MODIFIED NICHED PARETO MULTI-OBJECTIVE GENETIC ALGORITHM FOR CONSTRUCTION SCHEDULING OPTIMIZATION

A Thesis

by

KYUNGKI KIM

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2011

Major Subject: Civil Engineering

MODIFIED NICHED PARETO MULTI-OBJECTIVE GENETIC ALGORITHM FOR CONSTRUCTION SCHEDULING OPTIMIZATION

A Thesis

by

KYUNGKI KIM

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2011

Major Subject: Civil Engineering

ABSTRACT

Modified Niched Pareto Multi-objective Genetic Algorithm for Construction Scheduling Optimization. (August 2011)

Kyungki Kim, B.S., Dongguk University

Chair of Advisory Committee: Dr. John Walewski

This research proposes a Genetic Algorithm based decision support model that provides decision makers with a quantitative basis for multi-criteria decision making related to construction scheduling. In an attempt to overcome the drawbacks of similar efforts, the proposed multi-objective optimization model provides insight into construction scheduling problems. In order to generate optimal solutions in terms of the three important criteria which are project duration, cost, and variation in resource use, a new data structure is proposed to define a solution to the problem and a general Niched Pareto Genetic Algorithm (NPGA) is modified to facilitate optimization procedure.

The main features of the proposed Multi-Objective Genetic Algorithm (MOGA) are:

- A fitness sharing technique that maintains diversity of solutions.
- A non-dominated sorting method that assigns ranks to each individual solution in the population is beneficial to the tournament selection process.
- An external archive to prevent loss of optimal or near optimal solutions due to the random effect of genetic operators.

- A space normalization method to avoid scaling deficiencies.

The developed optimization model was applied to two case studies. The results indicate that a wider range of solutions can be obtained by employing the new approach when compared to previous models. Greater area in the decision space is considered and tradeoffs between all the objectives are found. In addition, various resource use options are found and visualized. Most importantly, the creation of a simultaneous optimization model provides better insight into what is obtainable by each option.

A limitation of this research is that schedules are created under the assumption of unlimited resource availability. Schedules created with this assumption in real world situations are often infeasible given that resources are commonly constrained and not readily available. As such, a discussion is provided regarding future research as to what data structure has to be developed in order to perform such scheduling under resource constraints.

DEDICATION

This work is dedicated to my beloved family. Without their support and love, I would never been able to complete this work.

ACKNOWLEDGEMENTS

I would like to express my appreciation to my committee chair, Dr. Walewski and my committee members, Dr. Zechman and Dr. Julian Kang, for their guidance and support throughout the course of this research. To my colleagues, Minsoo Kim and Hessam Sadatsafavi, I give my thanks for being together during my struggling times at Texas A&M University. Finally, thanks to my parents and sister for their encouragement.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

1. INTRODUCTION AND PROBLEM DEFINTION

For large construction projects, managing the efforts of project participants and activities towards the goal of completion is of utmost importance for successful delivery. Because of that, schedulers or modelers should create schedules taking into account multidisciplinary goals and various project conditions. With the coordinated plans with accurately predicted consequences, decision makers can make a scheduling decision that satisfies multiple requirements. However, it is very difficult to generate guaranteed optimal schedules since most construction scheduling problems are complex especially when there are many objectives to achieve.

The Critical Path Method (CPM) is one of the most well-known scheduling methods that were invented to achieve greater activity coordination. CPM's invention was prompted by prevailing deficiencies in existing project planning and scheduling systems and has been in wide use. Many scheduling systems were later developed based on the CPM technique in order to accommodate the needs arising from enhanced scheduling objectives and the sheer number projects that are often more complex.

Among the techniques, the heuristics is a category that has advantages over other approaches such as analytical method and exhaustive enumeration. In addition to the basic function of generating better solutions, such methods have the capacity to support decision making by providing a wide range of alternative solutions when applied to

This thesis follows the style of Economics and Mathematical Systems.

problems with multiple objectives. The capacity to deal with multi-attribute problems is an essential part for a model for complex construction schedule with many activities and objectives.

As well as supporting decision making, multi-attribute scheduling makes schedules realistic. For multidisciplinary construction projects, multiple participants and objectives need to be integrated into a model. Important criteria may include minimum project duration, cost and variation in resource use. Previous approaches attempted to optimize these objectives while figuring out relationships between them. However, some of the relationships are not revealed by the models as the approaches adopted traditional approaches that perform resource leveling only after optimal tradeoffs between cost and duration are found.

2. CONSTRUCTION SCHEDULING AND OPTIMIZATION FOR DECISION SUPPORT

2.1 Construction Scheduling

2.1.1 Elements

2.1.1.1 Activities (Tasks)

Activities are components of a project that should be completed before the project deliverable is considered to be completed. Also, a construction schedule can be defined by its activities and relations between them. The performance criteria of activities can be estimated in terms of duration, cost, and resource use.

2.1.1.2 Precedence Relations

For technical and managerial reason, a set of activities should be completed for another activity to start. For example, concrete placement can be performed only after the form is placed [10].

Also there are generalized relationships such as start-start (SS), finish-finish (FF), finished-start (FS), and start-finish (SF) that explain other types of relationship between activities. Minimal and maximal time lags describe the activity precedence relationship between multiple activities.

When s(A) is the start time of activity a and f(B) is the finish times of activity b, generalized relationships between two activities can be expressed as:

- $s(B) \geq s(A) + d$ (SS; demotes, activity B can start d time after activity A starts)
- $s(B) \geq f(A) + d$ (FS; denotes, activity B can start d time after activity A finishes)
- $f(B) \geq f(A) + d$ (FF; denotes, activity B can finish d time after activity A finishes)
- $f(B) \geq s(A) + d$ (SF; denotes, activity B can finish d time after activity A starts)

2.1.1.3 Resources

Resources include construction material, labor, and money that are needed in order to perform the activities of the project. Since the availability of resources often define the problems associated with construction projects, it is very important to properly consider resource in the scheduling process.

2.1.2 Objectives Employed in Construction Scheduling

For construction, there are several objectives to be achieved such as duration minimization, net present value minimization, quality maximization, cost minimization, total earliness of activities minimization, and total tardiness of activities minimization. The objectives to be optimized in this thesis are construction duration minimization, construction cost minimization, and minimum resource use variation.

2.1.2.1 Duration Minimization

Total construction duration is the duration between the starting time of the first activity and the finishing time of the last activity. When the duration is minimized, a time-critical path is generated.

2.1.2.2 Cost Minimization

Construction project cost involves minimization of direct cost of project activities, minimization of cost resulting from fluctuation in resource use and minimization of penalties from earliness/tardiness.

2.1.2.3 Optimal Resource Requirements

Fluctuation in resource use should be reduced to avoid the difficulties of frequent hiring and firing and loss of learning effects of labors. Optimal resource requirements can be achieved by resource smoothing that adjusts activity dates without changing the total construction duration.

2.2 Invention and Technical Development of Critical Path Method: Literature Review

Before the Critical Path Method was invented in the late 1950s, existing planning systems had deficiencies such as lack of coordination and oversimplification that prompted an invention of a method to obtain a higher degree of coordination of project activities toward a single goal [12]. At that time, project groups had worked independently with their own plans and schedules, and detailed planning and scheduling were developed based on gross estimates of entire project and past experiences.

Critical Path Method had been intensively used for various forms of projects for more than two decades after its invention. However, as surveys conducted in the UK and

Egypt by Allam (1988) showed [1], there were growing doubts about applying this analytical method to real projects due to its arithmetic complexity. Therefore, many methods were developed based on this technique to deal with this issue. The approaches can be categorized into analytical (mathematical) and heuristic methods. Mathematical methods aim to calculate optimal solutions with accuracy and heuristic methods generate optimal or near optimal solutions depending on assigned priorities such as cost and duration. Although both mathematical and heuristic methods have their strengths and weaknesses according to a review conducted by Leu, et al. [14, 15], heuristic methods work better and are more in use for multi-objective scheduling because of their multiple advantages. Large scale, multi-objective construction scheduling problem is a kind of NP-hard problems, which stands for non-deterministic polynomial-time hard problem. For an NP-hard problem there is no known method of finding optimal solutions in polynomial time. Complexity and limited resources make it hard to solve real scheduling problems with mathematical methods. Heuristic models are capable of solving this kind of complexity more easily because of the simple format and easy application. The disadvantage of using this method is that it is problem-dependent and it does not always guarantee optimal solutions [1, 15].

Mathematical models to generate optimal schedules and optimal solutions were developed [8, 13]. Later, mathematical programming formulations were developed and discussed by Easa and Harris [4, 9]. However, it was only applicable to small projects with few activities because a great deal of computation effort was needed to create the mathematical formula.

Due to this limitation, the majority of efforts to date have been heuristic scheduling methods. Senouci and Eldin (2004) developed a single-objective genetic algorithm, and its objective is to find a schedule with the minimum project cost under resource and duration constraints. The solution encoding structure is composed of activity duration part and start date part. Although multiple resources, time-cost tradeoffs were integrated considering all possible activity relationships, this method did not provide any insight into decision options and the obtainable consequence because its objective was to generate a single solution [20]. In 2008, Senouci and Al-Derham used Multi-objective Genetic Algorithm to minimize project duration and total project cost. Construction material, crew and overtime were combined to create resource utilization options. This model shows optimal and near optimal trade-offs between cost and duration, however resource leveling did not take place in this model [19]. Leu and Yang (1999) proposed a model to search optimal combinations of project cost and duration under limited quantities of resources. It is a multi-objective scheduling model under resource constraint using Genetic Algorithm-based searching technique. Leu and Yang (1999) also proposed a computational multi-criteria scheduling optimization model that integrates a time/cost tradeoff model, limited resource allocation model, and resource leveling model. Though this is an advanced model compared to their previous model, it failed to consider relationships between the degree of resource leveling and other objectives. Later in 2000 these researchers, added a decision support system to the previous research in order to assist scheduling decision makers with the optimization result. However, the inability to consider the relationships between the degree of resource leveling and other objectives was not solved in this research [16].

The heuristic methods developed for construction scheduling often claim to have integrated important scheduling criteria into a single optimization. However, the relationship between all the competing objectives and provide a clear insight into the problem have not been a focus of these models. Understanding that this limitation arises from the traditional scheduling procedure where resource leveling is performed after optimizing cost and duration, this research aims to propose a heuristic model that supplements the drawbacks. The focus of this research is on developing an advanced heuristic scheduling model that can be applied to complex scheduling problems.

2.3 Multi-objective Optimization for Scheduling Decision Support

2.3.1 Decision Making

The role of a scheduler or a modeler can be distinguished from that of a decision maker (DM). The scheduler is responsible for informing the DM with enough information about what is obtainable from alternative solutions. Based on the information, the DM makes a decision using specific criteria and makes modifications to the schedule as needed. Taking that into account, it is of critical importance that a scheduler provides the DM with best alternatives with the prediction of obtainable results. With that insight, the DM understands what is obtainable and what tradeoffs between objectives have to be considered.

Conventional decision making methods for project scheduling have three steps to reach a decision:

1. Requirements of multidisciplinary stakeholders are studied by a participant such as designer, scheduler to generate options.
2. Professions from different disciplines build models to conduct analysis and determine feasibility of the options.
3. Decision is made according to effectiveness of each solution.

Taking into account the role of a scheduler as a decision supporter, following this procedure for decision making has some drawbacks. Most of all, sufficient options are not guaranteed since only a limited number of options can be generated and evaluated due to limited time and resource using such a procedure. In this case, decision makers have to make decisions from a limited set of options and then implement it. Furthermore, evaluating options one by one is an ineffective and significantly time-consuming task when requirements are not pre-integrated while alternatives are created. It depends heavily on a person's insight into the problem since options are generated based on perceived requirements from stakeholders.

Since generating and evaluating enough solutions for a large construction project using traditional decision making procedure is not efficient, it is greatly beneficial if scheduling requirements can be pre-integrated when schedule options are generated relying on a computer's process speed. As introduced in the literature review section, various kinds of scheduling techniques have been developed in order to make the schedule more realistic and satisfactory. They have been developed in a way that more requirements are integrated and rely less on mathematical computation. Major criteria for scheduling decision may include project duration, cost, resource leveling and other considerations such as safety and distance resource consideration [5].

2.3.2 Multi-objective Optimization

Optimization is a type of modeling method to determine the best solutions from available alternatives. For each solution, a quantitative evaluation is provided rather than a subjective one. When it is used for optimization for a multi-objective decision making support, the aim can be to find a group of non-dominated solutions forming trade-offs called the Pareto frontier, as opposed to single-objective optimization which is to find a single best solution.

In the Figure 1 below, the concept of Pareto optimal for bi-objective minimization optimization is demonstrated. Pareto frontier is the line that links five individual solutions from P1 to P5. A solution is Pareto optimal when one fitness value of the solution cannot be upgraded without degrading another fitness value.

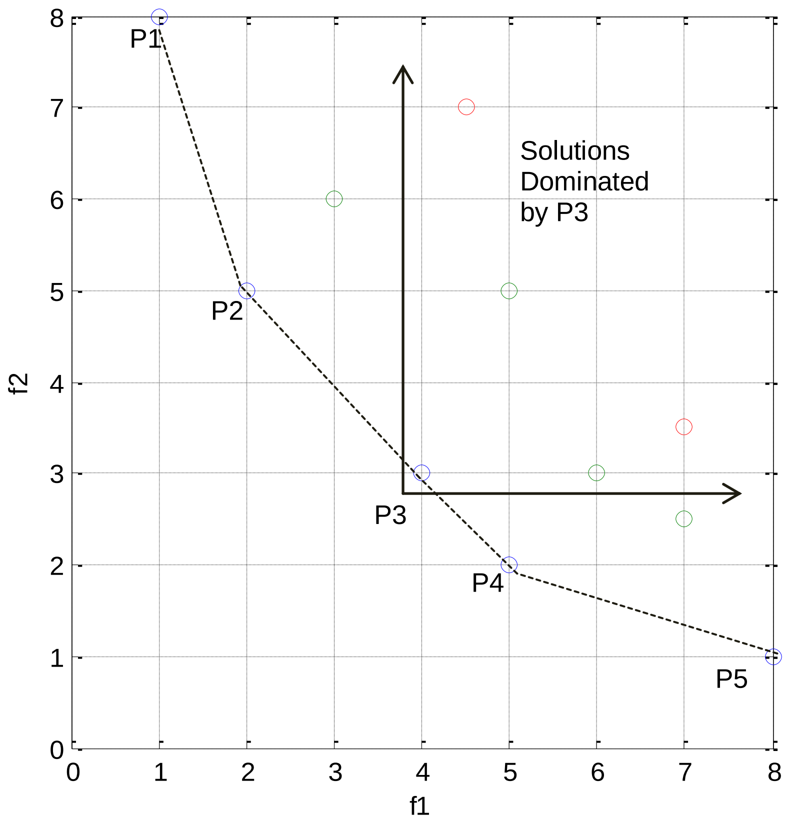


Figure 1 Pareto front for bi-objective minimization problem

P1, P2, P3, P4 and P5 are optimal solutions because there is no solution that has objective values f1 and f2 both are better (lower) than one of the five solutions, and a fitness value of one solution cannot be reduced without increasing the other. As can be seen, no solution is found under the dotted line that links the optimal solutions. On the upper right corner of the figure, four solutions are found to have worse values for both f1 and f2 than solution P3. In this case, P3 dominates the four solutions while P3 is a non-dominated solution. By using Pareto frontier, we can concentrate on this optimal trade-off without having to consider any inferior solutions.

In this regards, decision making can benefit from using multi-objective optimization techniques when solving complex scheduling problems that usually involves coordination of many activities and evaluation in terms of multiple objectives. The need for optimization for scheduling arises from the huge number of possible solutions to a problem. Each solution is composed of many decision components, and there are alternatives for each component forming a decision space for a problem. The size becomes exponentially larger as more decision components are added as seen in Figure 2. A scheduler cannot choose and evaluate all possible alternatives in the space due to limited time and resource. It becomes even complex when each possibility is evaluated by multiple criteria.

5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

Figure 2 A chromosome with 20 decision components

In Figure 2, a chromosome is presented to explain the complexity of problems and difficulty of evaluation related to scheduling. In general, a chromosome is representation of a solution that is composed of a set of parameters or decision elements. A wide variety of data structure such as binary arrays or real values can be used for each chromosome component. A more detailed explanation about chromosome will be given in the Genetic Algorithm section. In the array of cells, the number in each cell represents a component of a decision. Assuming that each cell can have one of six integer values between 0 and five, there are 6^{20} (3,656,158,440,062,980) possibilities scattered in the decision space. Evaluating each point is exhaustive, and solutions selected and evaluated by subjective opinion are apt to be sub-optimal. Evaluation and selection become more difficult in situations of multi-objective optimization because a change in one criterion may degrade others.

There are single objective optimization and multi-objective optimization depending on the number of criteria the optimization algorithm aims to minimize or maximize. Construction scheduling problems often involve many criteria. Though there are ways single-objective optimization can deal with multi-objective problems - such as constraint method and weighting method - they have disadvantages as a decision support model. Constraint methods need to pre-specify levels of constraints before performing several runs which is infeasible because we do not know the ranges of solutions before optimization and the size of decision space is too large. It is also necessary for weighting methods to assign weights for objectives before optimization which can be classified as decision making not decision supporting. Implementation of the two single-objective optimization requires decision making supporters to make some decisions which are a decision makers' responsibility. However, Multi-Objective Optimization has a capacity generate many non-inferior solutions that provide the decision maker with the insight into the problem.

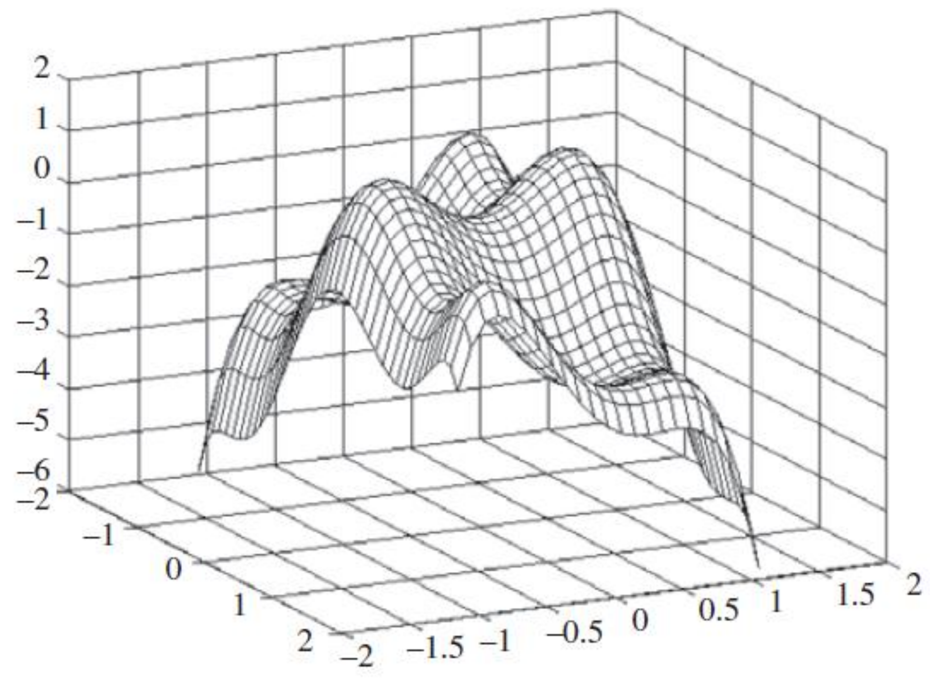


Figure 3 Function D3 (source: Jie, Kharma et al, 2010)

The idea of presenting a group of Pareto points for decision making is to provide the decision maker a clear sight of what is achievable in what area of decision space thus leading the decision maker focus on a certain region rather than spending time and resource in assessing solutions in an area without best solutions or randomly exploring the solution space. Figure 3 provides a search space for one objective optimization considering two factors optimized by Jie, et al. [11]. The fitness values are difficult to predict and formularize since there are multiple maximum and minimum points. The formula of function D3 is shown below:

$$D_3(x,y) = -\left(\left(4-2.1x^2+\frac{x^4}{3}\right)x^2+xy+(-4+y^2)y^2\right) \tag{2.1}$$

$$x \in [-1.9, 1.9], y \in [-1.1, 1.1].$$

Since multi-objective optimization does not make any modification to the solutions and helps to develop a family of best solutions, by using this method, decision supporters can provide alternatives that are not dominated by any other solutions, and decision makers can avoid choosing an inferior solution in their managerial decision making. One of the distinguishing advantages of multi-objective optimization is its mechanism considering tradeoffs among several objectives in selecting solutions. In order words, multi-objective optimization evaluates candidate solutions for all the objectives thus treating the objectives equally important while single objective optimization method concentrate on finding one point in the decision space setting other values fixed.

By integrating several objectives into optimization model, models become realistic and capable of providing better information for decision making. A decision for a complex problem such as construction scheduling entails consideration from viewpoints of different stakeholders with various objectives. Taking many objectives into the decision making process makes analysis more practical, but doing so adds complexity to it at the same time [2]. Furthermore, due to the characteristics of real world problems that is becoming more complex, there is a number of sub-decisions and following consequences related to a decision that make it harder to evaluate an impact of a decision component on the overall result. Evaluating the results is not simple and easy because of interrelations between several objectives.

2.3.3 Genetic Algorithms

Genetic algorithms were invented in the 1960s by John Holland is search of a heuristic method that mimics the mechanism of natural adaptation. The procedure begins by generating a population of randomly generated candidate solutions that evolves towards an optimal solution through genetic iteration. The population is composed of string arrays each of which contains information on a single solution which is called chromosome. In each generation, candidate solutions are evaluated and selected based on fitness function which indicates how well the solution solves the problem. After a portion of the population is selected based on the fitness values, remaining solutions are combine and mutated by genetic operators such as crossover and mutation. Crossover operator combines candidate solutions and a mutation operator randomly mutates them. This process continues for each generation until a solution of a certain value is obtained or it is iterated by a predetermined times. The operation of Genetic Algorithms is visualized in Figure 4.

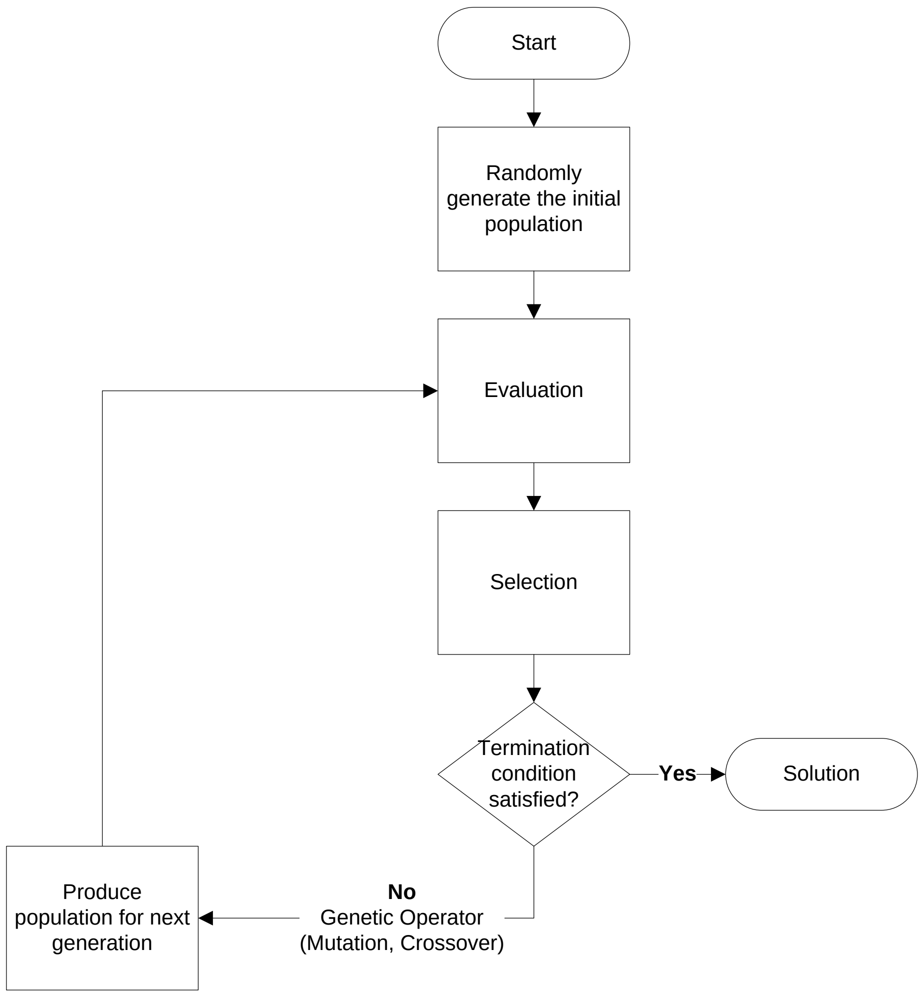

Start
Randomly generate the initial population
Evaluation
Selection
Termination condition satisfied?
Yes
Solution
No
Genetic Operator (Mutation, Crossover)
Produce population for next generation

Figure 4 Optimization procedure of genetic algorithms

A distinctive strength of Genetic Algorithms is its ability to enable a solution search without having to know so much about the domain of a problem. With little knowledge and information about the problem domain, this approach relies on the computer's process speed for finding solutions from the entire search space using constraints and fitness functions. Genetic Algorithms have been used because of the advantages in exploring the possible solutions. They have been applied to science, engineering problems such as large CPM problems and proved to be efficient for searching optimal solutions in a large solution space [6]. Those advantages make genetic algorithms one of the most effective search methods for a complex problem with a large search space [20].

