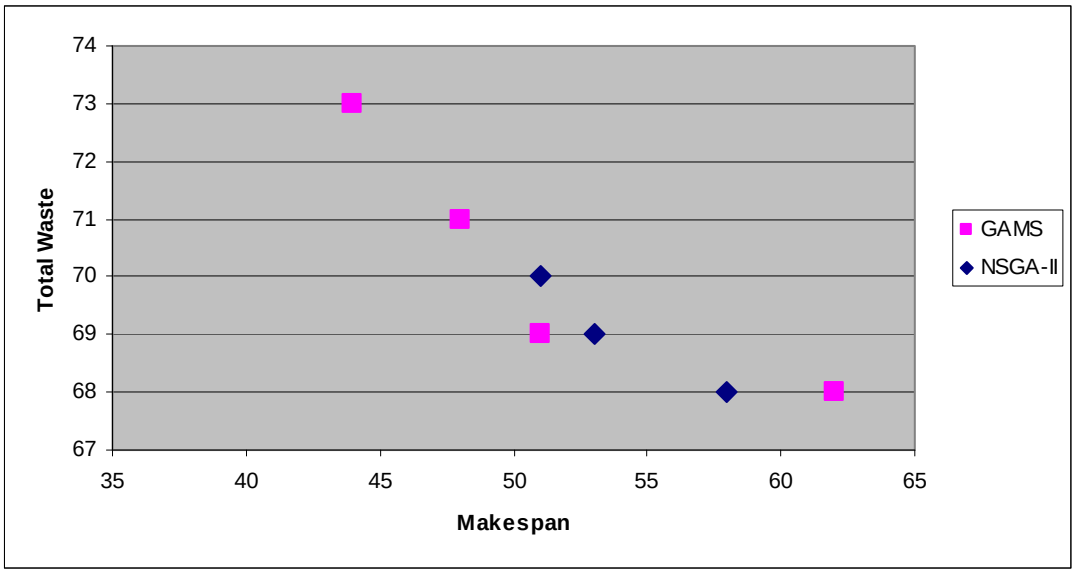

Figure 17 *Comparison of P2_1 for minimum HVR values*

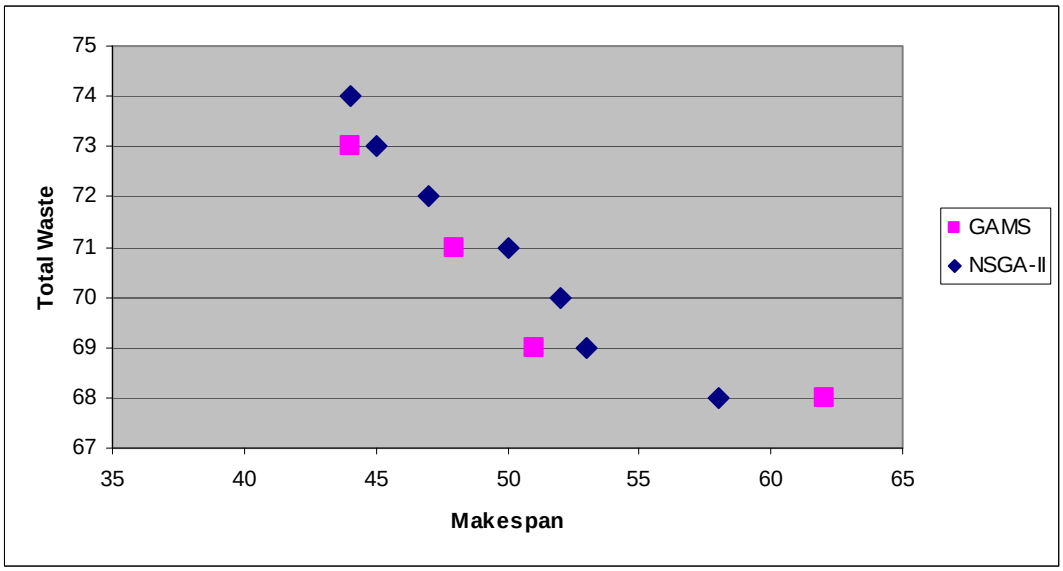

Figure 18 *Comparison of P2_1 for maximum HVR values*

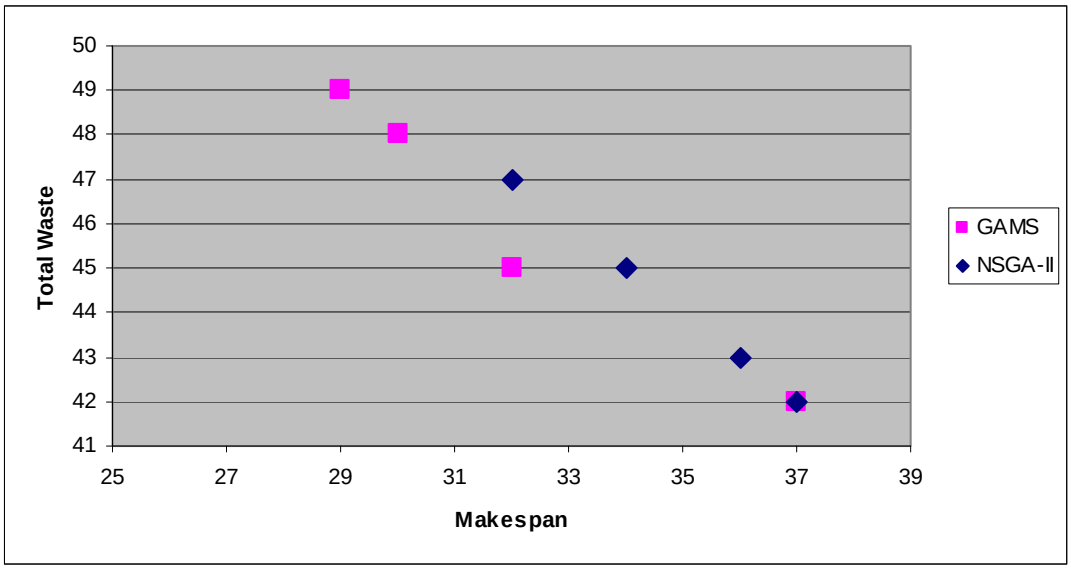

Figure 19 *Comparison of P2_2 for minimum HVR values*

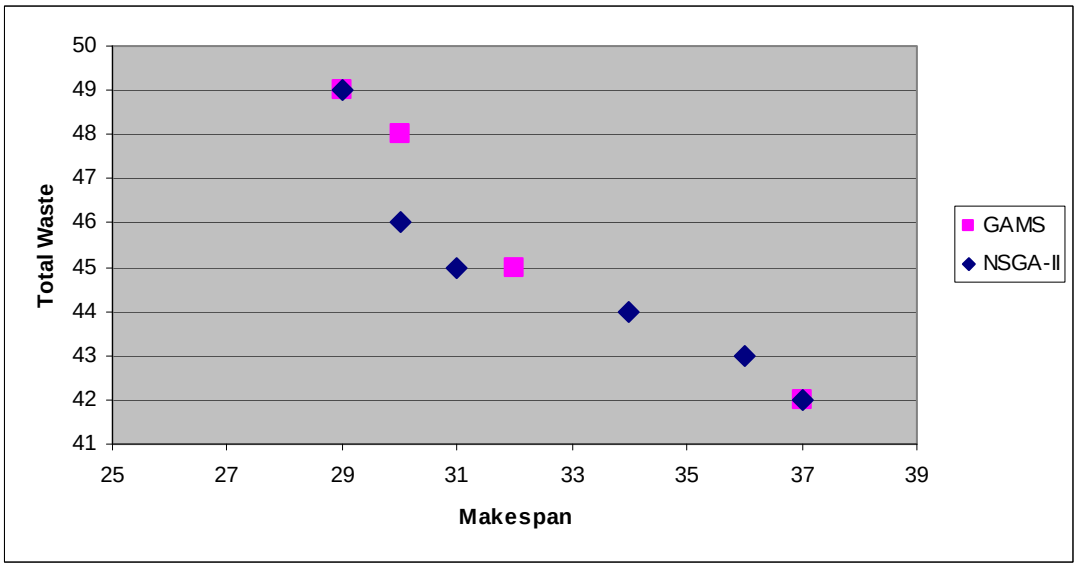

Figure 20 *Comparison of P2_2 for maximum HVR values*

