MULTI-OBJECTIVE NON-UNIT BASED REPETITIVE ACTIVITIES PROJECTS SCHEDULING USING GENETIC ALGORITHMS

A Thesis Submitted to the Faculty of Engineering in Fulfillment of the Requirements of the degree of Master of Science in Construction Engineering

Arab Academy for Science & Technology and Maritime Transport, Cairo Branch, Egypt, 2012

Prepared By

Mohamed Saeid Eid

Teaching Assistant, Construction and Building Engineering Department, College of Engineering, Arab Academy for Science & Technology and Maritime Transport, Egypt.

Supervised by:

Prof. Dr. Emad Elbeltagi

Professor of Construction Management, Faculty of Engineering, Mansoura University, Egypt.

Prof. Dr. Mohamed E. Abdelrazek

Professor of Construction Engineering and Management at the Construction and Building Engineering Department, College of Engineering, Arab Academy for Science & Technology and Maritime Transport, Egypt

Abstract

Multi-Objective Non-Unit Based Repetitive Activities Project Scheduling using Genetic Algorithms

By

Mohamed Saeid Eid

Achieving a successful construction project, one should meet decision maker various needs and objectives through a construction plan and schedule. A schedule that utilizes the use of resources to achieve minimum construction cost as well as minimum construction duration. In this study, a repetitive activities project scheduling and optimization model is developed to achieve the decision maker objectives.

The proposed scheduling model optimizes the project cost, duration, crews' interruptions and units' delivery dates delay simultaneously. The model consists of two modules; a scheduling module and an optimization module. The scheduling module takes into consideration the logical and resource start dates, different units' quantities, different production rates for assigned construction methods, as well as the transportation duration and cost of moving crews to schedule a repetitive activity project. The optimization module uses a Multi-Objective Genetic Algorithms to define a set of non-dominated solutions for the decision maker to choose from depending on the construction project conditions.

The model is implemented a computer program and introduced to several examples and case studies to evaluate its fitness through analysing the results. It was found fit and applicable on medium size repetitive activities projects.

Acknowledgments

First, I am grateful to Allah for aiding and supporting me throughout my carrier and studies, for my deeds are by him and for him to aid my brothers and sisters of humanity.

I would like to thank everybody by their names that supported me physically and mentally through the years and through this research. My family, the big loving and caring home that gives a lot and asks for few. My professor and teachers of ethics and manners, Prof. Mona Eid, Prof. Emad Elbeltagi, Prof. Mohamed Emam, Prof. Sanad, Prof. Mostafa Khalifa, Prof. Lobna Elsherif. Eng. Mohamed El-Abbassy. And many thanks to friends that helped me to excel, and special thanks to Ehab Amer for the effort that aided me a lot.

Last, but not in any means the least, I would love to thank my Country, Egypt, and my fellow countrymen, for I learnt the best in this land, and to you I present this thesis.

TABLE OF CONTENTS

List of Tables

CHAPTER 4: MODEL IMPLEMENTATION

CHAPTER 5: REAL LIFE CASE STUDY

List of Figures

CHAPTER 2: LITErature **REVIEW**

Chapter3: MULTI-OBJECTIVES REPETITIVE ACTIVITIES PROJECT SCHEDULING MODEL

CHAPTER 4: MODEL IMPLEMENTATION

CHAPTER 5: REAL LIFE CASE STUDY

Chapter 1

Introduction

"He who every morning plans the transactions of the day and follows out that plan, caries a thread that will guide him through the most busy life. But where no plan is laid, where the disposal of time is surrendered merely to the chance of incidence, chaos will soon reign." Victor Hugo – French poet & Novelist

Planning and scheduling are the essence of project management and control, without them any project manager will be lost in a sea of activities that can't be controlled. Many methods have been proposed to schedule construction projects and representing them, starting from bar charts, time-scaled diagrams to precedence diagrams and the most commonly used Critical Path Method (CPM) scheduling technique, as well as PERT.

However, due to the diversity in construction projects' types, one can't always use the same scheduling techniques to all types of construction projects. One of those types is the repetitive activities projects. The repetitive activities projects are characterized by the repetition of sets of activities through the projects' units. The repetition of activities can be linear, (e.g. pipeline, railway, highway constructions) or non-linear, (e.g. high rise building, housing compounds) (Moselhi and Hassanein 2003).