Genetic Algorithms can be used for one optimum value search or multidisciplinary optimization where tradeoff occurs among multiple objective functions. The latter is useful especially for a complex problem that has to meet the requirements of multiple participating disciplines. In this case, a group of optimal solutions are provided by the algorithm and the matter of selecting one decision from alternatives can be left to a decision maker. Genetic Algorithm can guide the search towards the Pareto frontier in order to enable a decision maker to be informed of the best trade-off possible and avoid excessive effort for evaluating sub-optimal points in the space.

Since mechanism of generating initial population is constructed and then the GA algorithm improves the population towards the Pareto optimum, it is important to have a proper chromosome structure that encodes solutions. If too much information on the problem domain is included in the algorithm like most analytical methods do, the genetic algorithm becomes too problem specific. Genetic algorithms have to be structured in a way that relies on capacity of computer more than problem specific formula.

2.3.4 Niched Pareto Genetic Algorithm

Genetic algorithms have been applied almost exclusively to single-attribute problems. However, many real-world problems are revealed as that their objective functions are multi-attribute. And, solutions do not spread linearly as can be seen in Figure 3. In general, solution space has several local maximums and minimums. GA methods with constraints and weights have been used as tools for combining multiple attributes. However, these methods are very sensitive to variations in the penalty function coefficients and weighting factors. Here, the need arises for Multi-objective Genetic Algorithms (MOGA) that finds good solutions overcoming this defect.

The purpose of Niched Pareto Genetic Algorithm is to generate optimal solutions called Pareto optimal while maintaining diversity at the same time. It produces optimized tradeoffs between conflicting objectives by finding non-dominated samples all along the Pareto front. According to a study comparing eight diversity-maintaining methods for multi-modal problems, sharing method can find out all the peaks although suboptimal solutions are included in the final population [21]. In addition to this strength of NPGA, the proposed model for this research supplements deficiencies of it by integrating effective features such as search space normalization method and external archive. Search space normalization technique is adopted in order to avoid scaling deficiency occurring while integrating multiple objectives of different scales. External archive is included in the model to prevent loss of good solutions due to random effects of genetic operators (crossover and mutation operators). Detailed explanations for them are provided in Sections 4.2.2 and 4.2.3, respectively.

In each generation in genetic algorithm iteration, a selection mechanism is used to select solutions in the population. Chosen solutions form a mating pool where crossover and mutation occur and provide the basis for the next generation. Tournament selection has been one of the most widely used selection mechanisms. In tournament selection, individual solutions of a predetermined size are selected at random, and a solution with the best fitness value is selected as a winner. The process is repeated until the desired size of mating pool is formed. Though the tournament selection has benefits such as efficiency in coding and easiness in adjusting the tournament size, solutions in a population tend to converge to a uniform solution after a large number of generation iterations [7]. Since NPGA intends to generate multiple points along the Pareto front the attribute space, it has to avoid convergence to a single point and maintain multiple solutions. Thus, two mechanisms were created: Pareto domination tournament and Sharing on the non-dominated frontier.

Pareto domination tournament is used since more domination pressure and control of that pressure are needed to know an individual's true domination ranking. The sampling scheme of NPGA is as follows.

1. Randomly select two candidate solutions and a comparison set.
2. Each candidate is compared against individuals in the comparison set.
3. Comparing candidates with comparison set.
   3.1 Non-dominated solution is selected for reproduction if one candidate is dominated by the comparison set and the other is not.
   3.2 'Sharing' is used when neither or both are dominated by the comparison set.

Figure 5 Pareto domination tournament of NPGA

In this process, sample size t_{dom} maintains the domination (selection) pressure. The goal of sharing that appears in 3.2 in Figure 5 is to distribute the solutions over different local optimums in the search space allocating individual solutions in the population in proportion to the magnitude of the peak. Shared fitness is calculated as following equation.

$$shared\ fitness = \frac{f_i}{m_i} \tag{2.2}$$

where, f_i is individual i's objective fitness and m_i is niche count (how crowded is the neighborhood of individual i)

$$m_i = \sum_{j \in Pop} Sh\left[d\left[i, j\right]\right] \tag{2.3}$$

where, $d\left[i, j\right]$ is distance between individual i and j. $Sh\left[d\right]$ is sharing function and defined as below.

$$Sh\left[d\right] = \begin{cases} 1 - d / \sigma_{share} & if\ d \le \sigma_{share} \\ 0 & else \end{cases} \tag{2.4}$$

Based on equation1, 2, and 3, individuals with σ_{share} distance of each other degrade each other's fitness. Thus, the convergence occurs within a niche. Figure 6 shows the pseudo code for NPGA. Sharing distance can be determined by "dividing the search space into a number of equal sized hyper-space equal to the number of sought out optima [3]." This selection mechanism of NPGA prevents generation of similar solutions after generation runs and thus well-distributed solutions can be obtained. When a decision maker is provided with a well distributed Pareto front, a decision can be made effectively considering greater space of solutions that could be ignored without having to explore any dominated solutions.

```
Initialize Population P
Evaluate Objective Value
For i=1 to g do
    Specialized Binary Tournament Selection
    Begin
        if Only Candidate 1 dominated then
            Select Candidate 2
        else if Only Candidate 2 dominated then
            Select Candidate 1
        else if Both are Dominated or Non-dominated then
            Perform specialized fitness sharing
            Return Candidate with lower niche count
        end if
    End
    Single Point Crossover
    Mutation
    Evaluate Objective Values
End for
```

Figure 6 Pseudo code of niched pareto genetic algorithm

3. CURRENT NEEDS AND RESEARCH SIGNIFICANCE

3.1 Previous Approaches

As stated in the previous sections, many multi-objective genetic algorithm models (MOGA) were proposed to solve scheduling problems. Important scheduling criteria were minimum duration, minimum cost and minimum variation in resource use and so on. However, none of the previous construction scheduling models has capability to thoroughly explore the entire solution space because of their sequential optimization process. The purpose of applying optimization methods is to solve complex scheduling problems using the useful traits: developing best possible solutions considering the tradeoffs between conflicting objectives. Though resource leveling is an importance criterion for a successful construction execution because of the negative impacts of variation in resource use on cost and productivity, previous approaches did not take into account the relationships between resource leveling and other objectives. Reducing variation in resource use promotes workers' loyalty and captures the benefits of learning, and ultimately saves cost by enhanced productivity and improved morale. Cyclic hiring and firing destroys workers' morale. Thus, resource leveling has to be considered equally important as other optimization criteria.

Resource leveling becomes important when there are sufficient amount of resource while it is important to minimize project duration extension for fixed-limits scheduling [4]. Then, for unlimited resource scheduling, fluctuations or deviations from desired resource use can be minimized without constraints of resource availability in order to avoid undesirable loss.

When a schedule is generated by traditional Critical Path Analysis, resource leveling is not incorporated when selecting shortest schedules. After a schedule is selected, manpower leveling starts and smoothing is performed until the desired curves are obtained [12]. The deficiency of this traditional method is that it is impossible to find different solutions other than initially found schedules after the schedule selection based on shortest time and lowest cost.

None of previous approaches has overcome this deficiency. Resource leveling model improvised by Leu also takes three steps as following [14]:

1. Analyze a time/cost trade-off model
2. Non-dominated solutions with project duration and cost are found
3. Another system receives information about non-dominated schedules from the process 2 and perform resource leveling process

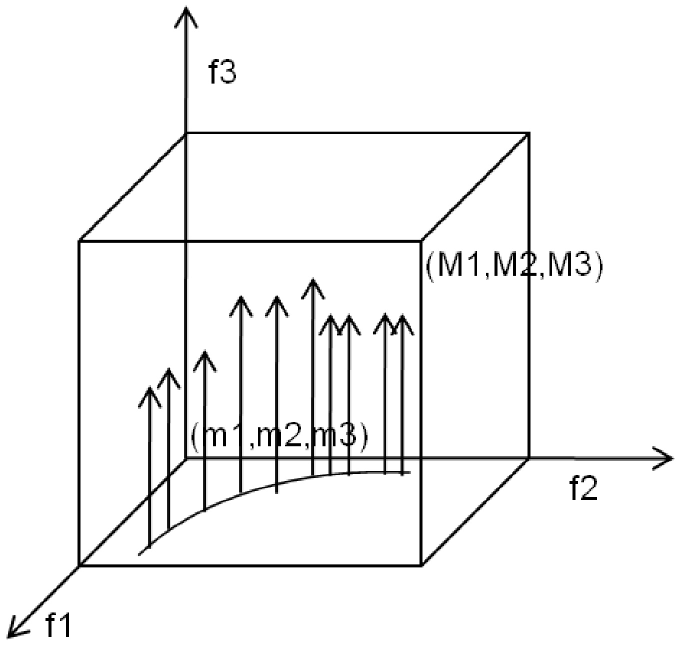


Figure 7 Optimum solution search by previous methods

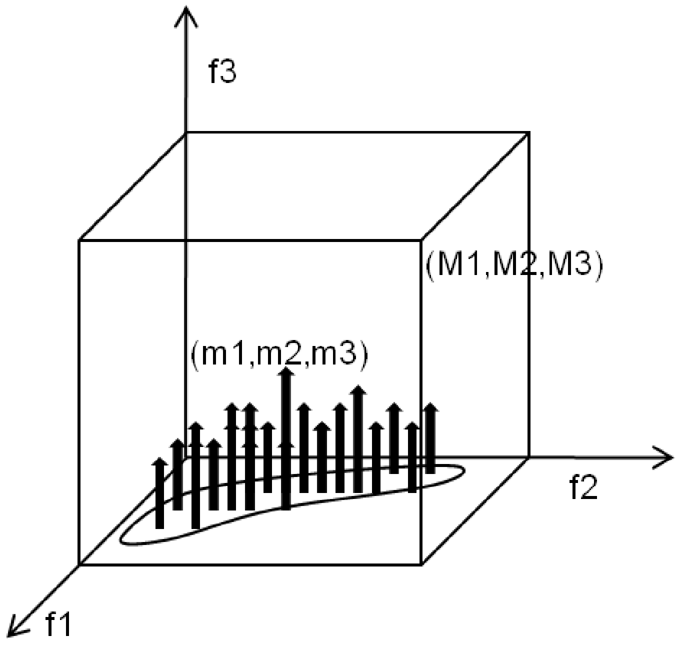


Figure 8 Desired optimum solution search

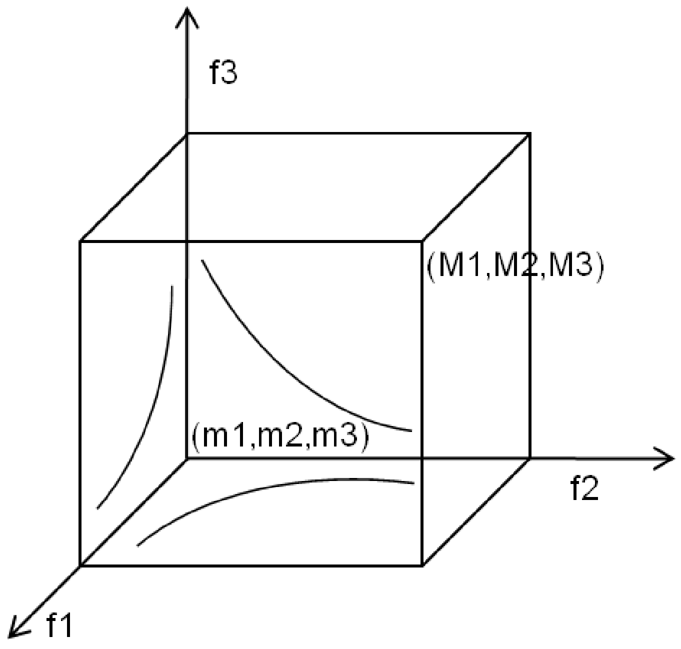


Figure 9 Tradeoffs between three objectives

Optimization process using this method is divided into two phases. This makes it impossible to optimize resource leveling simultaneously with cost and duration optimization. Here, optimal solutions are restricted to solutions above optimal tradeoff initially found on the XY-plain (time-cost tradeoff) in the objective space as seen in Figure 7. From that, it can be seen that combinations of costs and durations have to change depending on different level of resource leveling as seen in Figure 8. In order words, all the tradeoffs between objectives should be presented can be seen in Figure 9 that visualizes tradeoffs between three optimization criteria which past approaches could not achieve.

The new model proposed in this research addresses this drawback under the unlimited resource assumption. The model is capable of simultaneous generation of optimal or near optimal schedules in terms of multiple objectives (minimum duration, cost and resource leveling index) and thus better exploration into the solution space. Optimizing objectives one by one results in solution search that is not enough to provide the decision makers (DMs) with the insight into the best available alternatives because this is misses some part of the objective space and does not consider possible trade-off relations between conflicting objectives. Simultaneous optimization for all objectives is important for thorough objective space exploration and non-dominated solution search as visualized in Figure 9.

3.2 Research Significance

When creating schedules with the three criteria, all of them have to be pre-integrated into the model and all the tradeoffs between objectives have to be revealed. However, due to the drawbacks of previous approaches, it has not been possible to explore whole possible area in search space. From this research, a programming solution will be presented that enables simultaneous optimization. More importantly, an informed decision making related to construction scheduling problems will be enabled by searching a larger solution space.

3.3 Research Limitation

The proposed model does not account for resource limitation. Schedules generated under unlimited resource constraints may be unrealistic because there are few cases resource is available at any time in construction period in the real world. Though this model is intended to present a possibility to explore decision space better than previous approaches, it is not practically applicable because of this limitation that has to be solved by a future research.

4. PROPOSED OPTIMIZATION MODEL

4.1 Objectives and Approach

The aim of this new optimization model is to enable a simultaneous optimization in terms of three important scheduling criteria which has not been achieved by similar approaches. The objectives are minimizing project duration, cost, and resource use variation. For the purpose of quantitative assessment, they were formularized as objective functions in the equation (4.1), (4.2), and (4.3).

$$Total\_Duration = \sum_{i=1}^{N} D_i \tag{4.1}$$

$$Additional\_Cost = \begin{pmatrix} d_1 & d_2 & d_3 & \ldots\ldots & d_n \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \cdot \\ \cdot \\ \cdot \\ c_n \end{pmatrix} \tag{4.2}$$

$$RLI = \sum_{k=1}^{m} (w_k \cdot l_k) \tag{4.3}$$

$$l_k = \sum_{q=1}^{T} \left(\sum_{all\_i} \left| r_{ik} - \bar{r}_k \right| \right) \tag{4.4}$$

$$\bar{r}_k = \sum_{all\_i}^{n} \left(r_{ik} \cdot (d_{io} - d_i) \right) / T \tag{4.5}$$

Total_Duration is an objective function which is sum of durations in the critical path.

Additional_Cost is an objective function that calculates additional project cost caused by crash.

RLI is an objective function that shows how the resource allocation of a project deviates from average resource use during the project duration.

d_{oi} is the original duration of activity i.

d_i is the durations reduced from original project activity durations.

d_{io} is the original duration of activity i.

c_i is the additional cost per one day of crash for activity i.

r_{ik} is a daily needed of resource k for activity i.

n is the total number of activities in the project.

m is the number of resources.

D_i is the durations of critical activities.

N is the number of activities on the critical path.

RLI is resource leveling index for multiple resource leveling.

In an effort to incorporate the three objective functions into a single phase of optimization and obtain a decent result, modifications were made in two ways:

- A new type of chromosome structure
- modified NPGA optimization process

4.2 Modified Niched Pareto Genetic Algorithm

Previous approaches need multiple chromosome forms [14, 15, 16]. Creation of those chromosomes are dependent on others; ordering chromosome is created based on data from activity duration chromosome, starting date chromosome needs scheduling and resource data generated by the cost – duration tradeoff in the earlier optimization stage. Because of this dependency between chromosomes, those chromosomes cannot be integrated in to a single structure that yields cost, duration, and resource leveling index at the same time. The discrete optimization stages make it impossible to fully explore the design space. Since the method follows the optimization process that finds out the near optimal time – cost tradeoff first and performs resource leveling on the Pareto front only in terms of cost and duration. This has a serious defect as an optimization process because it did not account for cost – RLI tradeoff and time – RLI tradeoff that may exist in the problem.

4.2.1 Chromosome Structure

In this section, a different kind of chromosome structure is proposed to construct an aggregate objective function (AOF). It is a basic approach in order to optimize multi-objective problems that all the objective functions are combined into a single form. However, previous approaches failed to create a single AOF due to the limitation in their chromosome structures. The absence of an AOF is the reason previous multi-objective scheduling optimization models could not perform simultaneous optimization. With

AOF, genetic algorithm can combine important attributes of scheduling such as project cost, duration, and evenly distributed resource allocation into a single process.

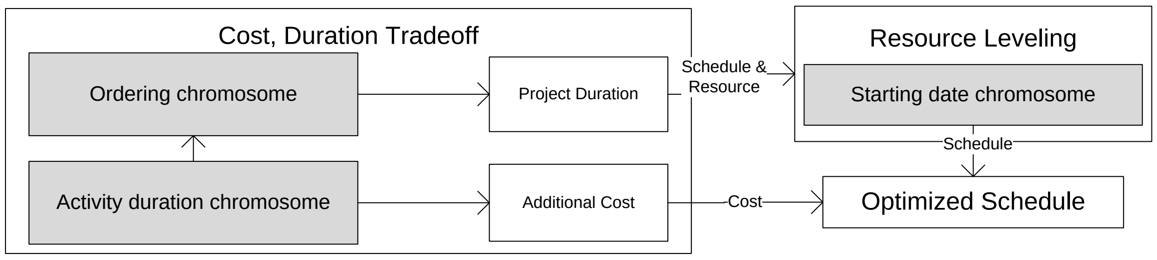


Figure 10 Schedules generation from chromosome structure of previous model

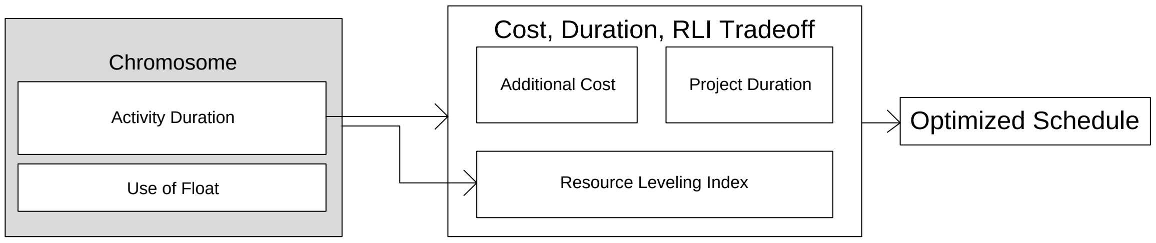


Figure 11 Schedules generation from chromosome structure of proposed model

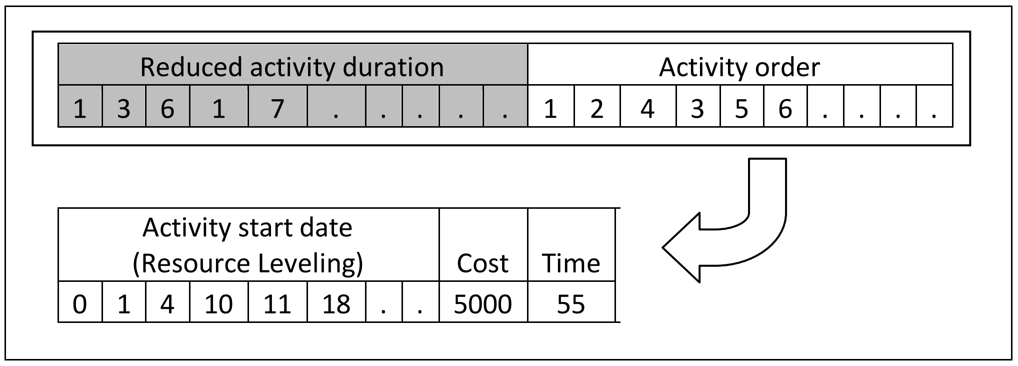


Figure 12 Chromosome representation of proposed model

The optimization procedure is compared to the proposed model of the same purpose in Figures 10 and 11. Dark boxes in the diagrams represent chromosomes. Since this research focuses on scheduling under unlimited resource leveling, 'ordering chromosome' can be ignored from the process in Figure 10 as its role is activity sequencing under the limited resource availability. The 'activity duration chromosome' defines how many days out of maximum reducible duration of each activity while the 'starting date chromosome' indicates when the activities start. Additional cost and total project duration are calculated from the reduced activity duration part, and activity start date part is used for calculating fluctuation in resource use [20]. Taking that into account, the chromosome structure of this approach is divided into two parts: cost-duration tradeoff phase and resource leveling phase while the proposed process has a single process with an integrated chromosome [16, 20].

The reason optimization process is divided into two phases as in the Figure 10 is that the 'starting date chromosome' for resource leveling process can be created only after the first optimization phase, cost-duration tradeoff, is completed. Due to this dependency, combinations of costs and durations do not change depending on how well resource distributed. Also, offspring chromosomes created by crossover or mutation operation have discrepancies between 'activity duration chromosome' and 'starting date chromosome' if we try to combine these three types into a single form. From this perspective, it can be seen that recent scheduling models still follow the scheduling approach of early Critical Path Method where resource leveling optimization takes place only after best time-cost tradeoff for a problem is found.

The drawback of this approach is its limited ability to search into the objective space. In Figure 7, time-cost tradeoff is found in XY axis and resource leveling takes place. Accordingly, tradeoffs between resource leveling and other objectives are not found by previous methods. However, ideal optimization for three-objective optimization should present tradeoffs between all the objectives as in Figure 9. This deficiency can be solved by proposing a genetic algorithm chromosome structured for direct generation of three fitness functions.

In order to enable simultaneous optimization showing all the existing tradeoffs, a chromosome structure is proposed in this thesis as can be seen in the Figure 12. From literature review section, it can be seen that, in general, different kinds of chromosome structure are applied for optimization for different objectives as following:

- Activity duration chromosome generates cost-duration tradeoff.
- Ordering chromosome decides the sequence of activities when resource is not available for multiple activities.
- Starting date chromosome performs resource leveling using available floats.