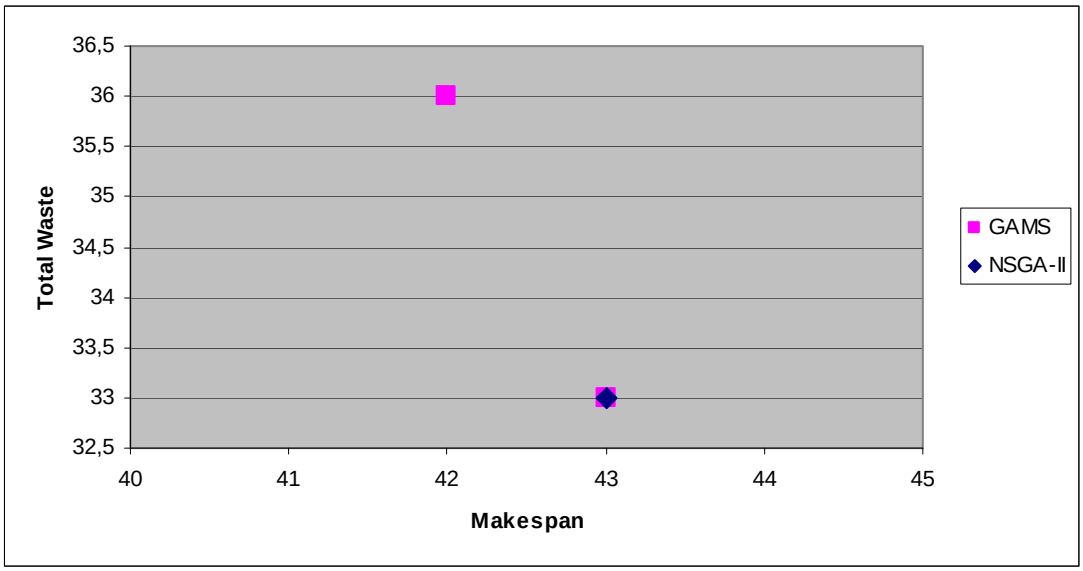

Figure 21 *Comparison of P2_3 for minimum HVR values*

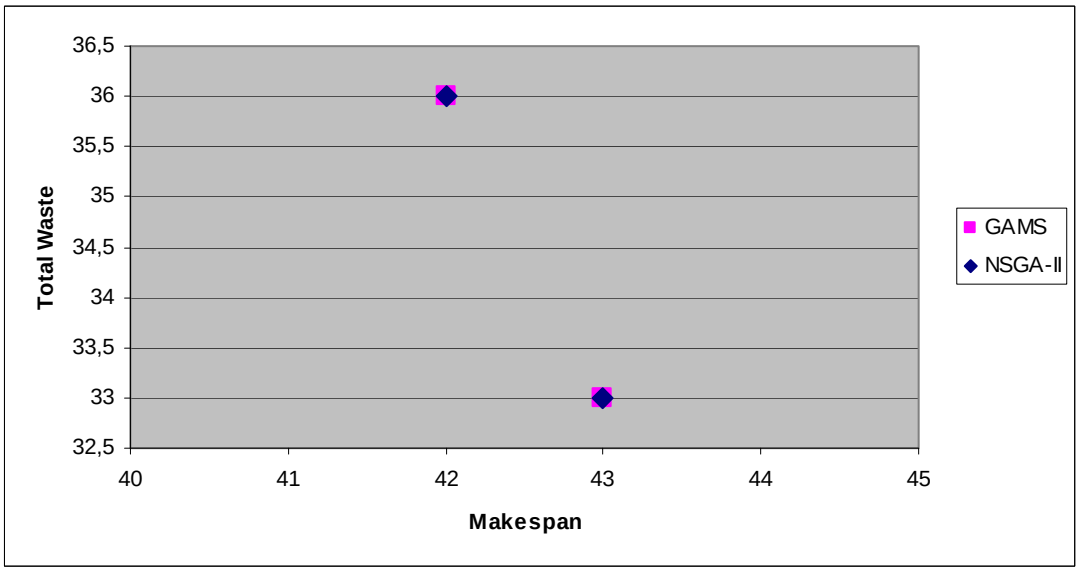

Figure 22 *Comparison of P2_3 for maximum HVR values*

APPENDIX F

COMPARISON WITH THE RANDOM SEARCH

Table 21 *Comparison for P3_1*

NGSA-II		**Random Search**	
Time (Sec.)	444,84	**Time (Sec.)**	444,84
HVR	0,7565	**HVR**	0,6751
Makespan	**Waste**	**Makespan**	**Waste**
56	3	55	37
41	4	52	52
39	5	49	57
38	10	48	61
		47	62
		46	67

Table 22 *Comparison for P3_2*

NGSA-II		**Random Search**	
Time (Sec.)	452,84	**Time (Sec.)**	452,84
HVR	0,7249	**HVR**	0,6880
Makespan	**Waste**	**Makespan**	**Waste**
61	14	65	48
53	15	61	49
48	16	60	50
44	17	53	52
42	19	50	55
		48	58
		46	67
		43	68