The repetition of the activities requires a scheduling technique that utilizes the resources assigned for the activities. Traditional network techniques, such as CPM, have been introduced to repetitive activities projects to schedule and control them. However, network techniques have been showing major drawbacks in repetitive activities projects (Stradel and Cacha 1982, Reda 1990, Suhail and Neale 1994, and Hegazy and Kamarah 2008). These drawbacks have encouraged the researchers to develop a number of specialized scheduling techniques for the repetitive activities projects. These scheduling techniques take into consideration several parameters:

1. Utilization of assigned resources.

2. Maintaining work continuity from one unit to another
3. Meeting projects' deadline through achieving a proper production rates for the crews.
4. Accounting for transportation of crews.
5. Different quantities for the different units.

However, most of these scheduling techniques didn't take into consideration optimizing all these factors simultaneously through one model.

1.1 Research Motivation

This research has been motivated by the lack of two aspects as follows:

- Inadequacy of Traditional Network Scheduling Techniques

  As mentioned earlier, traditional network scheduling techniques lack the ability to schedule the repetitive activities projects properly. These techniques can be implemented to limited amount of repetitive activities, but in a large scale repetitive activities project, it shows a lot of drawbacks as criticized by many researchers. The large amount of activities and their relationships makes it difficult for users to understand and control the project through visualizing it on a PDM. Also, traditional network scheduling techniques doesn't take into account the resources assigned to the activities, but uses resource management techniques that doesn't guarantee work continuity or meeting the deadline using the required production rates of activities. Moreover, traditional network scheduling techniques don't take into consideration the location of units or the transportation time of crews.

- Inefficient Optimization Techniques

  The need of optimization tools for construction projects is getting stronger as the diverse and opposing objectives of planners and project managers are increasing in an attempt to deliver the repetitive activities projects successfully. Several traditional mathematical optimization models have been introduced to the repetitive activities projects by researchers like linear programming and dynamic programming (El-Rayes and Moselhi

1998; and Moselhi and Hassanein 2003). However, the mathematical techniques don't ensure a global optimum solution for a multi-objective problem and may be trapped into local optimum solutions. Moreover, mathematical techniques don't give a set of schedules for the planner to choose from, but rather give one solution.

1.2 Research Objectives and Scope

The scope of this research is to schedule and optimize repetitive activities projects to achieve a set of optimum solutions that meets the planner requirements. The research aims to achieve the following objectives:

1- Develop a flexible resource driven model to schedule the repetitive activities projects that:
   a. Use multiple construction methods assignment strategy
   b. Determine the transportation duration and cost of crews moving from one unit to the other.
   c. Determine the duration of each unit separately depending of the quantities and assigned crew's production rate.
   d. Determine the total project cost, duration and takes into account decreasing the total crews' interruption and meeting the units' delivery dates.
2- Use a multi-objective non-traditional optimization technique, Genetic Algorithms, to determine the set of optimum schedules for a repetitive activities project.
3- Implement the scheduling and optimiztion model to a commercially wide spread and friendly interface computer program to be easily used by planners.

1.3 Research Methodology

The methodology used to achieve the research objectives involves the following:

1- Review of the recently developed repetitive activities projects' scheduling models and optimization techniques.
2- Develop a model that takes into consideration the repetitive activities projects nature, i.e. transportation of construction crews, distances

between units, different quantities in each unit …etc. Sets of feasible solutions will be created and optimized through the model to determine the start and finish dates of activities determining the project total duration and cost as well as the crew interruptions and units' delivery dates delays.

3- Apply Genetic Algorithms technique to the model to determine the set of optimum solution through optimizing the solution to achieve the multi-objective criteria needed by planners.
4- Integrate the model with a commercial project management software – MS Project – using VBA macros.
5- Apply the developed model on a case study project to validate the model in real life construction projects.

1.4 Thesis Organization

The research composed of four chapters in addition to the current one, and they are as follows:

Chapter two (Literature Review): presents a literature review of the recent repetitive activities projects' scheduling models and optimization techniques.

Chapter three (Proposed Model): presents the mathematical formulation of the proposed model and the factors affecting the scheduling model are stated and analyzed. The chapter also discusses the optimization technique used in this model.

Chapter four (Implementation and Case Study): presents the integration of the developed model into a computer program and the implementation on a validation example drawn from the literature.

Chapter five (Case Study): presents a real life case study to verify the model's effectiveness.

Chapter six (Summary, Conclusion and Recommendations): presents the thesis conclusions and discuss the recommendations for further studies on the current research.