Unlike the previous model, 'reduced activity duration' part of the chromosome generates cost-duration tradeoff and both 'use of float' part and 'reduced activity duration' are integrated to calculate resource leveling index. The chromosome structure is composed of percentage values for 'use of float' instead of integer values to enable crossover operations between different solutions in the population thereby enabling simultaneous optimization of three objectives as seen in Figure 9. As the same percent values can be translated into different number of days within total float within different time-cost tradeoffs, the limitation that occurs when activity start days are used for resource leveling can be solved.

1	3	6	1	7	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Figure 13 Reduced activity duration chromosome

Chromosome of reduced durations in the Figure 13 enables time-cost tradeoff. Additional cost is calculated using formula 1.1, and conventional Critical Path Method is applied to calculate find critical path and activity floats. Then, for conventional model, another chromosome in the Figure 14 representing starting dates is created to decide how many days within float are to be used for resource leveling while the second part represents move of starting dates of activities within float by percentages instead of starting dates. In a traditional CPM analysis, activities are assumed to start on the fastest possible date. However, in a real construction project, non-critical activities can shift within float times in order to have even resource profiles [16].

By using a certain percentage for a float in the chromosome as in Figure 15, genetic mechanisms can be used without error and one chromosome can produce cost, duration, and resource leveling information at the same time.

0	5	13	21	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Figure 14 Starting date chromosome

Reduced activity duration										Use of float time									
1	3	6	1	7	.	.	.	.	.	20%	10%	90%	.	.	.	.	.	.	.

Figure 15 Proposed chromosome structure

4.2.2 Objective Space Normalization

The necessity to normalize objective space arises in order to find out well-distributed Pareto front for multi-objective problem. In general, weights are assigned to account for subjective preferences. But this is only applicable to the case of the comparability of all components.

By conducting objective space normalization, 0 and 1 are assigned respectively to the minimum and maximum fitness values of each objective. Therefore, the search space for this problem will become a cube as a result. Finding optimal solutions in the normalized space is greatly beneficial in performing NPGA's sharing function that preserves solution diversity. Furthermore, doing so strengthens visibility of relationship between conflicting objectives. The advantages of 0 and 1 instead of its real values can be seen in Figures 16 and 17. Taking into account that sharing function of NPGA essential in the optimization process, the objective space in Figures 17 has an advantage over the space in the Figures 16. In the space with real scale, it is very difficult to set a sharing distance because of different scales of objectives, and in that case sharing function does not work effectively. Since sharing functions in the algorithm in a way that distributes solutions by dividing search space into spaces of the same size, sharing may not occur well for all the objectives in a space with different scales. However, radius setting becomes easier and sharing function works well in the normalized space in Figure 17. Thus, in the proposed model, objective space is normalized before optimization process starts in order to define an appropriate sharing radius.

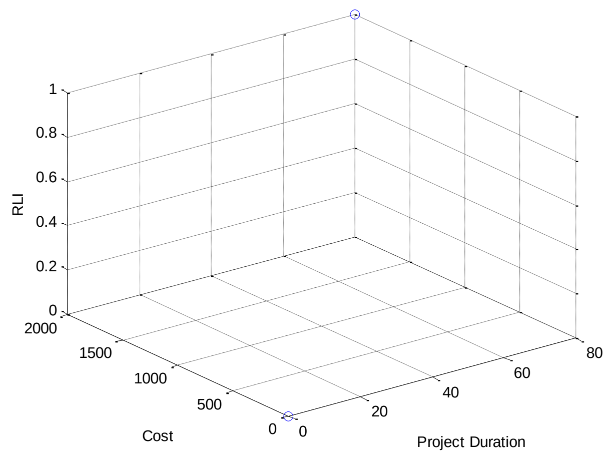


Figure 16 Original search space

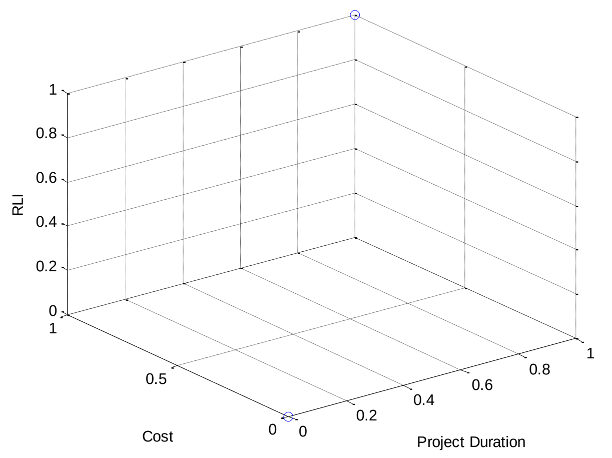


Figure 17 Normalized search space

The method of normalizing objective space is adopted from Normalized Normal Constraint (NNC) Method [17, 18]. This method is in line with efforts to develop a method of generating uniformly distributed Pareto front using Genetic Algorithms. It is an enumeration of single objective optimization with constraints occurring in the normalized objective space. From seven steps in the NNC method explained by the pseudo code in Figure 18, only the first two steps for space normalization method is selected because the proposed optimization model does not use single objective method as NNC.

```
Step -1: Anchor Points. Obtain anchor points
Step -2: Objective Mapping/Normalization
Step -3: Utopia Line Vector
Step -4: Normalized Increments
Step -5: Generate Utopia Line Points
Step -6: Pareto Points Generation
Step -7: Pareto Design Metrics Values
```

Figure 18 NNC method process

The theoretical introduction to the procedure focuses on bi-objective problem for simplicity in explanation. The mathematical representation is shown below:

$$\min_x \left[\mu_1(x)\mu_2(x)\right] \tag{4.6}$$

subject to:

$$\begin{gathered} g_q(x) \le 0, (1 \le q \le r) \\ h_k(x) = 0, (1 \le k \le s) \\ x_{li} \le x_i \le x_{ui,} (1 \le i \le n_x) \end{gathered} \tag{4.7}$$

where x represents the dimension vector of variables

$g_q(x)$ is the r inequality constraints.

$h_k(x)$ is the s equality constraints.

x_{li}, x_{ui} are the lower and upper limitation constraints in the dimension i.

Then, essential elements in Figure 19 are:

- Anchor points (μ^{i*}) are obtained by minimizing each objective independently. These points are deemed as both ends of the Pareto front.
- Utopia point (μ^u) is a point with components that are the optimum values of anchor points.

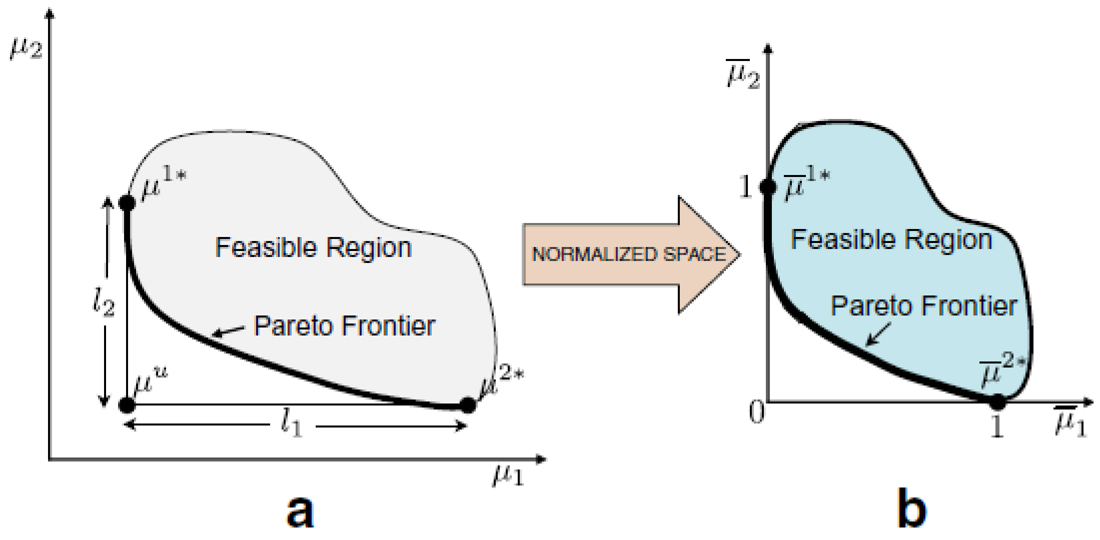


Figure 19 Space normalization for bi-objective optimization (Martínez et al., 2009)

The normalization steps that transform the space from a to b in Figure 19 are:

Step 1: Anchor Points (μ^{1*}, μ^{2*}, μ^{3*})

By minimizing each objective individually, obtain three anchor points since this is a problem with three objectives.

$$Min_x \mu_i(x)(i = 1, 2, 3) \tag{4.8}$$

These points are end points of the Pareto front, and Utopian point is defined by the optimized points of anchor points.

μ_1 = project duration

μ_2 = project cost

μ_3 = resource leveling index (RLI)

Step 2: Objective Mapping/Normalization

Optimization takes place in the normalized space in order to avoid scaling deficiencies. $\overline{\mu}$ is the normalized form of μ. L is defined as the maximum distance of each objective component. In this objective mapping of three-objective space, both the Utopia point (μ^u) and Nadir point (μ^N) are obtained as in equations (4.9) and (4.10):

$$\mu^u = \left[\mu_1(x^{1*}), \mu_2(x^{2*}), \mu_3(x^{3*})\right]^T, \tag{4.9}$$

where

μ^{1*} = optimum project duration

μ^{2*} = optimum project cost

μ^{3*} = optimum resource leveling index (RLI)

$$\mu^N = \left[\mu_1^N, \mu_2^N, \mu_3^N\right]^T, \tag{4.10}$$

where

$$\mu_i^N = \max\left[\mu_i(x^{1*}), \mu_i(x^{2*}), \mu_i(x^{3*})\right] \tag{4.11}$$

$$i \in \{1, 2, ..., n\}.$$

Maximum distances L are defined as:

$$L = \begin{Bmatrix} l_1 \\ l_2 \\ l_3 \end{Bmatrix} = \mu^N - \mu^u, \tag{4.12}$$

which normalizes the metrics as:

$$\overline{\mu_i} = \frac{\mu_i - \mu_i(x^{i*})}{l_i}, i = 1, 2, 3. \tag{4.13}$$

4.2.3 External Archive

The proposed optimization model incorporates an archive that copies best solutions found from each generation. From each generation run, non-dominated solutions are copied from the population to the external archive. Since individual solutions in this external archive can avoid generation cycles of genetic algorithm, it is not under the influence generation operators (crossover and mutation operators). Thus, optimal solutions can be preserved by avoiding the random effects. However, a part of Pareto optimal solutions from the external archive are also copied and sent to the mating pool to facilitate generation of better solutions. In the last run of optimization runs, the external archive is incorporated into the final population. Then, again, non-dominated Pareto solutions are selected from the individuals of both external archive and the final population by using the selection method of non-dominated sorting.

4.2.4 Optimization Process

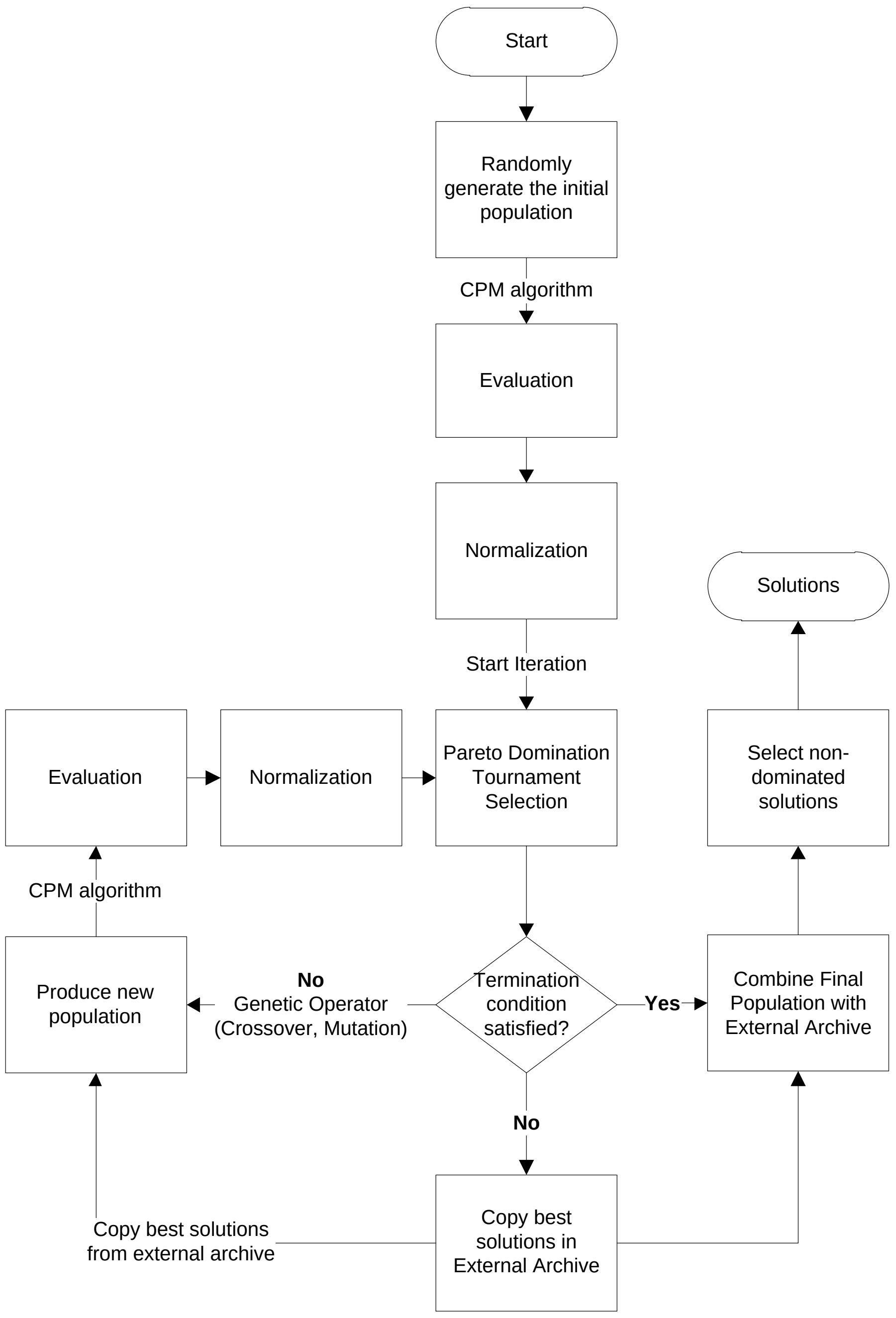

Start
Randomly generate the initial population
CPM algorithm
Evaluation
Normalization
Start Iteration
Solutions
Evaluation
Normalization
Pareto Domination Tournament Selection
Select non-dominated solutions
CPM algorithm
Produce new population
No
Genetic Operator (Crossover, Mutation)
Termination condition satisfied?
Yes
Combine Final Population with External Archive
No
Copy best solutions from external archive
Copy best solutions in External Archive

Figure 20 Optimization process of proposed model

Original NPGA optimization procedure was modified to increase efficiency of sharing function and to prevent optimal solutions from disappearing due random effects of genetic operations by using external archive. First of all, objective space is normalized to overcome scaling deficiency. External archive is adopted in the optimization process to preserve optimal solutions that may disappear due to random characteristic of genetic operators. External archive contributes to search toward Pareto front as the best solutions from the archive compose certain part of population for next generation.

The first step in Figure 20 is random generation of initial population. Chromosomes of predefined number are generated and each of them has reduced activity duration part and use of float time part. When there are X number of activities in the project, each chromosome is composed of X×2 number of decision elements. This information is sent to CPM engine. Then, the CPM engine calculates project cost and duration using information from reduced activity duration part and resource leveling index is calculated using both parts of chromosome. Since float use is represented by certain percentages instead of days, resource leveling index can be calculated using data in a chromosome without additional method. Therefore, it is possible to obtain three fitness values from each individual chromosome. Iterations start after the first population is generated, evaluated and normalized. Before the Pareto domination tournament selection stage, all the solutions are ranked. Then, elite solutions sorted by the tournament selection are sent to the pool of next generation. Remaining solutions are combined into the pool after they are transformed by crossover and mutation operators. In each generation, best solutions are selected by non-dominated sorting method and sent to the external archive. When termination condition is met, iteration stops and the final population are combined with solutions in the external archive. Pareto optimal solutions are found from the combined solutions by non-dominated sorting method.

5. CASE STUDY AND ANALYSIS OF FINDINGS

Before the model is applied to the construction schedule data sets, its performance is demonstrated using a pilot schedule with 30 activities. Its purpose is to show how much the solutions in the initial population improve in the final population and to show optimal solutions are preserved by the external archive. Figure 21 shows the time-cost tradeoff of the pilot project. Red stars in the figure represent the Pareto optimal solutions obtained from the external archive. Blue and red circles are the solutions in the initial and final population, respectively. From the result, it can be seen that the final population has better solutions than the initial population and the external archive prevents loss of optimal solutions from random effect of genetic operators.

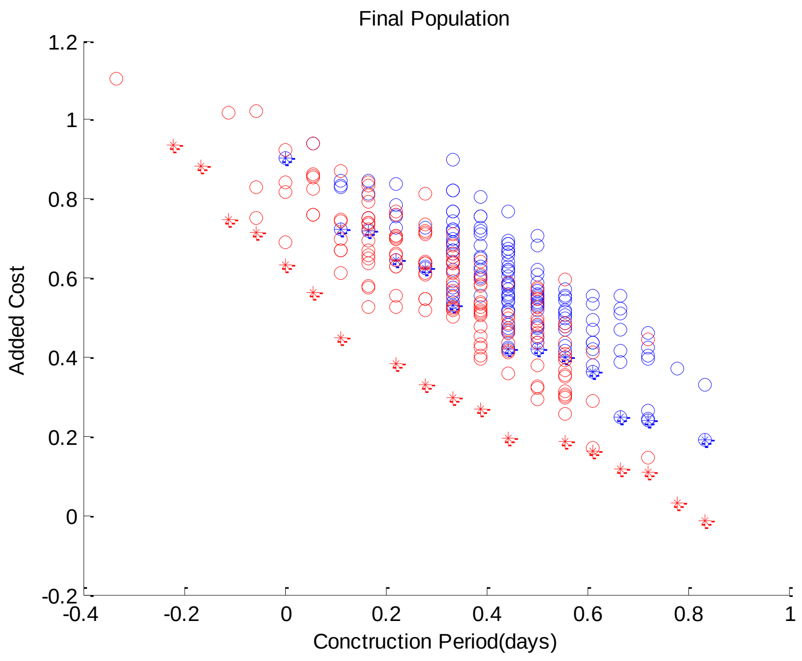


Figure 21 Cost-duration tradeoff found by proposed model

Then, the algorithm was applied to two cases. Case 1 is a scheduling of 11 activities with two objectives: minimum project cost and duration. Case 2 is a 9-activity scheduling optimization with three objectives: minimum cost, duration and resource leveling index. The activity CPM networks are taken from a similar research carried out to solve time-cost tradeoff problem and three-objective optimization problem, respectively [14].

5.1 Case 1: Bi-objective Scheduling with 11 Activities

This case study is to demonstrate the performance of this algorithm on time-cost tradeoff problem. Activity network is shown in Figure 22.

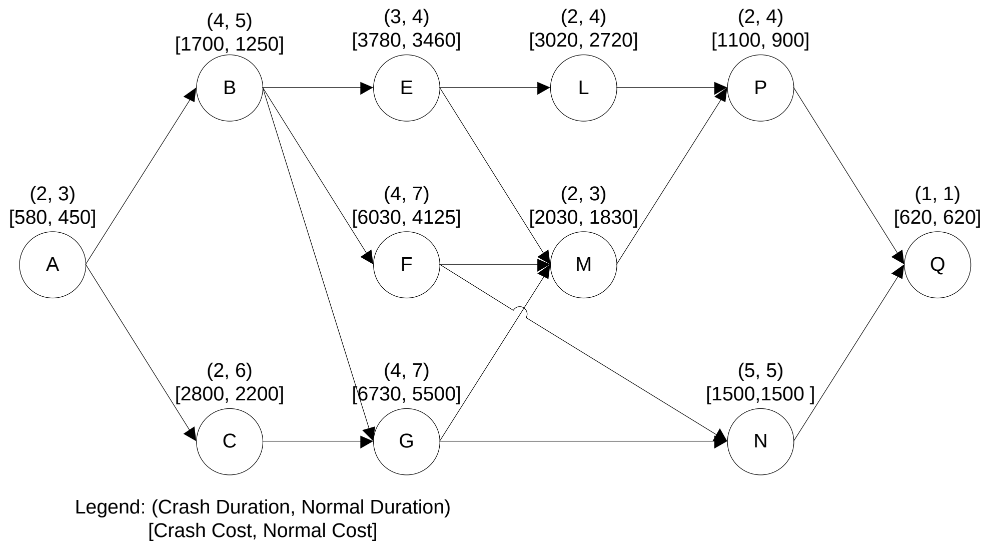


Figure 22 11 activity CPM network

There are 11 activities for this construction scheduling problem. Activity A should start first; other activities follow according to the activity relationship and activity

Q is the last activity. Related to determining the duration of each activity, a decision can be made as days between normal duration and crash duration. And, the activity cost increases when the activity duration decreases. For example of activity G which has normal duration of 7 and crash duration of 4, duration options are 4, 5, 6, and 7. The activity cost will become 6,730 instead of the normal cost (5,500) if the activity duration is crashed into 4 days.

Optimal solutions were generated by the proposed algorithm considering this conflicting relationship between cost and duration. In Figure 23, the red stars are Pareto frontier points generated by the proposed algorithm and are compared to the green circles obtained by random generation of 100,000 solutions. For each duration option, the algorithm tried to generate a solution with minimum cost. And, the relationship between the two objectives was revealed where the project costs tend to increase as the project durations decrease. The near-optimal solutions were comparatively better than the randomly generated solutions.

However, this algorithm could not overcome the disadvantage of heuristic method that generation of real optimum solutions is not guaranteed. It often could not generate all the existing optimal solutions and Figure 23 shows the result from one optimization where only partial optimal solutions were obtained. The algorithm generated 8 optimal solutions while the actual number of optimal solutions is 9 according to Leu, et al. [16].

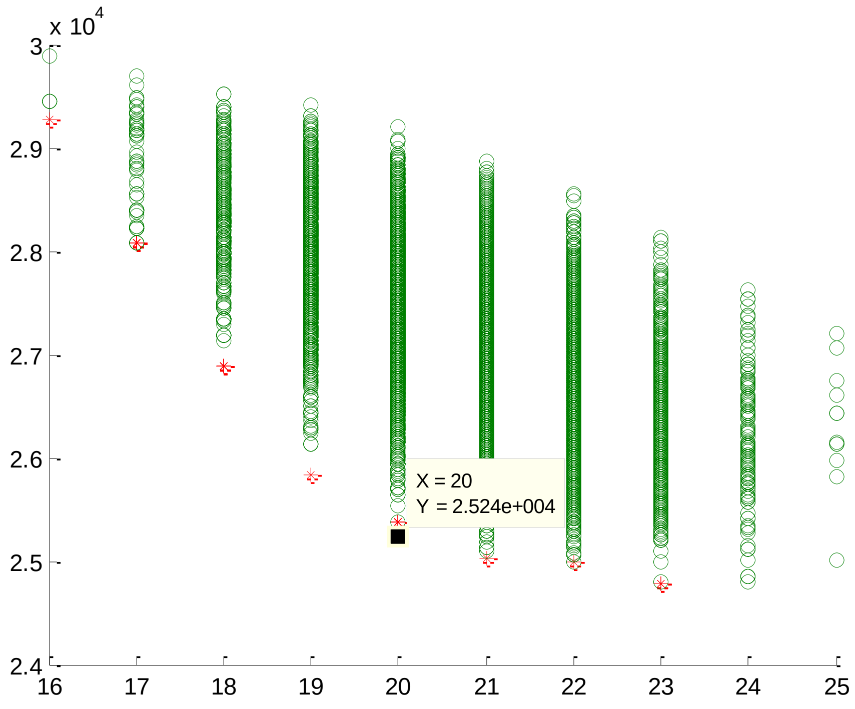


Figure 23 Time-cost tradeoff for case 1

5.2 Case 2: Three-objective Scheduling with 9 Activities

The case study 2 is to demonstrate the proposed optimization model on a three-objective scheduling problem. Thus, in addition to project cost and duration minimization, resource leveling index minimization was integrated into the optimization process. Figure 24 describes the activity precedence relations with normal duration, crash duration, normal cost and crash cost for each activity. In Table 1, activity cost and daily resource use settings for each duration option for each activity are presented.