Table 23 *Comparison for P3_3*

NGSA-II		Random Search	
Time (Sec.)	445,31	**Time (Sec.)**	445,31
HVR	0,6890	**HVR**	0,6508
Makespan	**Waste**	**Makespan**	**Waste**
59	8	55	39
43	9	54	40
		51	61
		48	65
		46	69
		45	73

Table 24 *Comparison for P4_1*

NGSA-II		Random Search	
Time (Sec.)	489,25	**Time (Sec.)**	489,25
HVR	0,6531	**HVR**	0,4664
Makespan	**Waste**	**Makespan**	**Waste**
74	54	103	124
71	56	100	184
62	60	98	208
		94	211

Table 25 *Comparison for P4_2*

NGSA-II		Random Search	
Time (Sec.)	492,45	**Time (Sec.)**	492,45
HVR	0,6288	**HVR**	0,5142
Makespan	**Waste**	**Makespan**	**Waste**
76	69	104	137
69	70	100	164
66	71	83	190
60	73	81	205
61	72	74	210
59	76		

Table 26 *Comparison for P4_3*

NGSA-II		Random Search	
Time (Sec.)	483,17	**Time (Sec.)**	483,17
HVR	0,6799	**HVR**	0,4906
Makespan	**Waste**	**Makespan**	**Waste**
64	58	85	145
58	60	82	194
56	61	76	197
53	64		
51	66		
49	73		

Table 27 *Comparison for P5_1*

NGSA-II		Random Search	
Time (Sec.)	3.027,20	**Time (Sec.)**	3.027,20
HVR	0,7554	**HVR**	0,72662
Makespan	**Waste**	**Makespan**	**Waste**
95	55	95	173
81	56	92	175
79	63	91	192
78	64	90	198
77	66	83	202
		82	211
		79	219

Table 28 *Comparison for P5_2*

NGSA-II		Random Search	
Time (Sec.)	3.034,61	**Time (Sec.)**	3.034,61
HVR	0,8023	**HVR**	0,7608
Makespan	**Waste**	**Makespan**	**Waste**
87	31	90	99
85	32	88	113
82	33	83	114
81	34	79	115
77	36	78	129
71	37	76	136
69	40	73	138
68	42		
67	48		
66	52		
65	56		

Table 29 *Comparison for P6_1*

NGSA-II		Random Search	
Time (Sec.)	3.718,67	**Time (Sec.)**	3.718,67
HVR	0,7256	**HVR**	0,6196
Makespan	**Waste**	**Makespan**	**Waste**
96	94	154	230
93	97	152	312
92	105	144	344
91	120	142	361
90	121	137	362
		136	375
		127	376
		120	396

Table 30 *Comparison for P6_2*

NGSA-II		Random Search	
Time (Sec.)	3.746,58	**Time (Sec.)**	3.746,58
HVR	0,6966	**HVR**	0,5790
Makespan	**Waste**	**Makespan**	**Waste**
108	137	158	329
106	142	147	472
105	145	143	501
101	151	141	503
100	157	136	505
		133	547

Table 31 *Comparison for P7_1*

NGSA-II		Random Search	
Time (Sec.)	10.064,33	**Time (Sec.)**	10.064,33
HVR	0,8655	**HVR**	0,8289
Makespan	**Waste**	**Makespan**	**Waste**
77	77	106	185
75	80	78	191
74	81	77	244
69	83		
68	93		
67	96		
66	99		
65	103		
64	106		

Table 32 *Comparison for P7_2*

NGSA-II		Random Search	
Time (Sec.)	10.874,06	**Time (Sec.)**	10.874,06
HVR	0,806369	**HVR**	0,775529
Makespan	**Waste**	**Makespan**	**Waste**
102	107	126	222
99	108	111	233
95	109	110	257
94	111	107	280
93	115	106	285
91	121	104	287
90	124	102	296
89	132	98	309
88	143		

Table 33 *Comparison for P8_1*

NGSA-II		Random Search	
Time (Sec.)	13.647,59	**Time (Sec.)**	13.647,59
HVR	0,7942	**HVR**	0,6870
Makespan	**Waste**	**Makespan**	**Waste**
116	217	151	457
113	220	134	525
98	223		
97	242		
96	254		
95	261		
94	267		
92	283		
91	291		
90	308		

Table 34 *Comparison for P8_2*

NGSA-II		**Random Search**	
Time (Sec.)	13.979,11	**Time (Sec.)**	13.979,11
HVR	0,7561	**HVR**	0,6394
Makespan	**Waste**	**Makespan**	**Waste**
141	318	175	626
137	320	174	936
129	321	173	977
127	328	169	1024
122	335		
121	346		
120	352		
119	358		
116	366		