Chapter Two

Literature Review

2.1 Introduction

Construction projects always face several challenges; completing the project on time, keeping the project expenses to minimum, increase the utilization of resources…etc. However these challenges increase with repetitive activities projects.

Repetitive activities projects are those projects where a set of activities are repeated through the whole project. Repetition can be due to geometric and location layouts or due to multiplications of units. Repetitive activities projects can be classified into two main categories: linear projects, such as pipe lines, highway, and railways, and nonlinear such as: multiple housing and high rise buildings (Moselhi and Hassanein 2003).

Repetition of activities requires from the project managers and planners to find the optimum plan and schedule that meets the project objectives through optimum utilization of their available resources.

Through this chapter, a discussion is made on the use of commercially wide spread critical path method (CPM) technique and precedence diagram method representation (PDM) in the repetitive activities projects and its limitation. Additionally, a review of the latest researches on the scheduling techniques for the repetitive activities projects is presented.

2.2 Network Techniques (CPM/PDM)

Network based methods, such as the critical path method (CPM), are proven to be powerful scheduling and progress control technique (Arditi et al. 2002). The critical path method have been used in the construction projects for decades, and have been proven to be easy to apply to most of the construction projects types. It is now widely known with the help of commercial application and computer programs that uses the CPM technique (Mattila and Park 2003). In addition,

aggressive marketing by CPM software developers has also helped CPM dominate the market (Duffy et. al 2011).

While CPM has been used on countless projects, it has been found inadequate when scheduling repetitive activities projects (Mattila and Park 2003). It has been reported that the CPM based on the network diagrams applied on the repetitive activities projects has many drawbacks. It was observed the inability of CPM to model the repetitive nature of linear construction (Stradel and Cacha 1982, Reda 1990, Cole 1991, Rahbar and Rowings 1992, Suhail and Neal 1994, and Harmelink 1995).

The CPM technique is a duration oriented technique that doesn't take into consideration the resources as an input drive the project schedule. A scheduling technique, as such, does not maintain the work continuity of resources. Maintaining the work continuity in a repetitive activities project is highly significant for the planner on moving the resources from one unit to the other without creating idle time to optimize the project schedule. Maintaining work continuity decreases the project total cost, avoiding idle time costs, as well decreasing the total project duration on keeping all the resources working to achieve the least project duration. Some limitations have been identified for CPM technique when scheduling continuous projects regarding the difficulty to maintain continuity in crew utilization (Yamin and Harmelink 2001). This makes the CPM scheduling techniques less effective for repetitive activities construction projects (Huang and Sun 2006). To maintain work continuity, repetitive units must be scheduled in such a way as to enable timely movement of crews from one unit to the next, avoiding crew idle time. (Ammar and Elbeltagi 2001).

Moreover, CPM technique doesn't takes into consideration the required output rate of each activity type to meet the delivery dates of units since it is duration oriented technique. CPM technique schedules the activities of each unit as soon as possible dates (early start dates) based only on the logical relationships with preceding activities (Wassef and Hegazy 2001) regardless of the desired production rates required for each activity type or the resources assigned to the

activity. Thus may fail the project manager to deliver the units on time and might create idle time for the resources.

In addition, CPM technique uses resource management techniques - after scheduling - to manage the resources assigned to activities with the priorities only given to the critical activities. This will result into variation delivery dates of project units as well as creating idle time for resources that are not on the critical path. Resource management with CPM is commonly done by plotting resource usage per day in a bar chart diagram. This graph must be viewed together with the CPM network to understand how moving resources from one activity could affect other activities (Yamin and Harmelink 2001). In addition, such a technique initially assume unlimited availability of resources in the development if a project schedule and through resource allocation require revision of the project schedule to comply with resource availability (El-Rayes and Moselhi 1998).

CPM doesn’t have any consideration for location of work in schedule (Hegazy and Kamarah 2008). Location of units are key items in repetitive activities projects, as it aids the planner to determine the transportation duration and cost of crews undertaking the units.

Also, considering that the CPM technique is usually applied to precedence diagram method PDM, it shows a great drawback in representing the repetitive activities projects. Repetitive activities project may consists of thousands of activities and representing them and their relationships using PDM technique makes it difficult for users and planners to visualize projects’ schedules and manage the work in progress as shown in Fig. 2.1.