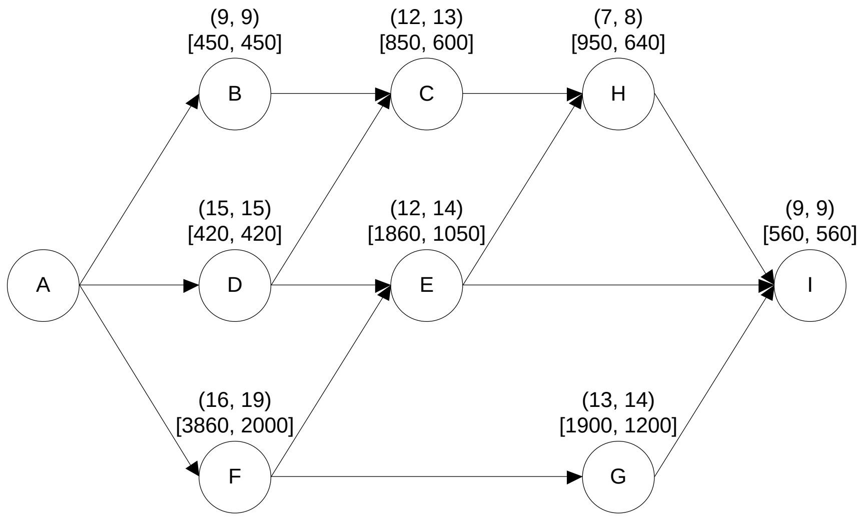


Figure 24 9-activity CPM network

Table 1 Optimal solutions generated by 200 runs

Activity	Duration	Cost	Resource 1	Resource 2	Resource 3
A	5	480	5	4	5
	6	300	3	4	5
B	9	450	4	5	2
C	12	850	4	6	6
	13	600	3	6	5
D	15	420	5	2	4
E	12	1860	1	5	6
	13	1450	1	5	4
	14	1050	1	5	2
F	16	3860	6	4	4
	17	3220	5	3	3
	18	2600	4	2	2
	19	2000	3	1	1
G	13	1900	3	3	6
	14	1200	3	2	5
H	7	950	6	4	3
	8	640	6	3	2
I	9	560	5	5	5

Like the case study 1, project cost and project duration have conflicting relation where one increases when the other decreases. Also, the amount of the resource used for each activity increases when an attempt is made to reduce the activity duration. Thus, it can be seen that activity crash contributes to increase in resource use peak. And, there is a higher chance of obtaining better resource leveling index when less number of activity durations are reduced by fewer days.

The preferences of minimizing the three objective values were pre-integrated into the model, and 30 solutions were obtained as a result of 100 generations. During the optimization process, hyper volume was calculated for each generation. Hyper volume is the volume of the space in the search space that is dominated by the Pareto optimal solutions. This presents a quantified measurement of improvements of best solutions as seen in the first figure on page 54. Although the model was run for 100 times, the algorithm started to generate the same or similar solutions from 80th generation. Because of high chances of mutation and crossover, high variability was observed until the generation approached 80 when the changes decreased gradually and stayed almost the same.

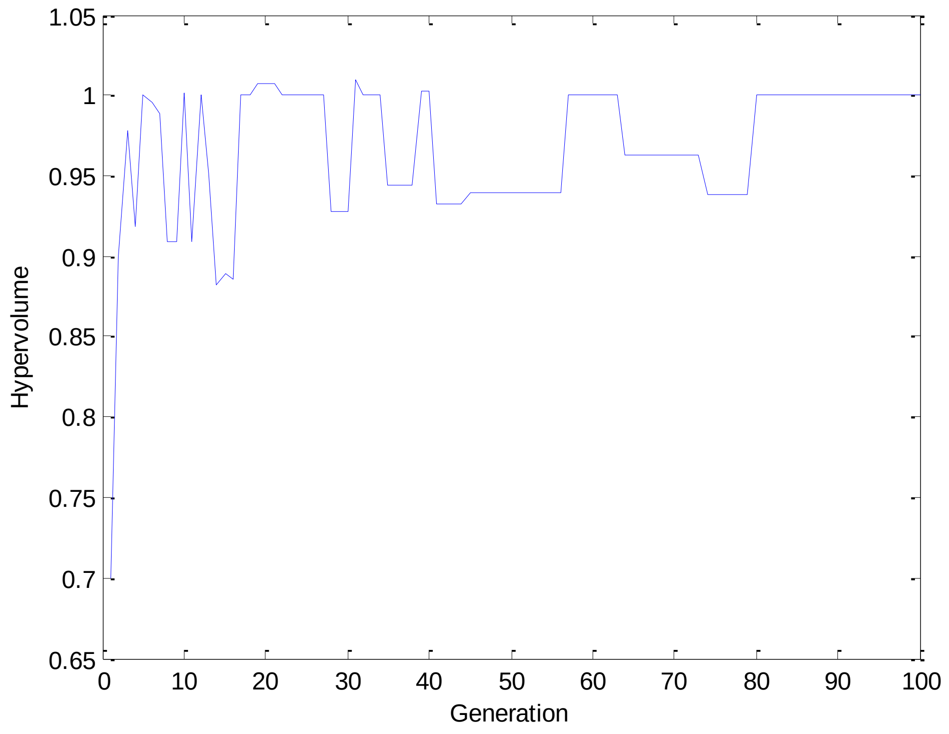


Figure 25 Hypervolume by generation runs

In Appendix B, actual solution data on chromosome, duration, cost and resource leveling index are attached. The solutions are visualized in the 3-dimension space as in Figure 25. The three axes denote project cost, project duration, and resource leveling index of solutions.

Figure 26 demonstrates time-cost tradeoff for this scheduling problem. Figures 27 and 28 show duration-resource leveling tradeoff and cost-resource leveling tradeoff, respectively. In general, project cost and project duration are conflicting since project cost tends to become greater when the project duration becomes shorter by using reduced activity durations instead of normal activity durations. Also, better resource leveling index is obtained when project duration becomes longer. However, seen from the Figure 29, resource leveling index becomes smaller (better resource leveling performed) when less total project cost is needed. This cost-resource leveling tradeoff is non-conflicting, indirect and can be interpreted by the relationship between project duration and resource leveling index. There is no direct relation between project cost and resource leveling in the problem. Resource leveling index is negatively influenced by the project duration, and activity durations determine activity costs too. Since the project duration and cost are in conflicting relation, both project cost and resource leveling index increase when project duration decreases. In Figure 30, resource allocation of a solution in the final population is shown.

The distribution of solutions in the cost-duration tradeoff in Figure 26 differs from that of the bi-objective optimization in case 1. In case 1, only one solution exists for one construction duration. However, multiple solutions were found for one construction duration or cost. Wider range of solution was obtained by integrating resource leveling index in the optimization. Even inferior solutions in terms of cost and duration tradeoff could be selected when resource leveling index was superior. In Appendix A, detailed data on optimal solutions are given. The data includes chromosome, objective values, and resource allocation of the solutions.

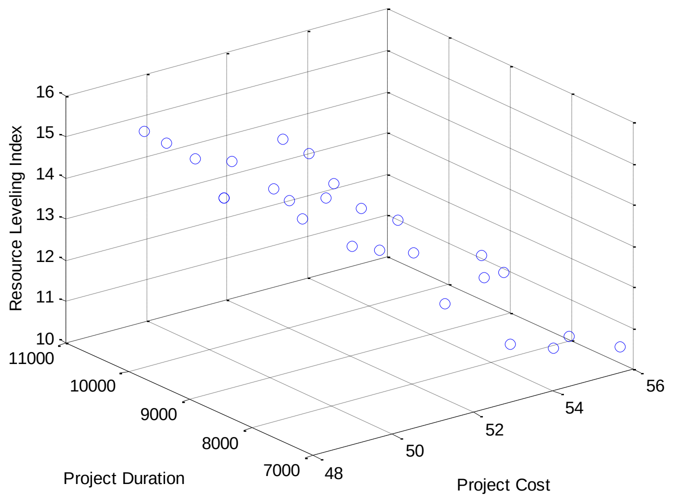


Figure 26 Optimum solutions in 3-dimension

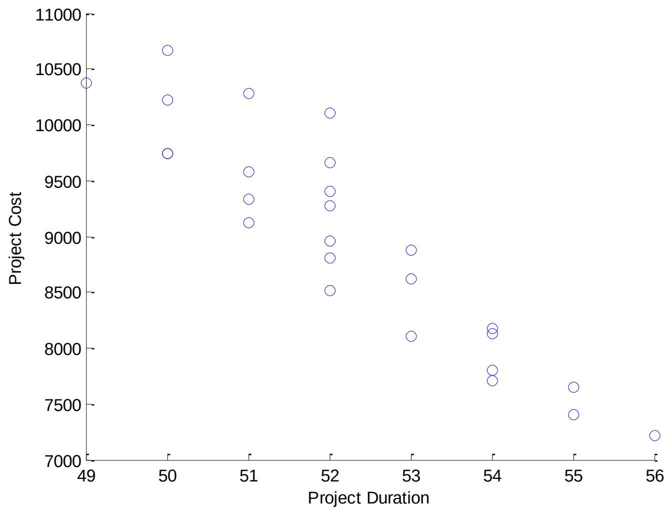


Figure 27 Cost-duration tradeoff

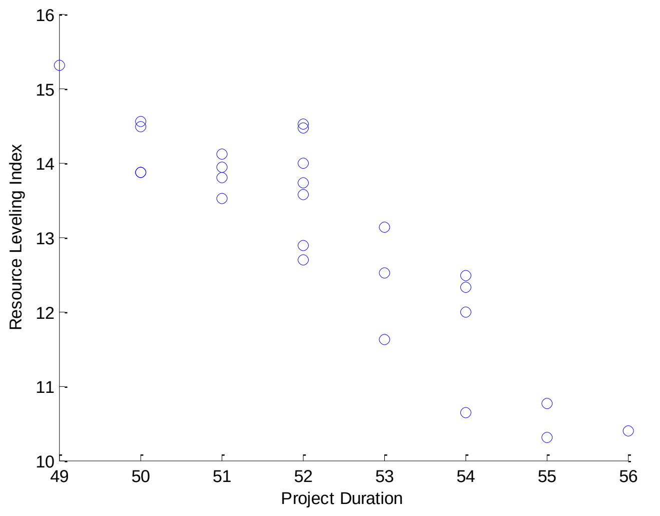


Figure 28 Duration-leveling tradeoff

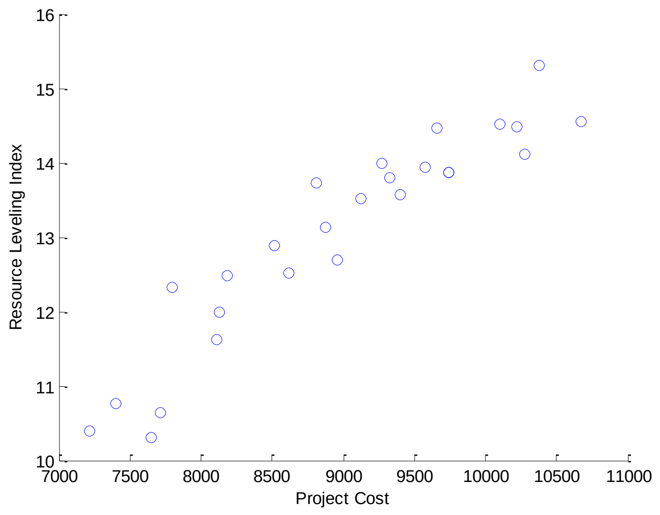


Figure 29 Cost-leveling tradeoff

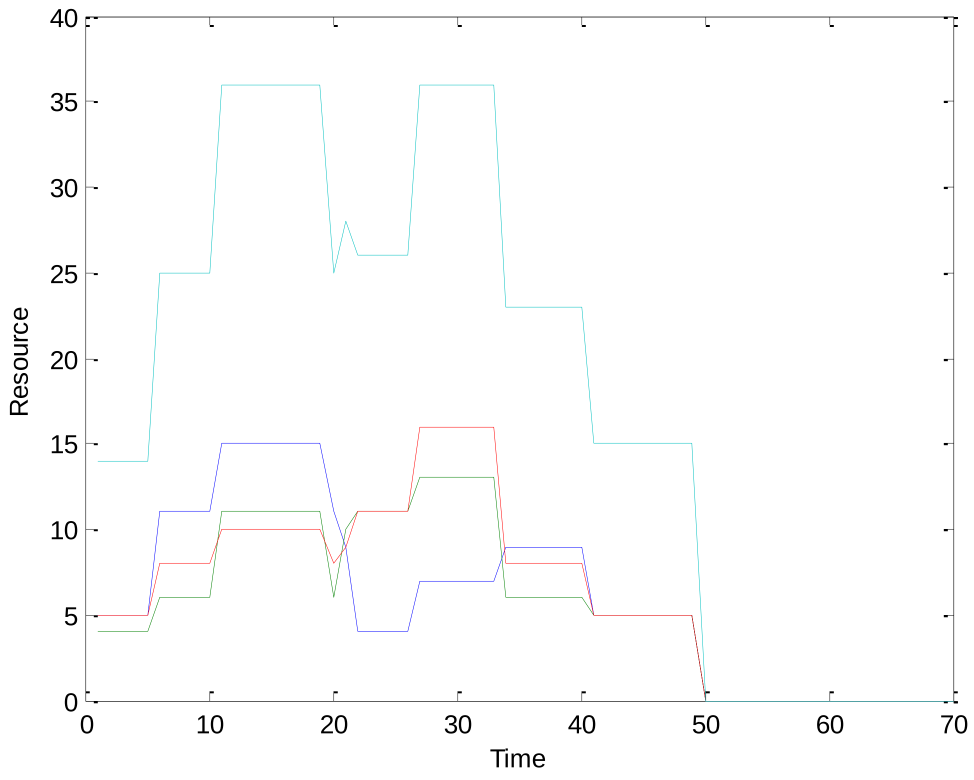


Figure 30 Resource allocation (blue: resource 1, green: resource2, red: resource 3)

6. SUMMARY AND CONCLUSION

The Critical Path Method (CPM) was invented to achieve a higher level of scheduling coordination and has been in wide use for several decades. However, it is becoming obsolete because of limitations such as arithmetic complexity. Because using CPM for large construction projects requires excessive computational efforts, many methods have been developed based on traditional CPM. Most of these efforts were to develop heuristic scheduling methods because of its simple format effectively overcomes the computational complexity. Also, heuristic methods are usable for scheduling large and complex construction projects.

Like many heuristic methods, the optimization model in this thesis attempts to solve a complex scheduling problem with multiple objectives. The proposed model goes a step further and attempts to achieve a simultaneous optimization in terms of three objectives, whereas previous approaches used sequential optimization processes. Expected benefits by enabling a simultaneous optimization include more thorough search space exploration, wider range of optimal solutions, and better performance of a heuristic method as a decision support tool for complex construction scheduling problems. Thus, in the proposed scheduling model presented in this thesis, a new data structure was developed to enable an integrated optimization process. Also, a search space normalization method and external archive were used to avoid scaling deficiencies and to prevent solutions from disappearing.

Since every objective was integrated when solutions were generated, the model was able to find all existing tradeoffs between the three objectives in the scheduling problems. Thus, compared to similar attempts, the proposed model provided more thorough information; wider insight into the scheduling problem, and clear consequences of the solutions as illustrated in Figure 31. Alternative solutions generated by the existing models were very limited when compared to the proposed model. Since there is only one optimal solution for one project duration using traditional methods, a decision maker has to select and modify it before actually applying it.

However, the proposed model provides multiple options that are optimal. For example, when the project has to be 51 days as shown in Figure 31, there are five alternative solutions. Within the five candidate solutions provided by the decision supporting tool, a decision maker can implement the optimal schedule according to his or her judgment between cost and resource use. This is a very important point for a decision maker since a decision can be made among alternative solutions that are non-inferior to any other solution.

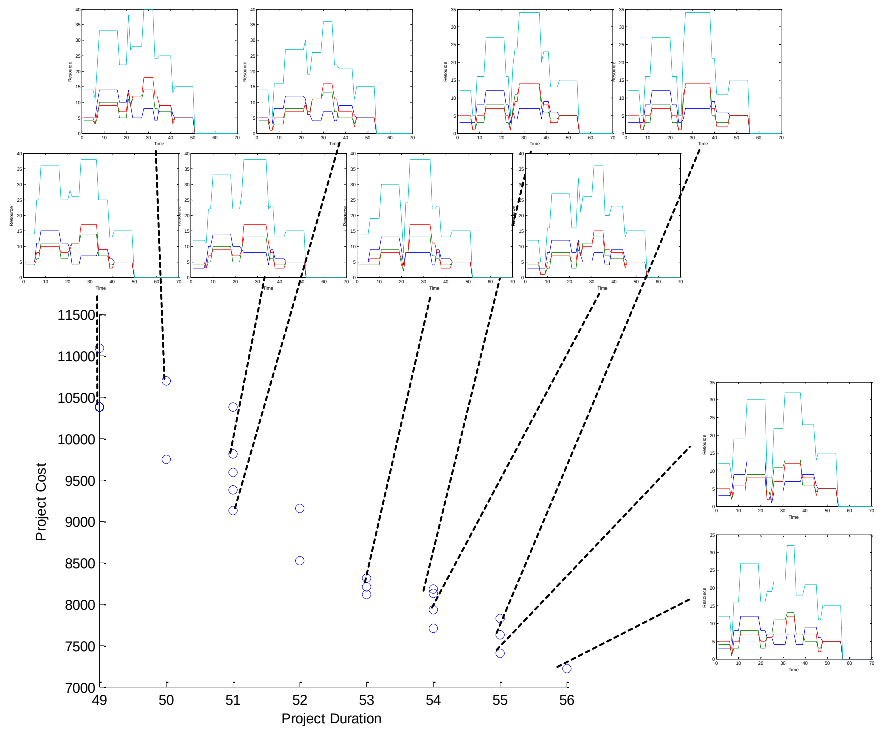


Figure 31 Demonstration of cost-duration tradeoff and obtainable resource leveling

7. FUTURE RESEARCH

The algorithm associated with this research was developed to propose a methodology to generate multiple scheduling options considering objectives equally important. To achieve this however, the model generates schedules assuming there are unlimited resources available. Applied to real-world construction projects, this can generate schedules that are unrealistic. Therefore, taking into account that any construction scheduling has to be done under limited resource availability, the scheduling model in this thesis needs further development to represent real world scheduling problems. Resource-constrained scheduling optimization may be realized in future research by using the chromosome structure shown in Figure 32. In addition to reduced activity duration and the use of float, ordering chromosomes will be integrated into the model. Ordering the chromosome's function will determine the order of activities under limited resources. As such, the expectation is the generation of more realistic schedules.

Reduced Activity Duration						Ordering						Use of Float					
2	1	3	4	2	1	1	3	2	4	5	6	10%	20%	5%	90%	15%	5%

Figure 32 Proposed chromosome structure for future research

In addition, more diverse activity relationships should be used in the model. This research uses the dominate Finish-to-Start relationship between construction activities. In typical construction schedules, there are also Start-to-Start, Start-to-Finish, and Finish-to-Finish relations. By including these relationships, real-world construction scheduling problems can be optimized by the proposed modeling approach.

REFERENCES

1. Allam SI (1988) Multi-Project Scheduling: A New Categorization for Heuristic Scheduling Rules in Construction Scheduling Problems. Constr Manag and Econ 6(2): 93-115

2. Cohon JL (2003) Multi-objective Programming and Planning. Dover Publications, Mineola, NY

3. Deb K, Goldberg DE (1989) An Investigation of Niche and Species Formation in Genetic Function Optimization. Proc of the 3rd Int Conf on Gen Algo Morgan Kaufmann Publishers Inc: 42-50

4. Easa S (1989) Resource Leveling in Construction by Optimization. J Constr Eng and Manag 115(2): 302-16

5. F Märki, Fischer M, Kunz J, Haymaker J (2007) Decision Making for Schedule Optimization. Tech Rep of Ctr for Integ Fac Eng (169): 1-23

6. Feng, Chung-Wei (1997) Using Genetic Algorithms to Solve Construction Time-Cost Trade-Off Problems. J Comp in Civil Eng 11(3): 184-89

7. Goldberg DE, Miller B (1995) Genetic Algorithms, Tournament Selection, and the Effects of Noise. Comp Syst 9(3): 493-212

8. Gray CF (1981) Essential of Project Management. Petrocelli Books, UK

9. Harris H (1990) Packing Method for Resource Leveling. J Constr Eng and Manag 116(2): 331-350

10. Hartmann S (1999) Project Scheduling under Limited Resources. Springer, Berlin

11. Jie Y, Kharma N, Grogono P (2010) Bi-Objective Multipopulation Genetic Algorithm for Multimodal Function Optimization. IEEE Trans Evol Comput 14(1): 80-102

12. Kelley JE, Walker M R (1959) Critical-Path Planning and Scheduling. Papers presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Comp Conf ACM, Boston, MA

13. Kurtulus I, Davis EW (1982) Multi-Project Scheduling: Categorization of Heuristic Rules Performance. Manag Sci 28(2): 161-72

14. Leu Sou-Sen, Yang C (1999) Ga-Based Multicriteria Optimal Model for Construction Scheduling. J Constr Eng and Manag 125(6): 420-27

15. Leu Sou-Sen, Yang C (1999) A Genetic-Algorithm-Based Resource-Constrained Construction Scheduling System. Constr Manag and Econ 17(6): 767-76

16. Leu Sou-Sen, Yang C, Huang J (2000) Resource Leveling in Construction by Genetic Algorithm-Based Optimization and Its Decision Support System Application. Auto in Constr 10(1): 27-41

17. Martínez M, et al. (2009) Genetic Algorithms Optimization for Normalized Normal Constraint Method under Pareto Construction. Adv in Eng Softw 40(4): 260-67

18. Messac A, Ismail-Yahaya A, Mattson C (2003) The Normalized Normal Constraint Method for Generating the Pareto Frontier. Struct and Multi Optim 25(2): 86-98

19. Senouci A, Al-Derham H (2008) Genetic Algorithm-Based Multi-Objective Model for Scheduling of Linear Construction Projects. Adv in Eng Softw 39(12): 1023-28

20. Senouci A, Eldin N (2004) Use of Genetic Algorithms in Resource Scheduling of Construction Projects. J Constr Eng and Manag 130(6): 869-77

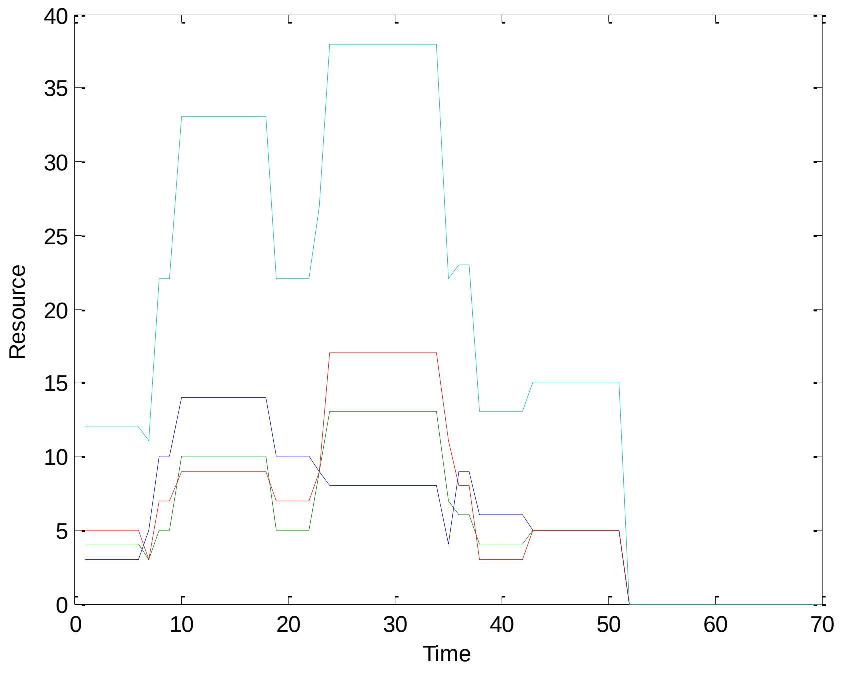
21. Singh G (2006) Comparison of Multi-Modal Optimization Algorithms Based on Evolutionary Algorithms. Proc of the 8th Ann Conf on Gene and Evol Comp: 1305-12

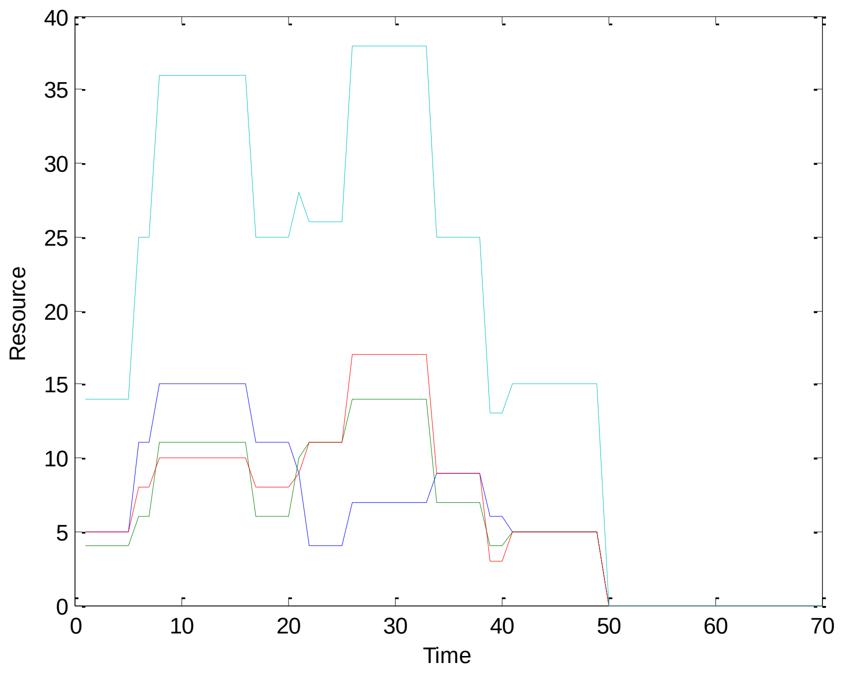
APPENDIX A

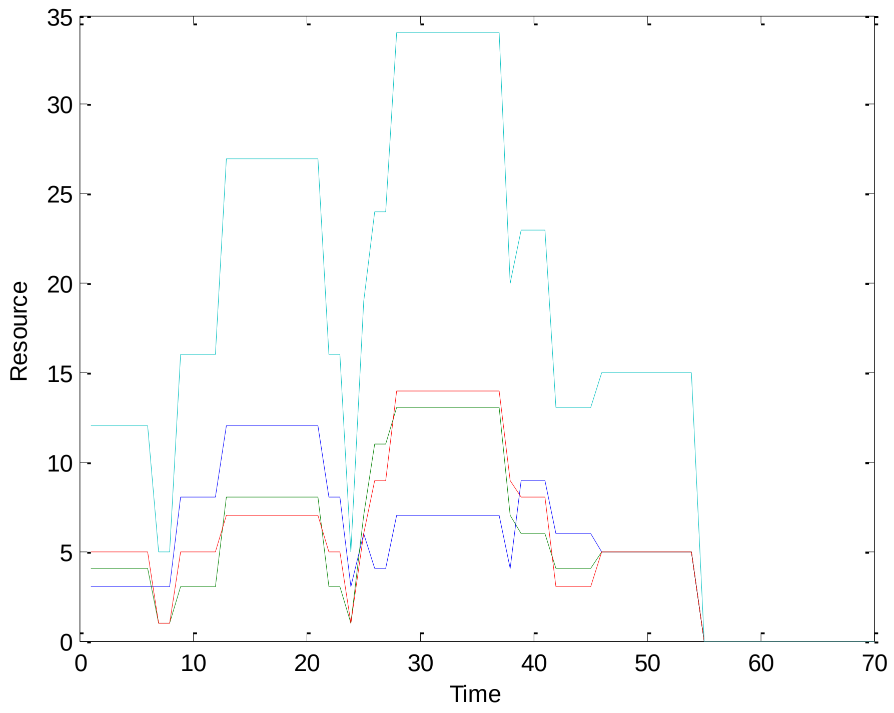
solution	Chromosome																		Result		
	Reduced Activity Duration									Use of Float									Duration	Cost	RLI
1	0	2	0	0	2	1	0	1	0	0.72	0.84	0.56	0.33	0.91	0.73	0.05	0.92	0.02	51	9810	14.85
2	1	3	0	0	2	0	1	1	0	0.78	0.75	0.10	0.39	0.85	0.49	0.59	0.64	0.83	49	11080	15.36
3	0	0	0	0	1	0	0	1	0	0.01	0.19	0.58	0.58	0.49	0.80	0.41	0.54	0.51	54	7930	12.62
4	1	1	0	0	2	1	0	1	0	0.42	0.97	0.13	0.56	0.42	0.22	0.00	0.47	0.93	51	9370	14.11
5	1	0	0	0	2	0	0	0	0	0.52	0.84	0.72	0.74	0.10	0.34	0.87	0.71	0.33	53	8210	12.45
6	1	2	0	0	2	1	1	1	0	0.33	0.93	0.69	0.28	0.80	0.24	0.82	0.26	0.48	50	10690	14.50
7	1	0	0	0	0	0	0	1	0	0.45	0.72	0.87	0.74	0.88	0.73	0.53	0.89	0.55	54	7710	11.99
8	0	0	0	0	1	1	0	1	0	0.38	0.02	0.81	0.47	0.46	0.37	0.90	0.33	0.57	54	8180	12.22
9	0	0	0	0	1	0	0	0	0	0.31	0.88	0.41	0.46	0.11	0.91	0.08	0.35	0.28	55	7620	12.94
10	0	1	0	0	0	0	0	1	0	0.22	0.23	0.47	0.69	0.17	0.97	0.80	0.49	0.97	54	8130	12.29
11	1	1	0	0	2	0	0	1	0	0.80	0.88	0.68	0.75	0.94	0.39	0.39	0.72	0.14	51	9120	13.66
12	1	2	0	0	1	1	0	1	0	0.65	0.98	0.28	0.69	0.99	0.23	0.96	0.08	0.86	51	9580	13.32
13	1	1	0	0	0	0	0	1	0	0.30	0.93	0.61	0.50	0.84	0.95	0.49	0.29	0.35	53	8310	12.68
14	1	2	0	0	2	0	0	1	0	0.32	0.73	0.58	0.17	0.06	0.88	0.88	0.43	0.89	50	9740	14.19
15	1	2	0	0	2	1	1	0	0	0.50	0.19	0.83	0.27	0.28	0.75	0.80	0.51	0.56	51	10380	14.34
16	1	0	0	0	1	0	0	1	0	0.97	0.90	0.52	0.50	0.43	0.30	0.77	0.07	0.35	53	8110	11.94
17	1	0	0	0	2	0	0	1	0	0.60	0.86	0.02	0.39	0.80	0.18	0.57	0.95	0.33	52	8520	12.77
18	0	2	0	0	1	0	0	1	0	0.75	0.93	0.76	0.34	0.61	0.19	0.82	0.49	0.89	52	9150	13.44
19	0	1	0	0	0	0	0	0	0	0.22	0.85	0.40	0.24	0.49	0.86	0.43	0.90	0.57	55	7820	12.35
20	1	3	0	0	2	0	0	1	0	0.61	0.78	0.46	0.42	0.57	0.98	0.96	0.09	0.94	49	10380	14.77
21	1	3	0	0	2	0	0	1	0	0.23	0.74	0.18	0.67	0.69	0.30	0.93	0.38	0.30	49	10380	14.77

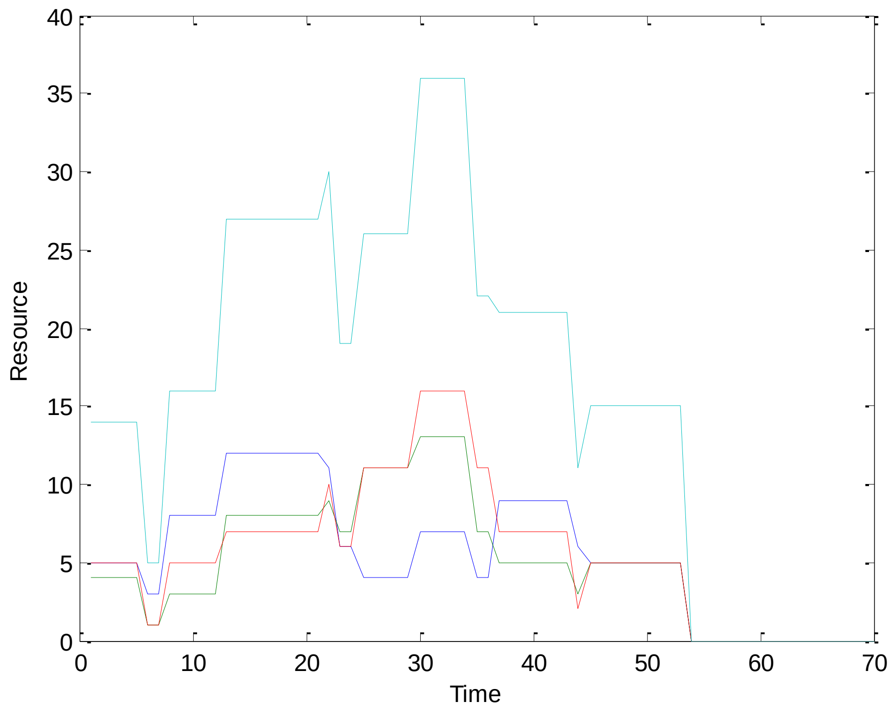
solution	Chromosome																		Result		
	Reduced Activity Duration									Use of Float									Duration	Cost	RLI
22	1	3	0	0	2	0	0	1	0	0.81	0.42	0.59	0.38	0.80	0.42	0.97	0.75	0.77	49	10380	14.77
23	1	3	0	0	2	0	0	1	0	0.46	0.92	0.93	0.91	0.16	0.94	0.98	0.13	0.84	49	10380	14.77
24	1	3	0	0	2	0	0	1	0	0.79	0.31	0.69	0.03	0.47	0.80	0.96	0.58	0.99	49	10380	14.77
25	1	3	0	0	2	0	0	1	0	0.36	0.44	0.06	0.33	0.61	0.39	0.95	0.58	0.44	49	10380	14.77
26	1	3	0	0	2	0	0	1	0	0.64	0.66	0.86	0.39	0.92	0.39	0.92	0.51	0.93	49	10380	14.77
27	1	0	0	0	0	0	0	0	0	0.43	0.35	0.78	0.03	0.48	0.40	0.99	0.27	0.44	55	7400	10.53
28	0	0	0	0	0	0	0	0	0	0.53	0.78	0.16	0.38	0.28	0.20	0.80	0.95	0.88	56	7220	10.71
29	1	3	0	0	2	0	0	1	0	0.62	0.78	0.64	0.16	0.09	0.63	0.93	0.24	0.19	49	10380	14.77
30	1	3	0	0	2	0	0	1	0	0.26	0.70	0.80	1.00	0.99	0.25	0.93	0.68	0.10	49	10380	14.77

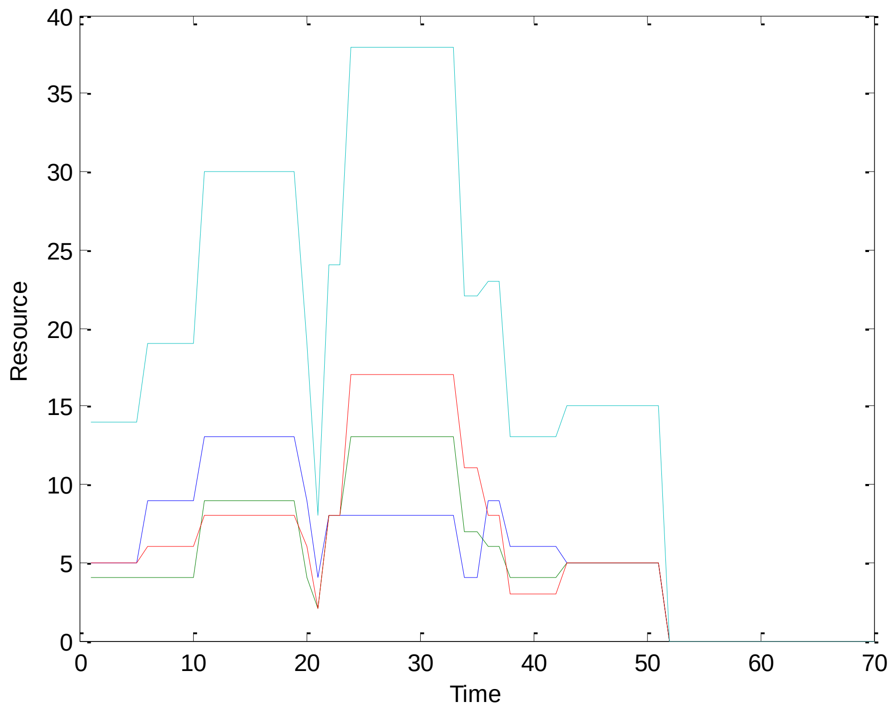
APPENDIX B


Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70

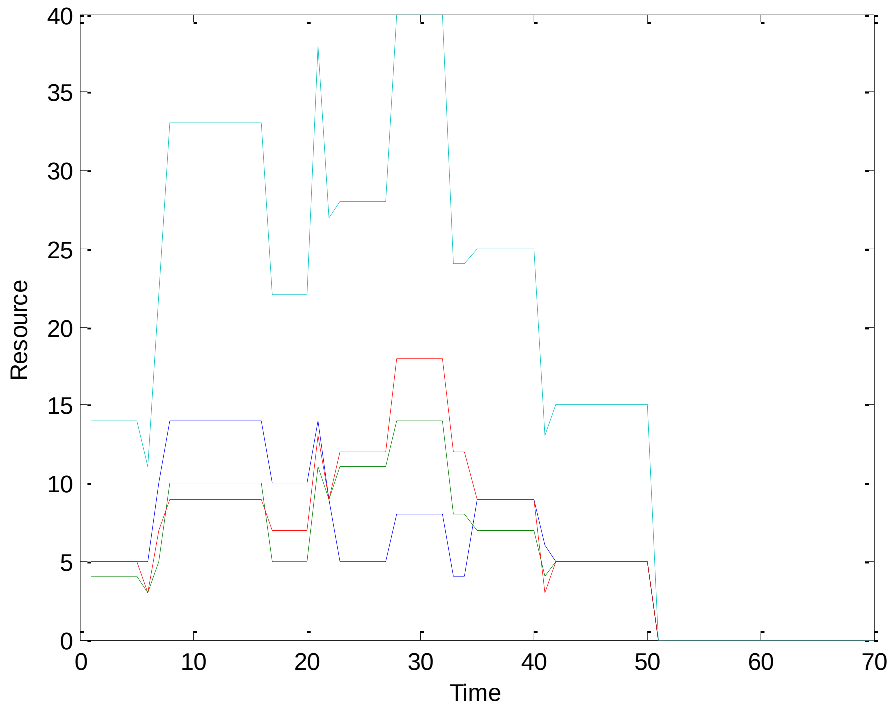

Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70


35
30
25
20
15
10
5
0
Resource
0
10
20
30
40
50
60
70
Time

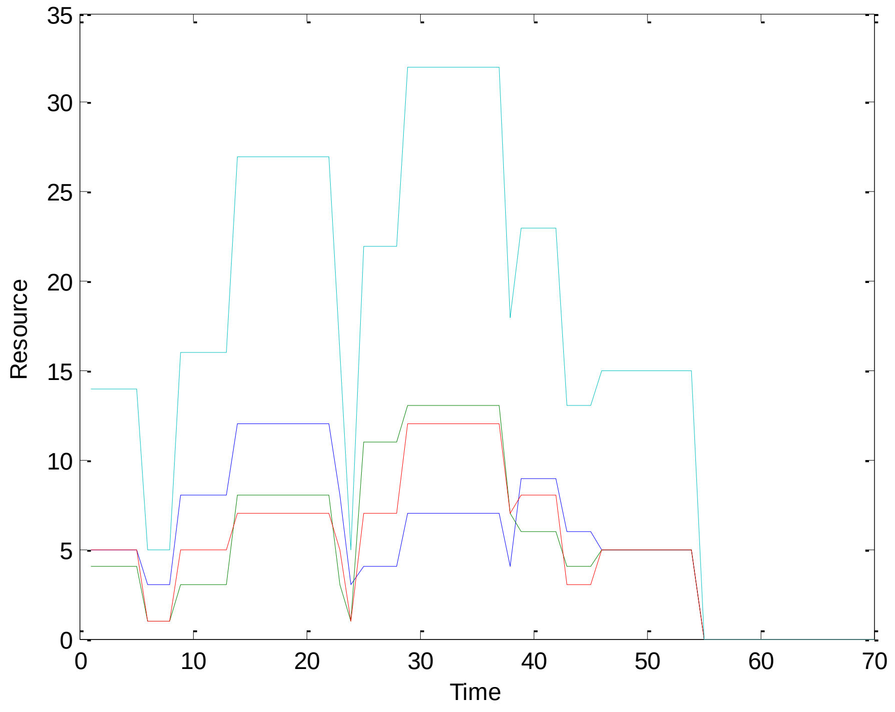

40
35
30
25
20
15
10
5
0
Resource
0
10
20
30
40
50
60
70
Time

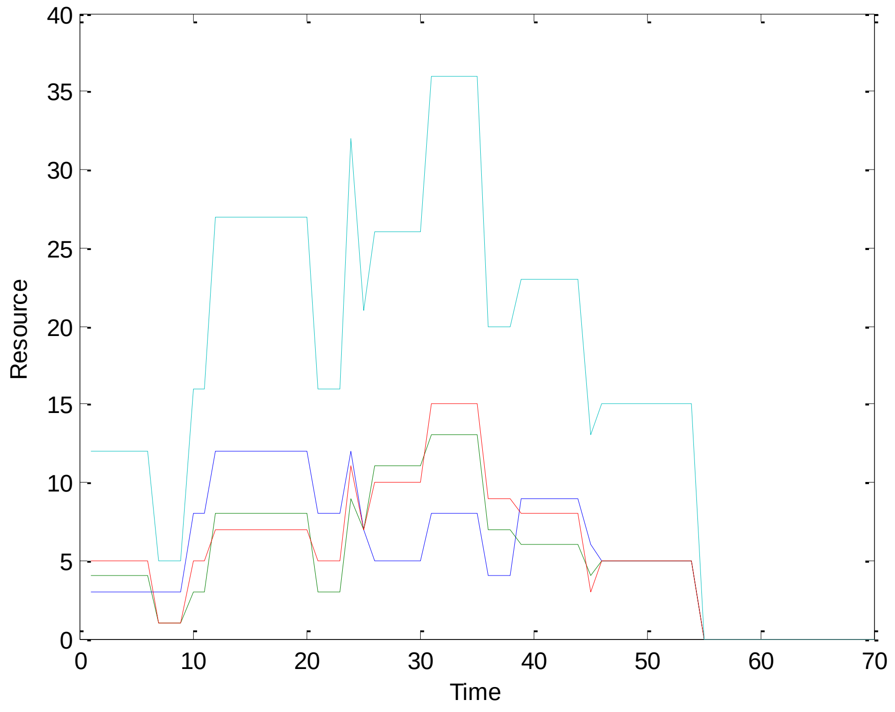


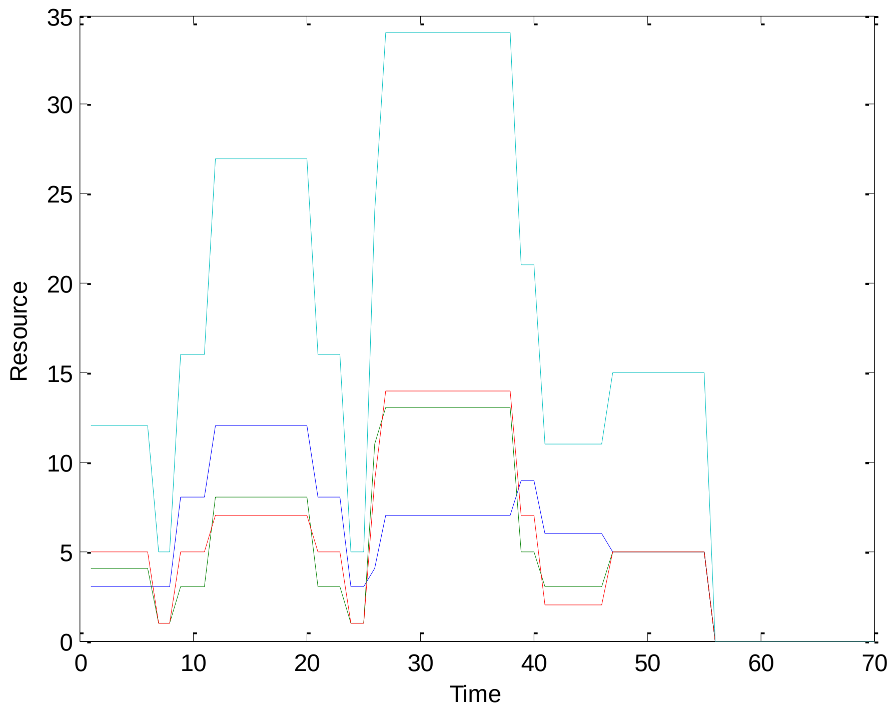
Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70