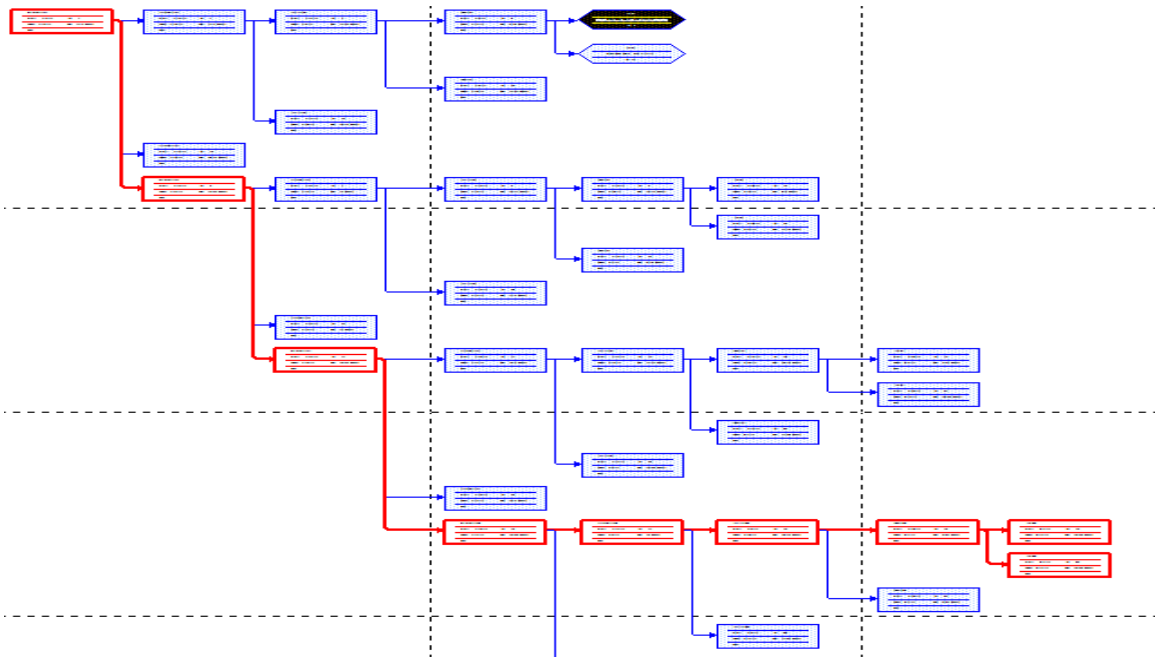


Fig. 2.1: Network Technique Representation for Repetitive Activities Project

On recognizing the drawbacks of applying the CPM technique to repetitive activities projects, a number of resource driven techniques have been developed to overcome these drawbacks taking into account the repetitive activities projects special nature. Some of these techniques are presented in the next sections.

2.3 Linear Scheduling Technique

A linear schedule is a visual representation for a repetitive activities project's plan. It shows the plan's logic and the relationships between activities. The schedule is displayed on a time-location diagram, with time on one axis and location on the other axis. Time and location can be on either axis, depending on which makes more sense to the construction project type. For a high-rise building, putting location on the vertical axis coincides with the building rising from floor to floor. For a highway project, putting location on the horizontal axis coincides with the dimensional nature of the project (Mattila and Park 2003). Fig. 2.2 illustrates an example of a linear schedule for a rural highway project.

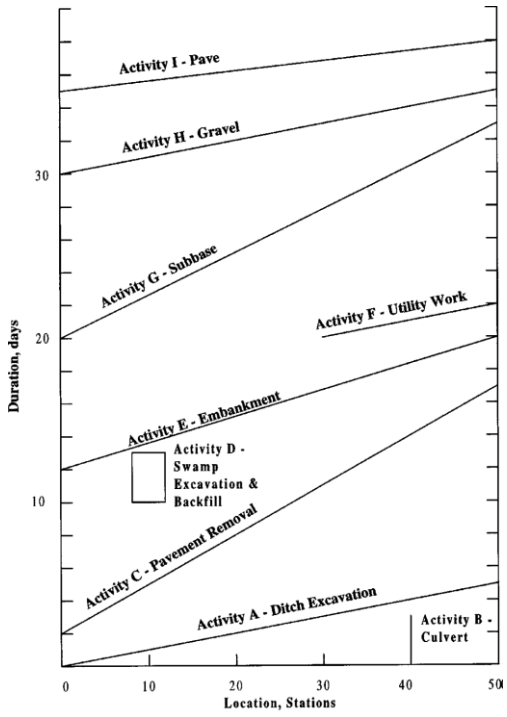


Fig. 2.2: Linear Schedule Example (Mattila 1997)