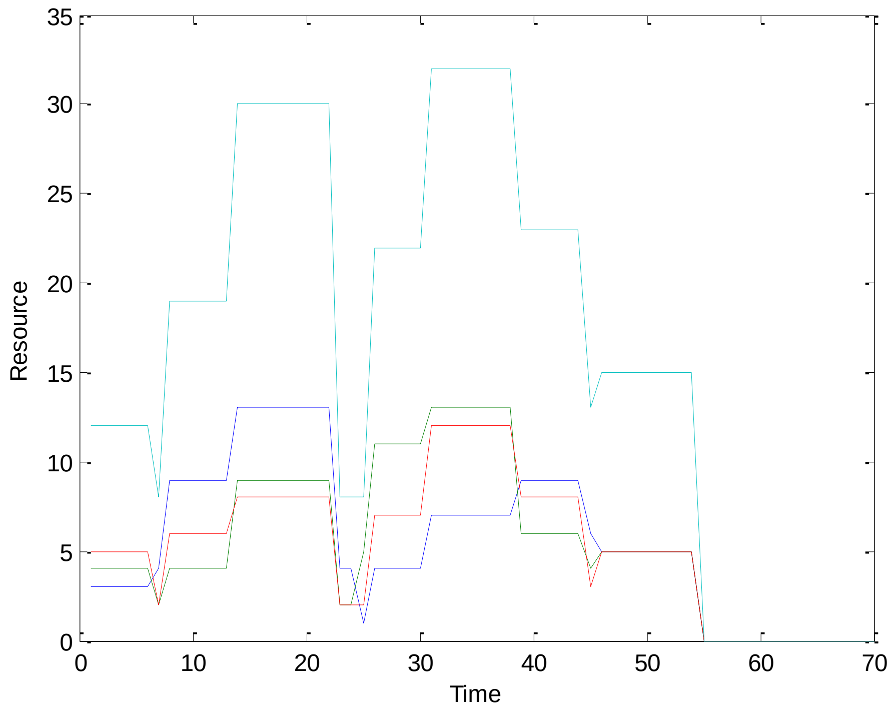


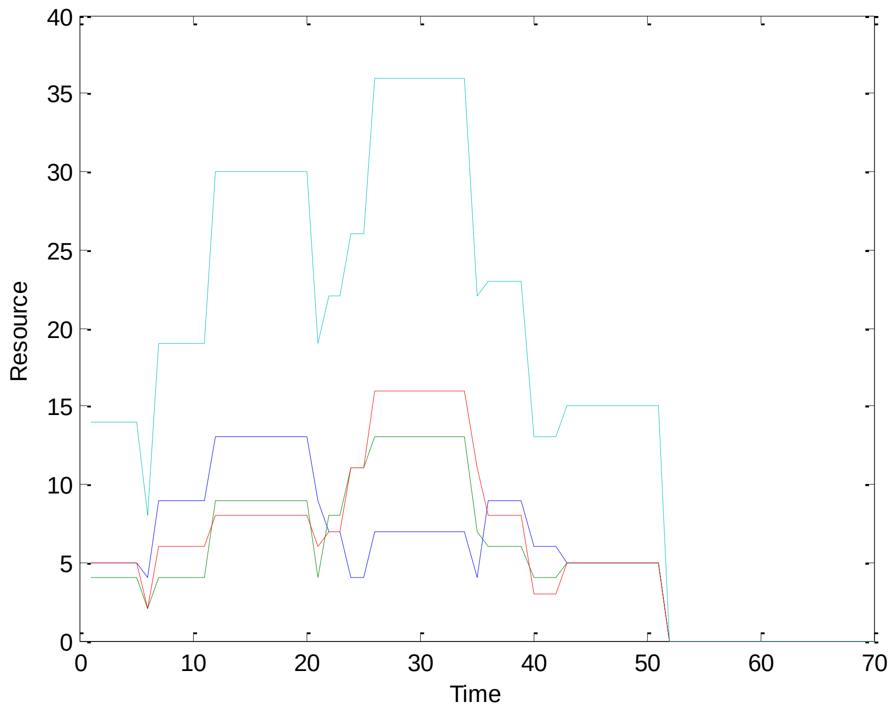
Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70

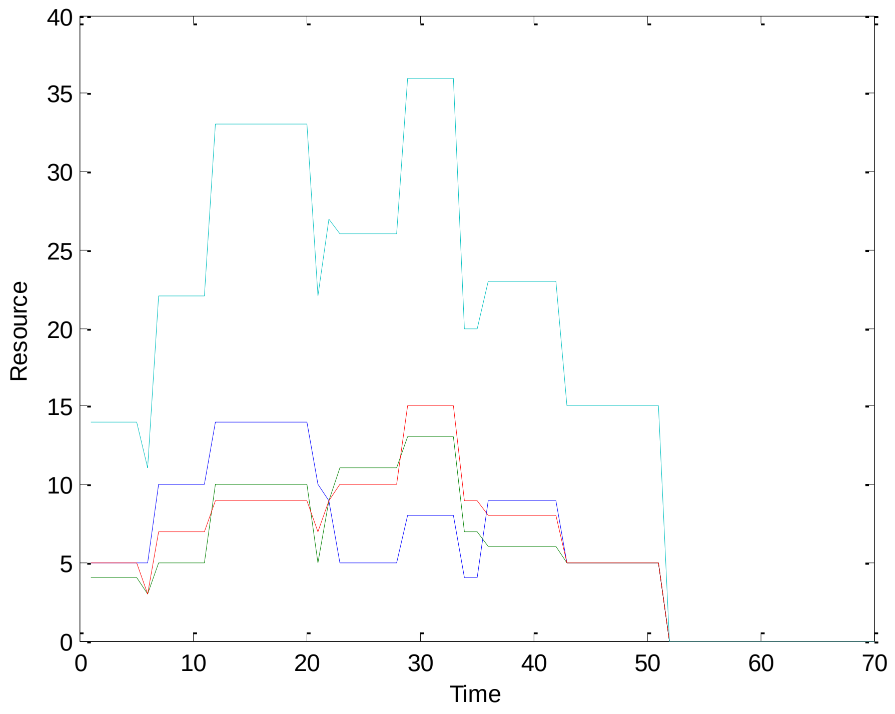

Resource
Time
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
70

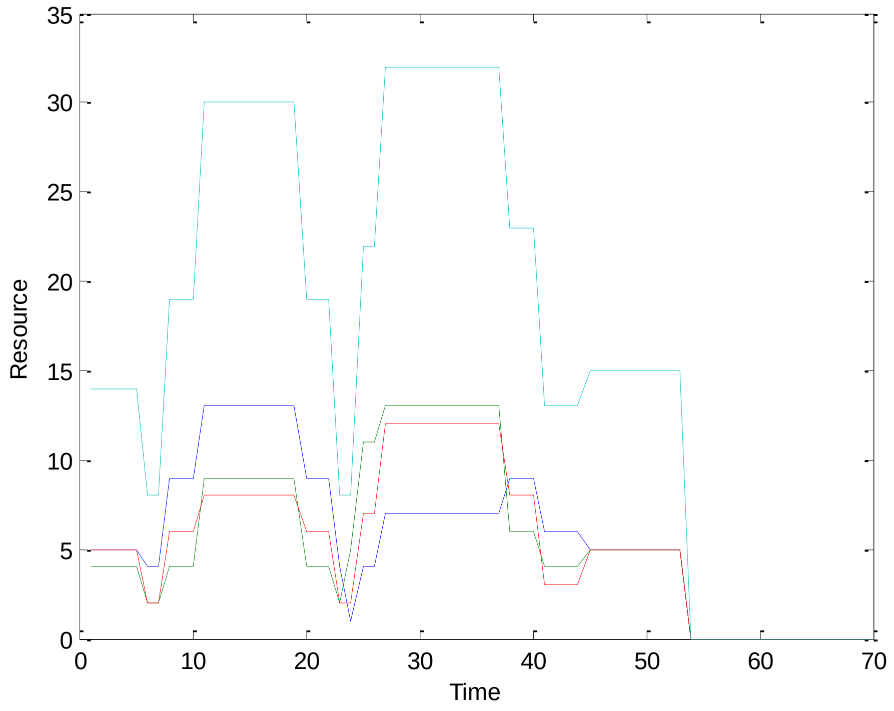

Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70

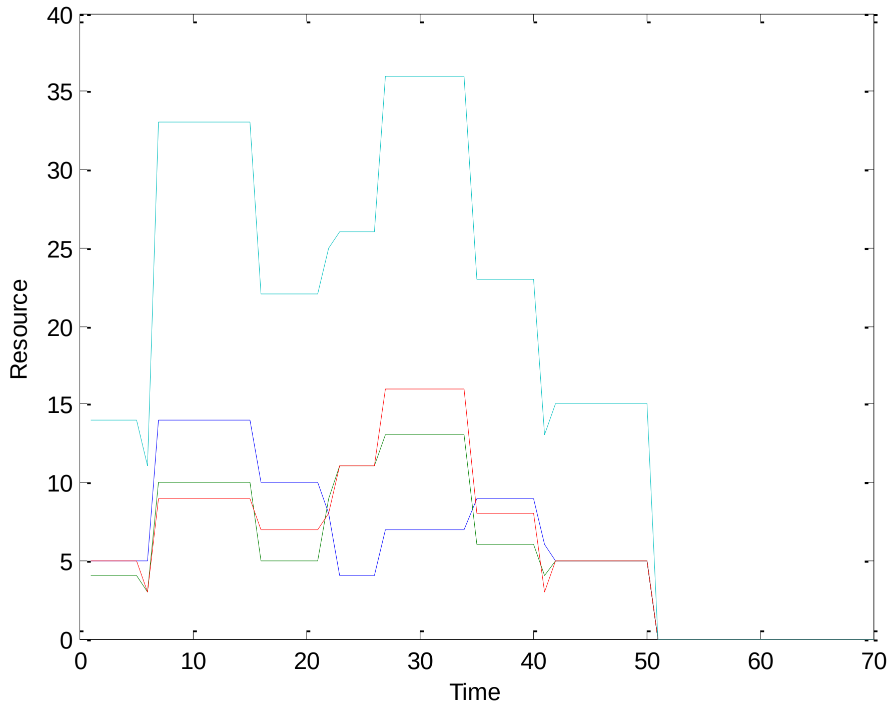

Resource
Time
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
70

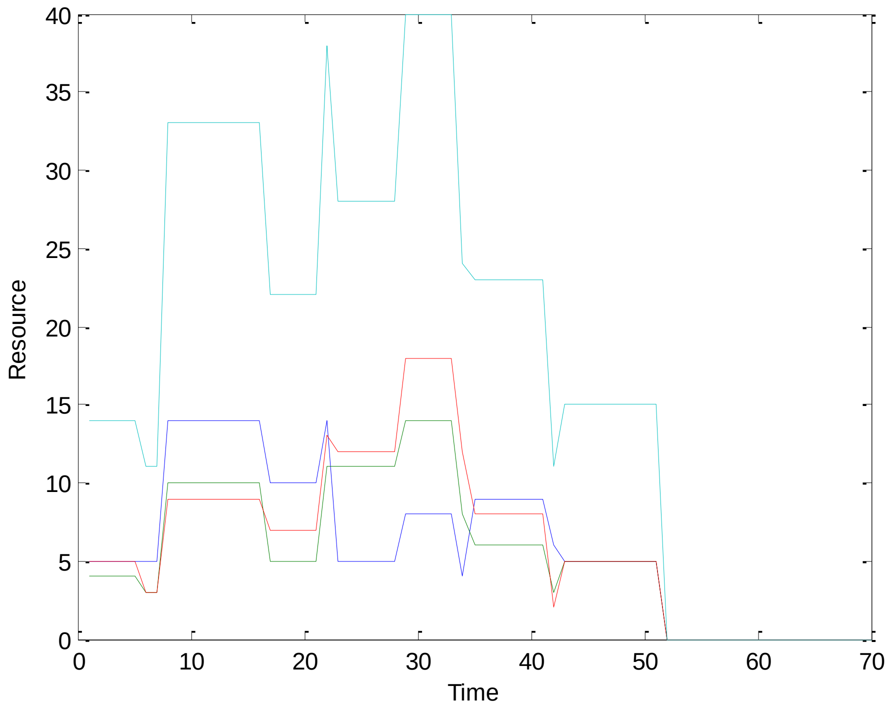

Resource
Time
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
70

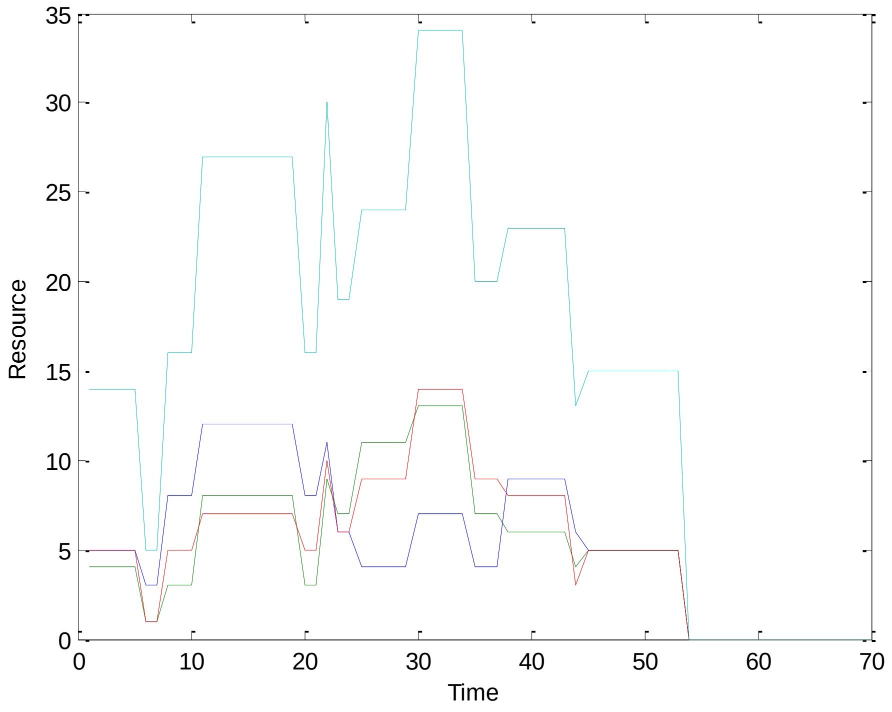

Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70

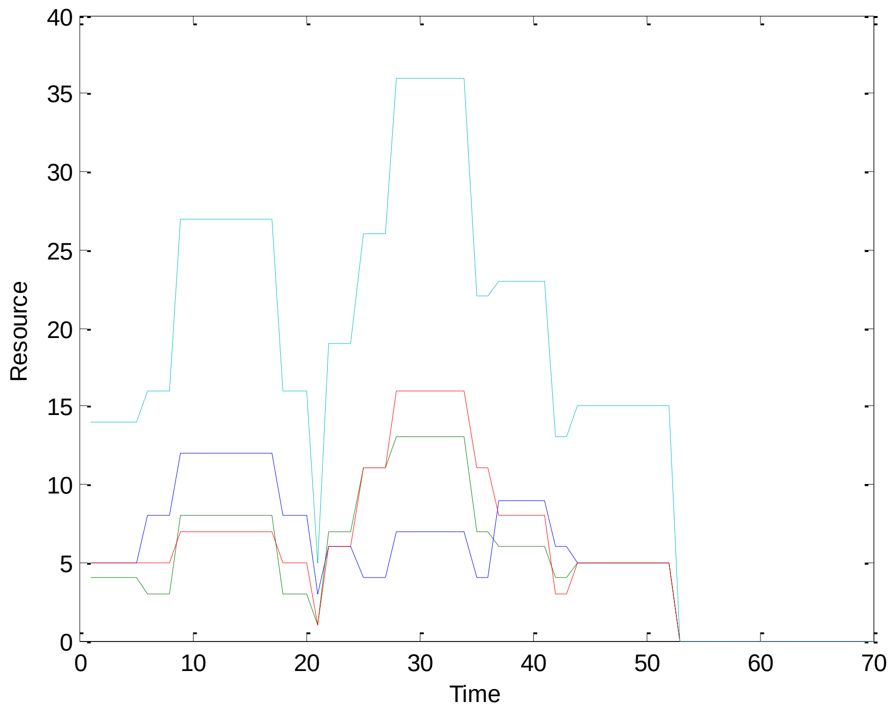

Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70


Resource
Time
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
70

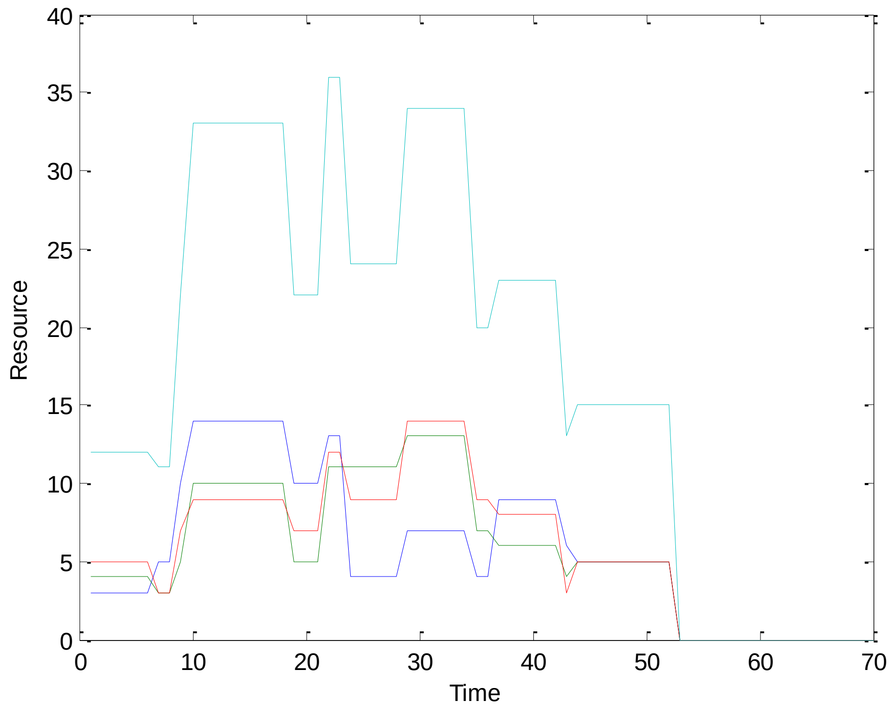

Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70


Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70

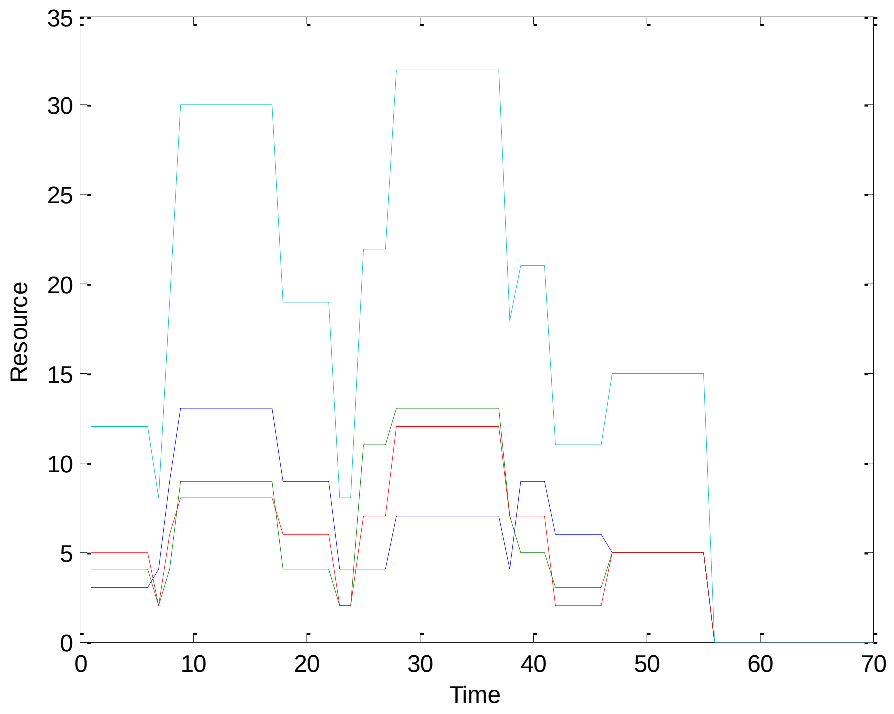

Resource
Time
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
70



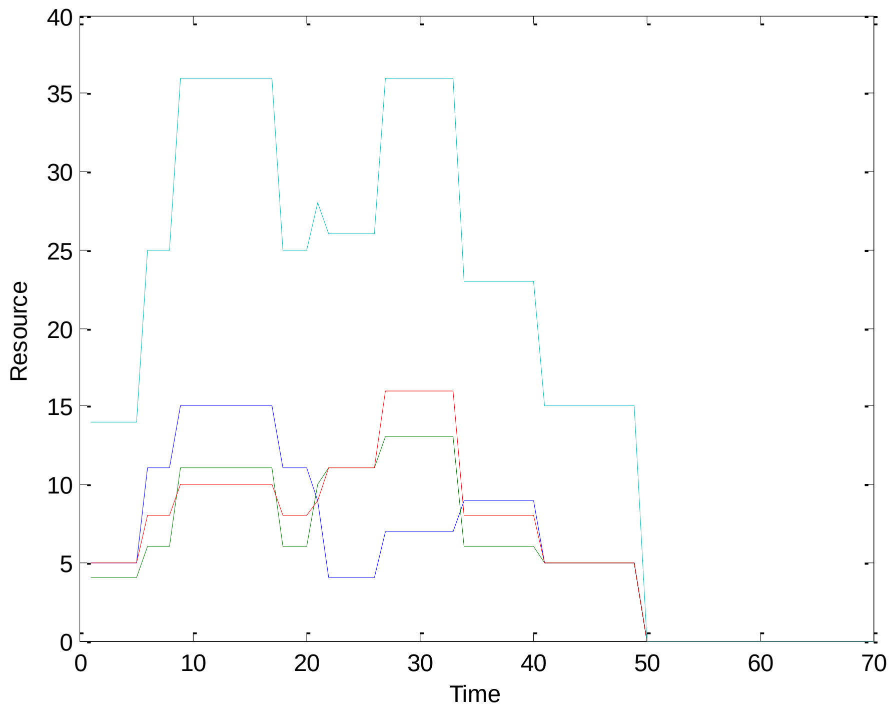
40
35
30
25
20
15
10
5
0
Resource
0
10
20
30
40
50
60
70
Time



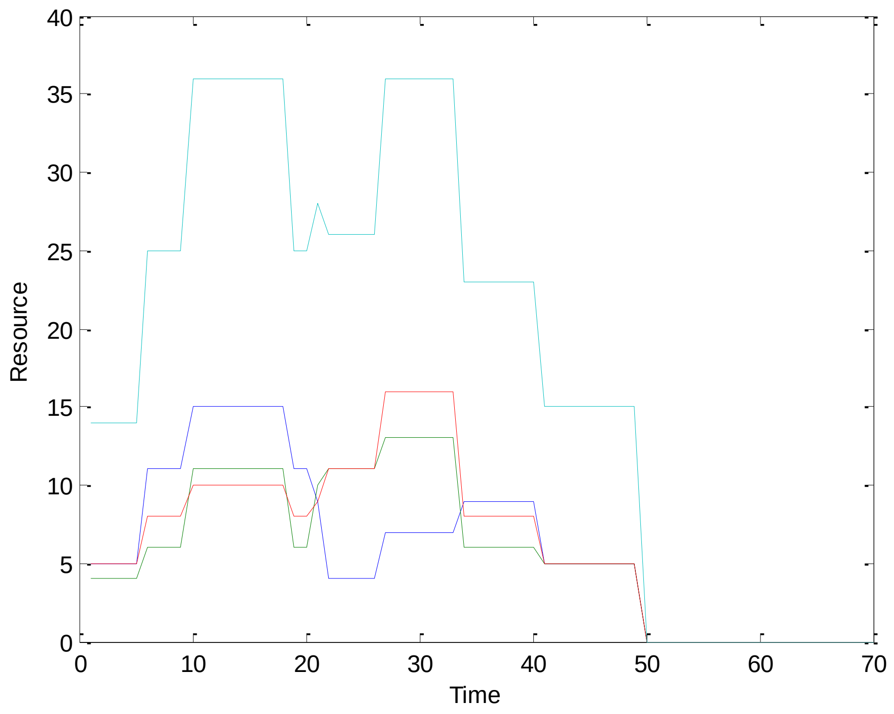
40
35
30
25
20
15
10
5
0
Resource
0
10
20
30
40
50
60
70
Time



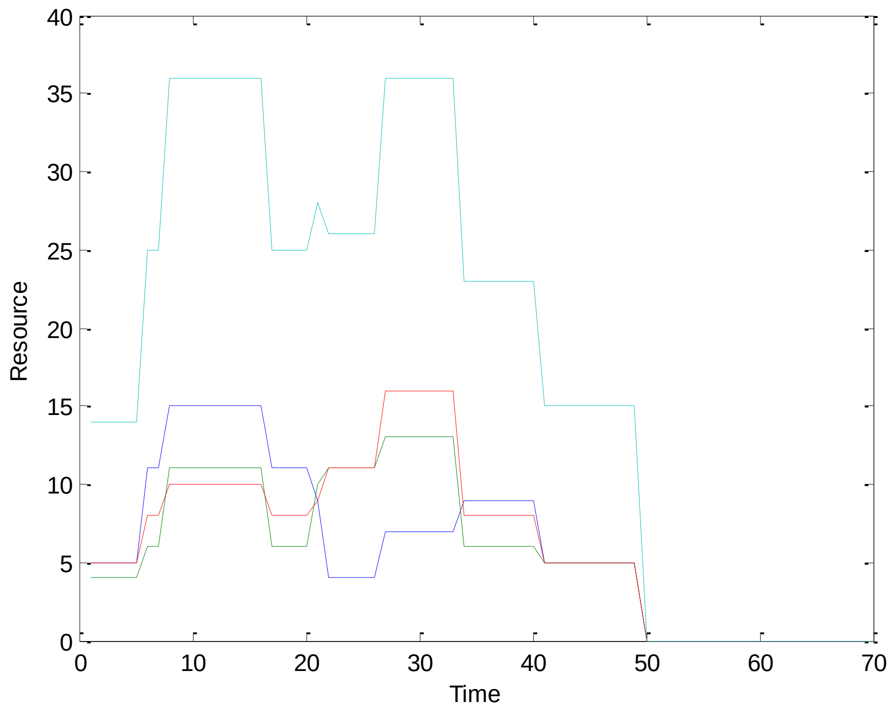
Resource
Time
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
70



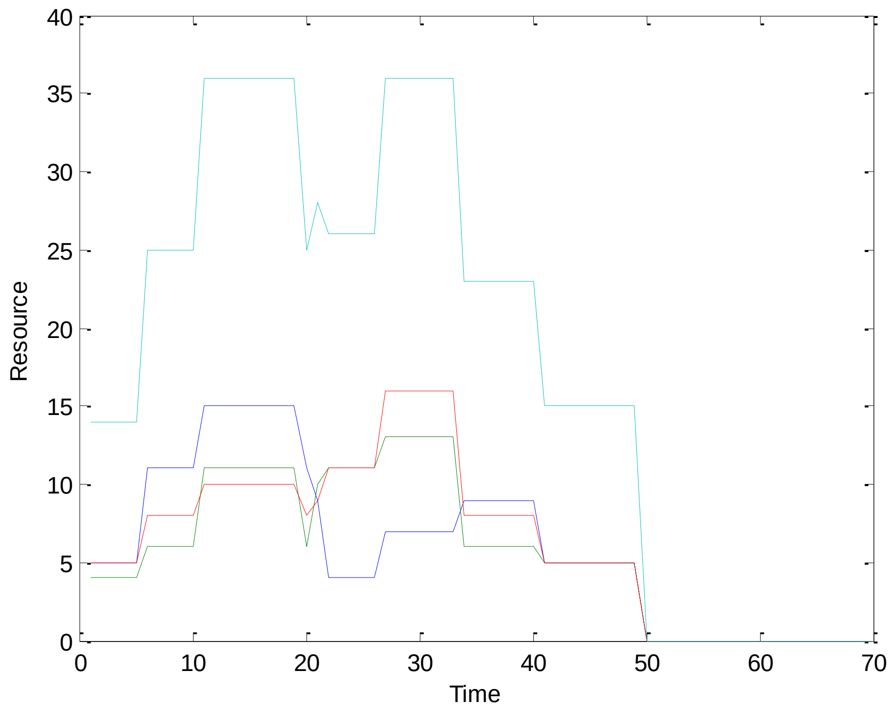
Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70



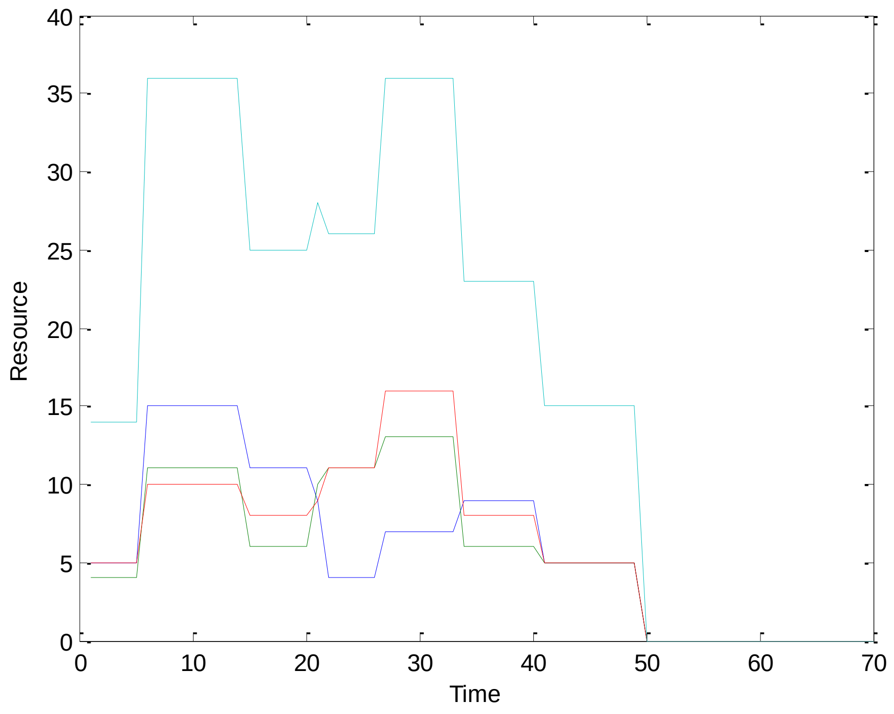
Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70



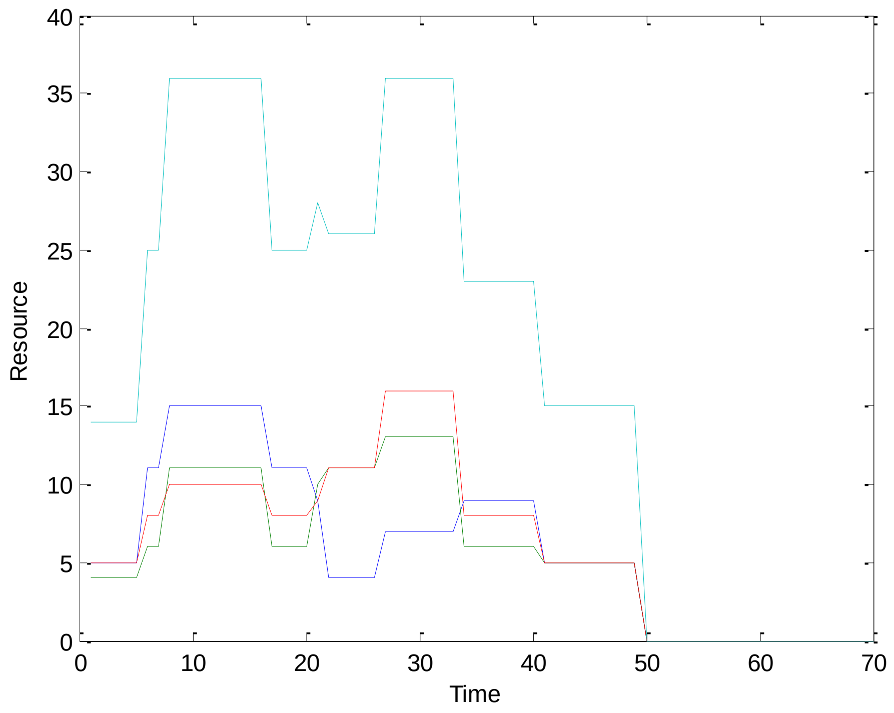
Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70

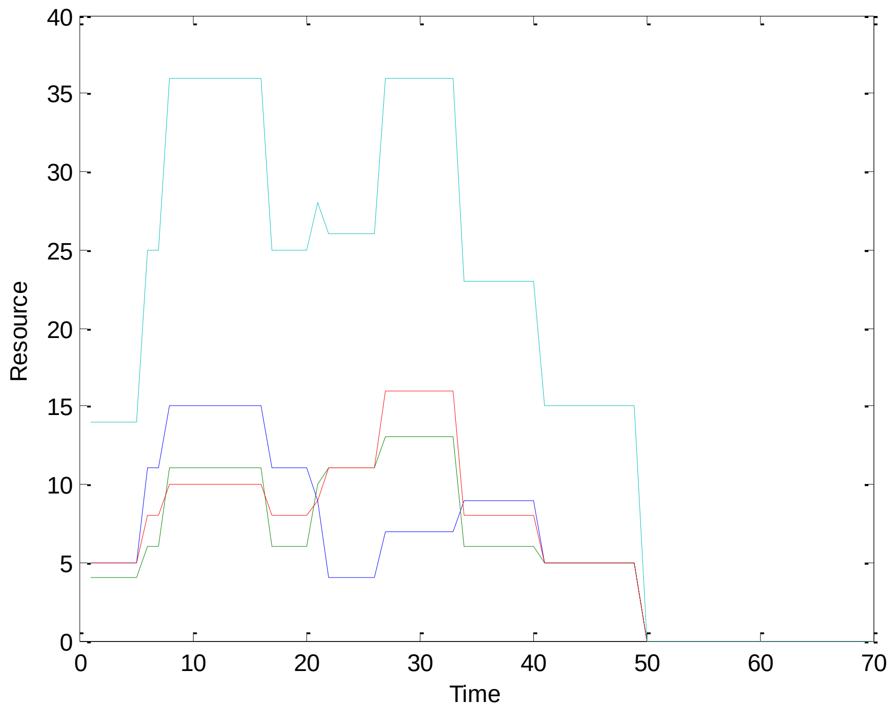


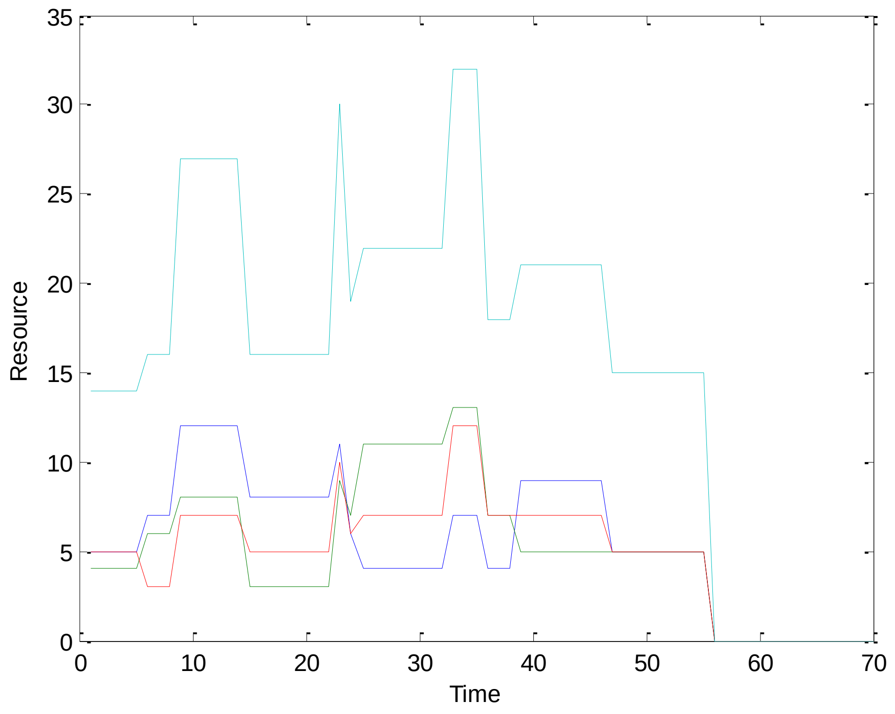
Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70

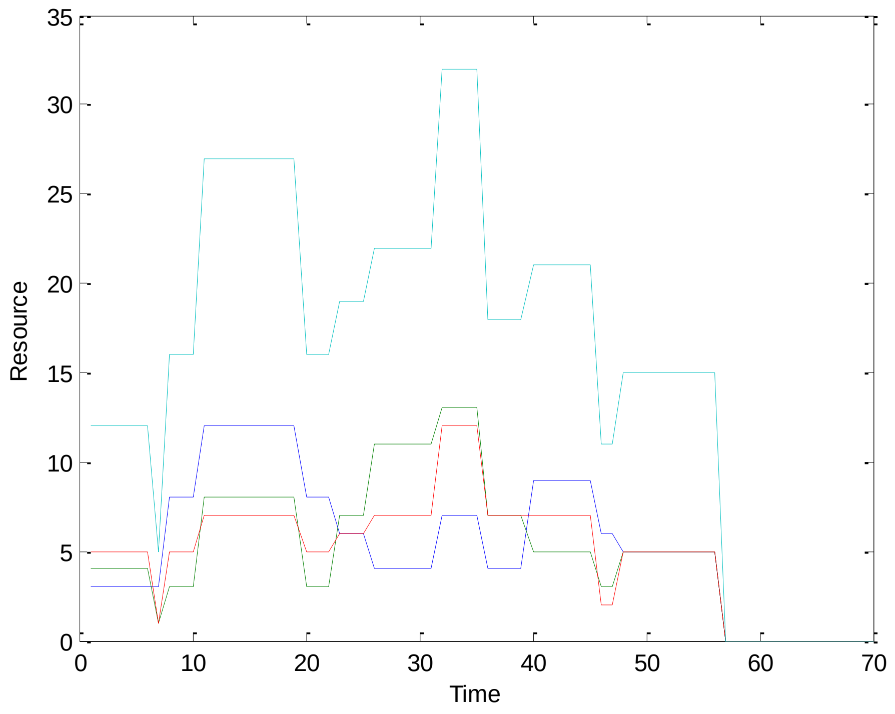


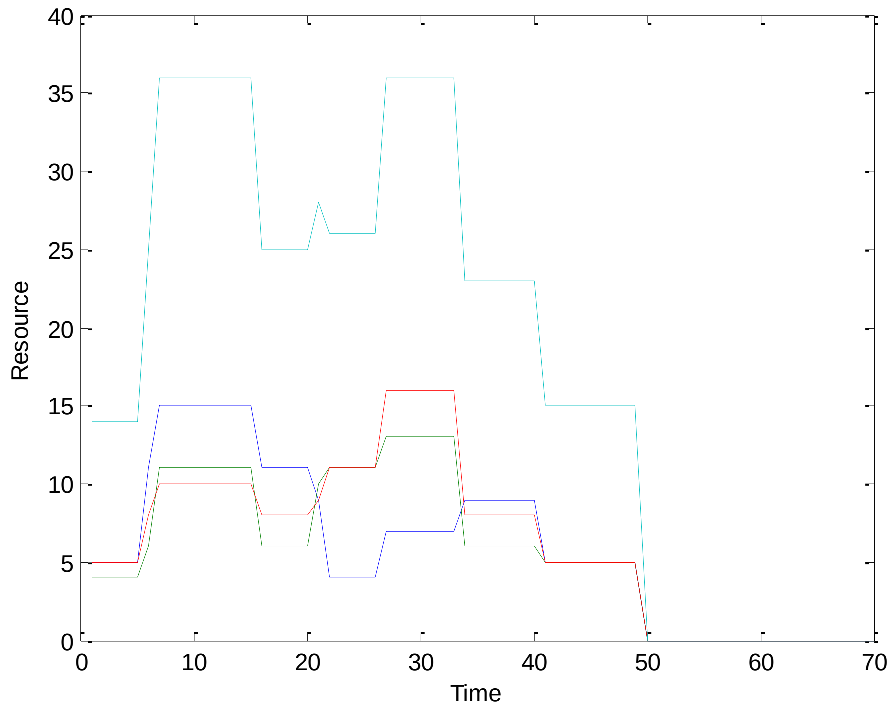
Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70

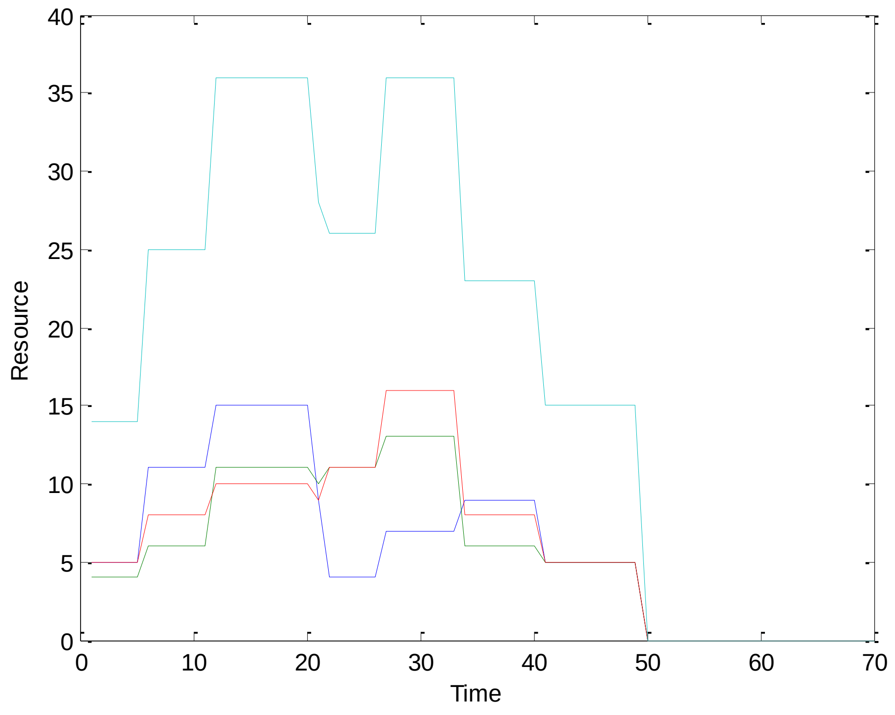

Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70


Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70


Resource
Time
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
70


Resource
Time
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
70


Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70


Resource
Time
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70

APPENDIX C

Main file

```
clear all
close all
clc

% Variable setting

tournamentSize = 10;
r_num_comp=5;
pop=100;
pool_size=pop;
tour_size=3;
gen=500;
cross_rate=0.9;
mutation_rate=0.5;
n_solution=30;
M=3;
V=18;
% sharing 1 is abs comparison, 2 is m comparison
sharing=1;
ut=2;

% for duration, cost and leveling index, ancor points are found
durationPF
costPF
levelingPF
% step 1 - find anchor points
anchor_cost
anchor_duration
anchor_leveling
%step 2 - objective mapping

utopia=[elite_duration,elite_cost,elite_leveling];
anchors=[anchor_duration;anchor_cost;anchor_leveling];


max_its_duration=max(anchors(:,1));
max_its_cost=max(anchors(:,2));
max_its_leveling=max(anchors(:,3));

l_1= max_its_duration-elite_duration;
l_2= max_its_cost-elite_cost;
l_3= max_its_leveling-elite_leveling;

% inital scheduling poopulation is generated
```

```
problem_setting
initialize_va

% space ormalization is performed
normalized=[population,(result_duration-elite_duration)/l_1,(plus_cost-
elite_cost)/l_2,(leveling_idx'-elite_leveling)/l_3];

nor_anchor_cost=[(max_its_duration-elite_duration)/l_1,0];
nor_anchor_duration=[0,(max_its_cost-elite_cost)/l_2];
nor_anchor_leveling=[0,0,(max_its_leveling-elite_leveling)/l_3];




clear plus_cost
clear result_duration
clear population
clear new_population
clear xoverKids

r_num_comp=3;
pop=100
pool_size=pop;
tour_size=3;
gen=200;
cross_rate=0.9;
M=3;
V=18;

% sharind redius is defined
sharing=1;
% Before generation loop is started, initial population is generated
again
problem_setting
initialize_va

% Individuals in the population are normalized using the outcome
obtained
% from anchor point search
norm_result_duration=(result_duration-elite_duration)/l_1;
norm_plus_cost=(plus_cost-elite_cost)/l_2;
norm_leveling=(leveling_idx-elite_leveling)/l_3;


result_duration=norm_result_duration;
plus_cost=norm_plus_cost;
leveling_idx=norm_leveling;

% to the population, the results are attached for coding convenience
chromosome=[population,result_duration,plus_cost,leveling_idx'];
```

```
% Non-dominated solutions are selected from population and sent to
external
% archive
chromosome_sort = non_domination_sort_mod(chromosome);
g=find(chromosome_sort(:,end-1)==1);
for i=1:size(g,1)
gt1(i)=chromosome_sort(i,end-4);
gt2(i)=chromosome_sort(i,end-3);
gt3(i)=chromosome_sort(i,end-2);
end
external=chromosome_sort(g,:)

% generation run starts
for ut=1:gen

    if ut~=1
norm_result_duration=(result_duration-elite_duration)/l_1;
norm_plus_cost=(plus_cost-elite_cost)/l_2;
norm_leveling=(leveling_idx-elite_leveling)/l_3;

result_duration=norm_result_duration;
plus_cost=norm_plus_cost;
leveling_idx=norm_leveling;

not_sorted=[population,result_duration,plus_cost,leveling_idx'];
chromosome_sort = non_domination_sort_mod(not_sorted);
g=find(chromosome_sort(:,end-1)==1);
external=[external;chromosome_sort(g,:)];
    end


if ut==gen
g=find(chromosome_sort(:,end-1)==1);
for i=1:size(g,1)
gt1(i)=chromosome_sort(i,end-4);
gt2(i)=chromosome_sort(i,end-3);
gt3(i)=chromosome_sort(i,end-2);
end

% best solutions are sent to external archive
external_pareto=non_domination_sort_mod(external(:,1:end-2));
h=find(external_pareto(:,end-1)==1);
external_to_pool=external_pareto(h,:);

for i=1:size(h,1)
ht1(i)=external_pareto(i,end-4);
ht2(i)=external_pareto(i,end-3);
ht3(i)=external_pareto(i,end-2);
end


end
```

```
iu=1;
while iu<pop+1
% pareto domination tournament selection starts
jj=find(chromosome_sort(:,end-1)<=4);
chromosome_high_rank=chromosome_sort(jj,:);
high_rank_duration=chromosome_high_rank(:,end-3);
high_rank_plus_cost=chromosome_high_rank(:,end-2);



r_num_indi=randsample(size(jj,1),2);
candidate=chromosome_high_rank(r_num_indi,:);
%candidate and comparison set define
candidate_duration=chromosome_high_rank(r_num_indi,end-3);
candidate_cost=chromosome_high_rank(r_num_indi,end-2);

comparison_count=randsample(size(jj,1),r_num_comp);
comparison_duration=chromosome_high_rank(comparison_count,end-3);
comparison_cost=chromosome_high_rank(comparison_count,end-2);



% candidate 1 comparison
a=length(find(candidate_duration(1)>comparison_duration));
b=length(find(candidate_cost(1)>comparison_cost));
% candidate 2 comparison
c=length(find(candidate_duration(2)>comparison_duration));
d=length(find(candidate_cost(2)>comparison_cost));

if a==0 & b==0
    if c~=0 | d~=0
        new_population(iu,:)=candidate(1,:);
% chromosome_high_rank(r_num_indi(1),:)=[];
        iu=iu+1;
    else

    sharing_r

    iu=iu+1;
   end

elseif a~=0 | b~=0
    if c==0 && d==0
        new_population(iu,:)=candidate(2,:);
        iu=iu+1;
    else

    sharing_r
    iu=iu+1;
   end
    end
end

%crossover operator
nKids = pop;
iq=1;
while iq<nKids+1

    if rand()>=cross_rate
      xoverKids(iq,:)=new_population(iq,1:end-5)
      iq=iq+1;
    else
    % get parents
    parent1 = new_population(iq,1:end-5);

    if iq~=pop
    parent2 = new_population(iq+1,1:end-5);
    end

    p2num=iq+1;

    % cut point
    xOverPoint = ceil(rand * (size(taskprecedence,1) - 1));
    % make one child

    xoverKids(iq,:)    = [ parent1(1:xOverPoint),parent2((xOverPoint +
1 ):
size(taskprecedence,1) ),parent1((size(taskprecedence,1)+1):(size(taskp
recedence,1))+xOverPoint),parent2((size(taskprecedence,1)+1)+xOverPoint
:end)];
    xoverKids(p2num,:) = [ parent2(1:xOverPoint),parent1((xOverPoint +
1 ):
size(taskprecedence,1) ),parent2((size(taskprecedence,1)+1):(size(taskp
recedence,1))+xOverPoint),parent1((size(taskprecedence,1)+1)+xOverPoint
:end)];
    iq=iq+2;
        end
end

% mutation
for iu=1:size(xoverKids,1)
    if rand()>=mutation_rate
        xoverKids(iu,:)=xoverKids(iu,:);
    else
        mutation_point=ceil(rand() * (size(taskprecedence,1) - 1));
        xoverKids(iu,mutation_point)=ceil(rand() *
(maxreduc(mutation_point) - 1));
        xoverKids(iu,mutation_point+size(taskprecedence,1))=rand;
    end
end
```

```
problem_setting
for ixo=1:pop
population(ixo)=xoverKids(ixo);
end
initialize_va


end




% Pareto optimal solutions are denormalized
tt1=ht1*l_1+elite_duration
tt2=ht2*l_2+elite_cost
tt3=ht3*l_3+elite_leveling
```

problem_setting.m

```
%problem setting
taskduration   = [6;19;15;9;14;13;14;8;9];
taskprecedence = [0  0  0  0  0  0  0  0  0;...
    1  0  0  0  0  0  0  0  0;...
    1  0  0  0  0  0  0  0  0;...
    1  0  0  0  0  0  0  0  0;...
    0  1  1  0  0  0  0  0  0;...
    0  0  1  1  0  0  0  0  0;...
    0  1  0  0  0  0  0  0  0;...
    0  0  0  0  1  1  0  0  0;...
    0  0  0  0  1  0  1  1  0;];




maxreduc    = [1;3;0;0;2;1;1;1;0];
    activity_cost_1 = [300;2000;420;450;1050;600;1200;640;560];
    activity_cost_2 = [480;2600;420;450;1450;850;1900;950;560];
    activity_cost_3 = [480;3220;420;450;1860;850;1900;950;560];
    activity_cost_4 = [480;3860;420;450;1860;850;1900;950;560];
activity_cost=
[activity_cost_1,activity_cost_2,activity_cost_3,activity_cost_4];

    resource_1_0day=[3;3;5;4;1;3;3;6;5]
    resource_1_1day=[5;4;5;4;1;4;3;6;5]
    resource_1_2day=[5;5;5;4;1;4;3;6;5]
    resource_1_3day=[5;6;5;4;1;4;3;6;5]
resource_1=[resource_1_0day,resource_1_1day,resource_1_2day,resource_1_
3day];
```

```
    resource_2_0day=[4;1;2;5;5;6;2;3;5]
    resource_2_1day=[4;2;2;5;5;6;3;4;5]
    resource_2_2day=[4;3;2;5;5;6;3;4;5]
    resource_2_3day=[4;4;2;5;5;6;3;4;5]
resource_2=[resource_2_0day,resource_2_1day,resource_2_2day,resource_2_
3day];

    resource_3_0day=[5;1;4;2;2;5;5;2;5]
    resource_3_1day=[5;2;4;2;4;6;6;3;5]
    resource_3_2day=[5;3;4;2;6;6;6;3;5]
    resource_3_3day=[5;4;4;2;6;6;6;3;5]
resource_3=[resource_3_0day,resource_3_1day,resource_3_2day,resource_3_
3day];


tot_duration = 0;
num_pre=zeros(size(taskprecedence,1),2);
pre_activity=zeros(size(taskprecedence,1),size(taskprecedence,1));

est = zeros(size(taskprecedence,1),1);
eft = zeros(size(taskprecedence,1),1);
lst = zeros(size(taskprecedence,1),1);
lft = zeros(size(taskprecedence,1),1);
c_p = zeros(size(taskprecedence,1),1);
inde = zeros(1,size(taskprecedence,1));

% output setting
plus_cost=zeros(1,pop);
result_duration=zeros(1,pop);

% initialize
for ik=1:pop
for cr=1:size(maxreduc)
    if maxreduc(cr)==0
       cromosome_s(cr)=0;
    end
    if maxreduc(cr)~=0
       cromosome_s(cr) = round((rand(1,1) * maxreduc(cr)) + 0);
    end
end

population(ik,:)=[cromosome_s,rand(1,size(maxreduc))];
end
```

initialize_va.m

```
plus_cost=zeros(pop,1);
```

```
for i=1:pop
    plus_cost(i)=0;
    for j=1:size(population,2)/2
        plus_cost(i)=plus_cost(i)+activity_cost(j,(population(i,j)+1));
    end
end

for kk=1:pop
new_duration(kk,:)=taskduration-population(kk,1:size(population,2)/2)';
end


resource_cumul_1=zeros(pop,70);
resource_cumul_2=zeros(pop,70);
resource_cumul_3=zeros(pop,70);
resource_cumul_total=zeros(pop,70);
% early duration
result_duration=zeros(pop,1);
for iu=1:pop
t=0;
%find number of predecessor
for i=1:size(taskprecedence,1)%18
    i;
    num_pre(i,1) = i;
    num_pre(i,2) = length(find(taskprecedence(i,:)==1));

    %no predecessor
    if(num_pre(i,2)==0)
    est(i,1)=0;
    eft(i,1)=est(i,1)+new_duration(iu,i);
    t=t+1;
    end

    %predecessor exist
    if(num_pre(i,2)~=0)
        i;
    tem = zeros(1,size(taskprecedence,1));
    tem = taskprecedence(i,:);
    c = find(tem==1);

    c;

    est(i,1) = max(eft(c));
    eft(i,1) = est(i,1) + new_duration(iu,i);
    end
end
result_duration(iu)=max(eft);
```

```
%late duration
    lft = zeros(size(taskprecedence,1),1);
    for i=size(taskprecedence,1):-1:1

        num_succ(i,1) = i;
        num_succ(i,2) = length(find(taskprecedence(:,i)==1));

        if (num_succ(i,2)==0)
        final_activity=find(eft==result_duration(iu));
        lft(i,1)=eft(final_activity);
        lst(i,1)=eft(final_activity)-new_duration(iu,final_activity);

        elseif (num_succ(i,2)~=0)

        tem2 = taskprecedence(:,i);
        c2 = find(tem2==1);

        lft(i,1)=min(lst(c2));
        lst(i,1)=lft(i,1) - new_duration(iu,i);
        end
    end
    iu;
    total_float=[lft-eft]';

use_float(iu,:)=round(total_float.*population(iu,size(population,2)/2+1
:end));
    start_date(iu,:)=est'+use_float(iu,:);


if ut==1
    init_resource_cumul_1=zeros(pop,70);
    init_resource_cumul_2=zeros(pop,70);
    init_resource_cumul_3=zeros(pop,70);
    init_resource_cumul_total=zeros(pop,70);

    for k=1:size(taskprecedence,1)
        resource_1_crash(k)=resource_1(k,population(iu,k)+1);
    end

    for l=1:size(taskprecedence,1)

        for aaa=start_date(iu,l)+1:start_date(iu,l)+new_duration(iu,l)

init_resource_cumul_1(iu,aaa)=init_resource_cumul_1(iu,aaa)+resource_1_
crash(l);

        end
    end

    for k=1:size(taskprecedence,1)
        resource_2_crash(k)=resource_2(k,population(iu,k)+1);
    end

    for l=1:size(taskprecedence,1)

        for aaa=start_date(iu,l)+1:start_date(iu,l)+new_duration(iu,l)

init_resource_cumul_2(iu,aaa)=init_resource_cumul_2(iu,aaa)+resource_2_
crash(l);

        end
    end


    for k=1:size(taskprecedence,1)
        resource_3_crash(k)=resource_3(k,population(iu,k)+1);
    end

    for l=1:size(taskprecedence,1)

        for aaa=start_date(iu,l)+1:start_date(iu,l)+new_duration(iu,l)

init_resource_cumul_3(iu,aaa)=init_resource_cumul_3(iu,aaa)+resource_3_
crash(l);

        end
    end

init_resource_cumul_total(iu,:)=init_resource_cumul_1(iu,:)+init_resour
ce_cumul_2(iu,:)+init_resource_cumul_3(iu,:);

init_leveling_idx(iu)=std(init_resource_cumul_1(iu,:))+std(init_resourc
e_cumul_2(iu,:))+std(init_resource_cumul_3(iu,:));


else
    for k=1:size(taskprecedence,1)
        resource_1_crash(k)=resource_1(k,population(iu,k)+1);
    end

    for l=1:size(taskprecedence,1)

        for aaa=start_date(iu,l)+1:start_date(iu,l)+new_duration(iu,l)

resource_cumul_1(iu,aaa)=resource_cumul_1(iu,aaa)+resource_1_crash(l);

        end
    end

    for k=1:size(taskprecedence,1)
        resource_2_crash(k)=resource_2(k,population(iu,k)+1);
    end
    for l=1:size(taskprecedence,1)

        for aaa=start_date(iu,l)+1:start_date(iu,l)+new_duration(iu,l)

resource_cumul_2(iu,aaa)=resource_cumul_2(iu,aaa)+resource_2_crash(l);

        end
    end


    for k=1:size(taskprecedence,1)
        resource_3_crash(k)=resource_3(k,population(iu,k)+1);
    end

    for l=1:size(taskprecedence,1)

        for aaa=start_date(iu,l)+1:start_date(iu,l)+new_duration(iu,l)

resource_cumul_3(iu,aaa)=resource_cumul_3(iu,aaa)+resource_3_crash(l);

        end
    end

resource_cumul_total(iu,:)=resource_cumul_1(iu,:)+resource_cumul_2(iu,:
)+resource_cumul_3(iu,:);

leveling_idx(iu)=std(resource_cumul_1(iu,:))+std(resource_cumul_2(iu,:)
)+std(resource_cumul_3(iu,:));
end



  w_i=resource_cumul_1(k,:);
  x_i=resource_cumul_2(k,:);
  y_i=resource_cumul_3(k,:);
  z_i=resource_cumul_total(k,:);
  plot(w_i,'DisplayName','w','YDataSource','w');hold
all;plot(x_i,'DisplayName','x','YDataSource','x');plot(y_i,'DisplayName
','y','YDataSource','y');plot(z_i,'DisplayName','z','YDataSource','z');
hold off;figure(gcf);
   xlabel('Time')
   ylabel('Resource')

end
```

durationPF.m

```
problem_setting
```

```
initialize_va

subplot(1,1,1)
scatter(result_duration, plus_cost, 'DisplayName', 'plus_cost vs
result_duration', 'XDataSource', 'result_duration', 'YDataSource',
'plus_cost'); figure(gcf)
xlabel('Conctruction Period(days)')
ylabel('Added Cost')
title('Initial Population')

elite_duration=min(result_duration);

for ut=1:gen



chromosome=[population,result_duration,plus_cost,leveling_idx'];

candidate_elite_duration=min(chromosome(:,end-2));
candidate_elite=chromosome(find(chromosome(:,end-
2)==candidate_elite_duration),:);
if candidate_elite_duration<=elite_duration
    elite_duration=candidate_elite_duration;
    elite=candidate_elite(1,:);
end
elite_duration_list(ut,:)=elite;

subplot(1,1,1)

playerlist = ceil(10 * rand(pop/2-1,tournamentSize));

playerSize = size(playerlist,1);

for i = 1:playerSize
    players = chromosome(playerlist(i,:),:);
    players_duration=players(:,end-2);

    winner = players(1,:); % Assume that the first player is the winner
    for j = 2:size(players,1) % Winner plays against each other
consecutively
        score1 = winner(end);
        score2 = players_duration(j);
        if score2(1) < score1(1)
            winner = players(j,:);
        elseif score2(1) == score1(1)
            try % socre(2) may not be present for single objective
problems
                if score2(2) < score1(2)
                    winner = players(j,:);
                end
            catch
            end
        end
    end
    champions(i,:) = winner;
end
champions_plus = [champions;elite];

%crossover operator
new_population=[champions;elite];
nKids = pop/2;
% xoverKids = zeros(nKids,size(taskprecedence,1));
iq=1;
while iq<nKids

    if rand()>=cross_rate
      xoverKids(iq,:)=new_population(iq,1:end-3)
      iq=iq+1;
    else
    % get parents
    parent1 = new_population(iq,1:end-3)

    parent2 = new_population(iq+1,1:end-3)
    p2num=iq+1;

    % cut point
    xOverPoint = ceil(rand(1,1) * (size(taskprecedence,1) - 1));
    % make one child
    xoverKids(iq,:)    = [ parent1(1:xOverPoint),parent2((xOverPoint +
1 ):
size(taskprecedence,1) ),parent1((size(taskprecedence,1)+1):(size(taskp
recedence,1))+xOverPoint),parent2((size(taskprecedence,1)+1)+xOverPoint
:end)];
    xoverKids(p2num,:) = [ parent2(1:xOverPoint),parent1((xOverPoint +
1 ):
size(taskprecedence,1) ),parent2((size(taskprecedence,1)+1):(size(taskp
recedence,1))+xOverPoint),parent1((size(taskprecedence,1)+1)+xOverPoint
:end)];
    iq=iq+2;
    end
end

% mutation
for iu=1:size(xoverKids,1)
    if rand()>=mutation_rate
        xoverKids(iu,:)=xoverKids(iu,:);
    else
        mutation_point=ceil(rand() * (size(taskprecedence,1) - 1));
        xoverKids(iu,mutation_point)=ceil(rand() *
(maxreduc(mutation_point) - 1));
        xoverKids(iu,mutation_point+size(taskprecedence,1))=rand;
    end
end
```

```
problem_setting
for ixo=1:pop
population(ixo)=xoverKids(ixo);
end
population=[xoverKids;champions_plus(:,1:end-3)];
if size(population,1)==99 |size(population,1)==999
    population=[population;elite(1:end-3)];
end
initialize_va

end
tt=elite_duration_list(find(elite_duration_list(:,end-
2)==min(elite_duration_list(:,end-2))),:)
its_cost=max(tt(:,end-1))
its_leveling=max(tt(:,end))
anchor_duration=[elite_duration,its_cost,its_leveling]
```

costPF.m

```
problem_setting
initialize_va


elite_cost=min(plus_cost);


for ut=1:gen



chromosome=[population,result_duration,plus_cost,leveling_idx'];

candidate_elite_cost=min(chromosome(:,end-1));
candidate_elite=chromosome(find(chromosome(:,end-
1)==candidate_elite_cost),:);
if candidate_elite_cost<=elite_cost
    elite_cost=candidate_elite_cost;
    elite=candidate_elite(1,:);
end

elite_cost_list(ut,:)=elite;
%
subplot(1,1,1)
```

```
scatter(elite(end-1),elite(end),  'DisplayName', 'plus_cost vs
result_duration', 'XDataSource', 'result_duration', 'YDataSource',
'plus_cost'); figure(gcf)
% hold on;

%tournament selection


% Choose the players
playerlist = ceil(10 * rand(pop/2-1,tournamentSize));
% Play tournament

playerSize = size(playerlist,1);
% champions = zeros(1,playerSize);
% For each set of players
for i = 1:playerSize
    players = chromosome(playerlist(i,:),:);
    players_cost=players(:,end-1);
    % For each tournament
    winner = players(1,:); % Assume that the first player is the winner
    for j = 2:size(players,1) % Winner plays against each other
consecutively
        score1 = winner(end);
        score2 = players_cost(j);
        if score2(1) < score1(1)
            winner = players(j,:);
        elseif score2(1) == score1(1)
            try % socre(2) may not be present for single objective
problems
                if score2(2) < score1(2)
                    winner = players(j,:);
                end
            catch
            end
        end
    end
    champions(i,:) = winner;
end
champions_plus = [champions;elite];

%crossover operator
new_population=[champions;elite];
nKids = pop/2;
% xoverKids = zeros(nKids,size(taskprecedence,1));
iq=1;
while iq<nKids

    if rand()>=cross_rate
      xoverKids(iq,:)=new_population(iq,1:end-3);
      iq=iq+1;
    else
    % get parents
    parent1 = new_population(iq,1:end-3);

    parent2 = new_population(iq+1,1:end-3);
    p2num=iq+1;

    % cut point
    xOverPoint = ceil(rand(1,2) * (size(taskprecedence,1) - 1));
    % make one child
    xoverKids(iq,:)    = [ parent1(1:xOverPoint),parent2((xOverPoint +
1 ):
size(taskprecedence,1) ),parent1((size(taskprecedence,1)+1):(size(taskp
recedence,1))+xOverPoint),parent2((size(taskprecedence,1)+1)+xOverPoint
:end)];
    xoverKids(p2num,:) = [ parent2(1:xOverPoint),parent1((xOverPoint +
1 ):
size(taskprecedence,1) ),parent2((size(taskprecedence,1)+1):(size(taskp
recedence,1))+xOverPoint),parent1((size(taskprecedence,1)+1)+xOverPoint
:end)];    iq=iq+2;

        end
end

% mutation
for iu=1:size(xoverKids,1)
    if rand()>=mutation_rate
        xoverKids(iu,:)=xoverKids(iu,:);
    else
        mutation_point=ceil(rand() * (size(taskprecedence,1) - 1));
        xoverKids(iu,mutation_point)=ceil(rand() *
(maxreduc(mutation_point) - 1));
        xoverKids(iu,mutation_point+size(taskprecedence,1))=rand;
    end
end



problem_setting
for ixo=1:pop
population(ixo)=xoverKids(ixo);
end
population=[xoverKids;champions_plus(:,1:end-3)];
if size(population,1)==99 |size(population,1)==999
    population=[population;elite(1:end-3)];
end
initialize_va


end

tt=elite_cost_list(find(elite_cost_list(:,end-
1)==min(elite_cost_list(:,end-1))),:);
its_duration=max(tt(:,end-2));
```

```
its_leveling=max(tt(:,end));
anchor_cost=[its_duration,elite_cost,its_leveling];
```

levelingPF.m

```
problem_setting
initialize_va

subplot(1,1,1)
scatter(result_duration, plus_cost, 'DisplayName', 'plus_cost vs
result_duration', 'XDataSource', 'result_duration', 'YDataSource',
'plus_cost'); figure(gcf)
xlabel('Conctruction Period(days)')
ylabel('Added Cost')
title('Initial Population')

elite_leveling=min(leveling_idx);


for ut=1:gen


chromosome=[population,result_duration,plus_cost,leveling_idx'];

candidate_elite_leveling=min(chromosome(:,end));
candidate_elite=chromosome(find(chromosome(:,end)==candidate_elite_leve
ling),:);
if candidate_elite_leveling<=elite_leveling
    elite_leveling=candidate_elite_leveling;
    elite=candidate_elite(1,:);
end
elite_leveling_list(ut,:)=elite;

subplot(1,1,1)
scatter(elite(end-1),elite(end),  'DisplayName', 'plus_cost vs
result_duration', 'XDataSource', 'result_duration', 'YDataSource',
'plus_cost'); figure(gcf)


% Choose the players
playerlist = ceil(10 * rand(pop/2-1,tournamentSize));
% Play tournament

playerSize = size(playerlist,1);
% champions = zeros(1,playerSize);
% For each set of players
```

```
for i = 1:playerSize
    players = chromosome(playerlist(i,:),:);
    players_leveling=players(:,end);
    % For each tournament
    winner = players(1,:); % Assume that the first player is the winner
    for j = 2:size(players,1) % Winner plays against each other
consecutively
        score1 = winner(end);
        score2 = players_leveling(j);
        if score2(1) < score1(1)
            winner = players(j,:);
        elseif score2(1) == score1(1)
            try % socre(2) may not be present for single objective
problems
                if score2(2) < score1(2)
                    winner = players(j,:);
                end
            catch
            end
        end
    end
    champions(i,:) = winner;
end
champions_plus = [champions;elite];

%crossover operator
new_population=[champions;elite];
nKids = pop/2;
iq=1;
while iq<nKids

    if rand()>=cross_rate
      xoverKids(iq,:)=new_population(iq,1:end-3);
      iq=iq+1;
    else
    % get parents
    parent1 = new_population(iq,1:end-3);

    parent2 = new_population(iq+1,1:end-3);
    p2num=iq+1;

    % cut point
    xOverPoint = ceil(rand(1,2) * (size(taskprecedence,1) - 1));
    % make one child
    xoverKids(iq,:)    = [ parent1(1:xOverPoint),parent2((xOverPoint +
1 ):
size(taskprecedence,1) ),parent1((size(taskprecedence,1)+1):(size(taskp
recedence,1))+xOverPoint),parent2((size(taskprecedence,1)+1)+xOverPoint
:end)];
    xoverKids(p2num,:) = [ parent2(1:xOverPoint),parent1((xOverPoint +
1 ):
size(taskprecedence,1) ),parent2((size(taskprecedence,1)+1):(size(taskp
recedence,1))+xOverPoint),parent1((size(taskprecedence,1)+1)+xOverPoint
:end)];    iq=iq+2;

        end
end

% mutation
for iu=1:size(xoverKids,1)
    if rand()>=mutation_rate
        xoverKids(iu,:)=xoverKids(iu,:);
    else
        mutation_point=ceil(rand() * (size(taskprecedence,1) - 1));
        xoverKids(iu,mutation_point)=ceil(rand() *
(maxreduc(mutation_point) - 1));
        xoverKids(iu,mutation_point+size(taskprecedence,1))=rand;
    end
end




problem_setting
for ixo=1:pop
population(ixo)=xoverKids(ixo);
end
population=[xoverKids;champions_plus(:,1:end-3)];
if size(population,1)==99 |size(population,1)==999
    population=[population;elite(1:end-3)];
end
initialize_va

end
tt=elite_leveling_list(find(elite_leveling_list(:,end)==min(elite_level
ing_list(:,end))),:);
its_cost=max(tt(:,end-1));
its_duration=max(tt(:,end-2));
anchor_leveling=[its_duration,its_cost,elite_leveling];
```

sharing.m

```
shared_dist=0.5;


candidate_1=candidate(1,:);
candidate_2=candidate(2,:);

result_duration=chromosome_sort(:,end-3);
plus_cost=chromosome_sort(:,end-2);

m_1=0;
for is=1:pop
```

```
   dist_1=sqrt(((candidate_duration(1)-
result_duration(is))/10)^2+((candidate_cost(1)-plus_cost(is))/1)^2);

   if dist_1==0
   sh_1=1;
   elseif dist_1<shared_dist
   sh_1=1-dist_1/shared_dist;
   else
   sh_1=0;
   end
   m_1=m_1+sh_1;
end
deg_duration_1=candidate_duration(1)/m_1;
deg_cost_1=candidate_cost(1)/m_1;
abs_dis_1=sqrt(((deg_duration_1-0)*2)^2+(deg_cost_1-0)^2);

m_2=0;
for iss=1:pop
   dist_2=sqrt(((candidate_duration(2)-
result_duration(iss))/10)^2+((candidate_cost(2)-plus_cost(iss))/1)^2);

   if dist_2==0
       sh_2=1;
   elseif dist_2<=shared_dist
   sh_2=1-dist_2/shared_dist;
   else
   sh_2=0;
   end
   m_2=m_2+sh_2;


end
deg_duration_2=candidate_duration(2)/m_2;
deg_cost_2=candidate_cost(2)/m_2;
abs_dis_2=sqrt(((deg_duration_2-0)*2)^2+(deg_cost_2-0)^2);

if abs_dis_2>=abs_dis_1
    tt=candidate_1(1:end);
    new_population(iu,:)=tt;
else
    tt=candidate_2(1:end);
    new_population(iu,:)=tt;
end
```

VITA

Name:	Kyungki Kim
Address:	Department of Civil Engineering, Texas A&M University, College Station, TX 77843-3136
Email Address:	burning_k@naver.com
Education:	B.S., Architectural Engineering, Dongguk University, 2009
	M.S., Civil Engineering, Texas A&M University, 2011