Vorester et al. (1992) suggested that there are three types of activities that can appear in linear schedule: linear, block and bar. However, Harmelink and Rowings (1998) refined the linear activity types into the following six specific subtypes:

1. Continuous full-span linear (CFL)
2. Intermittent full-span linear (IFL)
3. Continuous partial-span linear (CPL)
4. Intermittent partial-span linear (IPL)
5. Full-span block (FB)
6. Partial-span black (PB)

Linear scheduling has long been regarded as a technique that provides significant advantages when applied to linear construction (Johnson 1981). LSM is also very easy to understand, and it can be used at every level of the repetitive activities construction projects (Yamin and Harmelink 2001). However, it has been viewed essentially as a graphical technique that is not as easily adaptable to computerization as network models (Chrzanowski and Johnston 1988). At the heart of network model-based scheduling methods is the ability to determine the

critical path. This path identifies those activities that, if their duration changes, the duration of the entire project changes. For linear scheduling to be accepted as a viable tool in project planning and management, it also must be able to determine a set of controlling activities (Harmelink and Rowings 1998).

Harmelink and Rowings (1998) proposed an analytical basis for formulation of computer-based linear scheduling algorithms through determination of the controlling activity path from a linear schedule. The procedure to determine the controlling activity path in a linear schedule involves the following three steps:

1) Activity sequence list:

The activity sequence list identifies all of the possible logical sequences through the activities on a linear schedule. The controlling activity path is defined as the continuous path of longest duration through the project and defines the sequence of activities that must be completed as planned to finish the project within the overall planned duration. The activity sequence with the longest duration (or the least free time) contains all of the activities on the controlling activity path.

2) Upwards pass

The goal of the upward pass is to determine which activities or portions of activities could potentially be controlling. The process starts with the beginning of the project and progresses upward, identifying the path with the least free time between each pair of continuous full-span activities. The potential controlling segment of the origin activity can be determined by examining the relationship between these two activities.

3) Backward pass

The purpose of the backward pass is to determine which portions of the potential controlling segments are actually on the controlling activity path. This step can be compared with the backward pass used in CPM scheduling, which identifies activities that do not have any float.

Ammar and Elbeltagi (2001) introduced an effective scheduling algorithm for linear and repetitive projects using CPM technique for the scheduling of

activities considering logic constraints while satisfying resource continuity and to define the controlling path. The model benefits from an earlier analysis by Harris and Ioannou (1998) to determine the controlling path. The model assumes a constant production rate along the different repetitive units of each activity and calculated as follows:

$$r_i = 1 / d_i \quad (2.1)$$

Where r_i and d_i denote production rate and duration of activity i respectively. The model also assumes that only the most significant resource will be considered and initially requires the following data:

- A precedence network for a typical unit.
- Unit duration for each activity in the network (d).
- The number of repetitive units (n).

The algorithm is carried out by three steps:

- Step 1: Specifying relationship type:

  This step specifies the relationship type among different activities, which is considered the most important aspect to maintain the resource continuity usage. To specify the relationship between two consecutive activities, the production rate of each activity is compared with that of its successors. The production rate of the successor can be one of the following cases:

  - Greater than the current activity's production rate, thus will create a Start-to-Start (SS) relationship with a lag time equals to the current activity's duration.
  - Less than the current activity's production rate, thus will create a Finish-to-Finish (FF) relationship with a lag time equals to the successor activity's duration.
  - Equal to the current activity's production rate. In this case, any of the above cases can be used with a FF or SS relationship with consideration to the lag time.

- Step 2: Network scheduling

  After obtaining the data from the previous step, network calculations similar to that of CPM technique are done. Forward path calculations are done to determine the early times of each activity, while the backward path determines the late times. Also the critical activities are specified. The critical activities will be checked in Step 3 to specify logic and resource critical units.

- Ste 3: Determining controlling path

  After timing all activities are determined, the critical units (logic and resource) are specified based on the activities' production rates. The production rate of each activity is compared with that of its preceding and succeeding activities and by applying a set of rules the controlling path can be easily determined.

2.4 Line of Balance Scheduling Technique

Line of balance (LOB) is one of the earliest scheduling techniques created for repetitive activities projects. It was first used by the US Navy in early 1940's as a graphical scheduling technique to control the production of warships. Several researchers have been attempted to implement it in the construction industry.

Line of balance is a graphical resource oriented scheduling technique that takes into consideration the special nature of the repetitive activities projects, respecting the units' locations, usage of multiple crews and shows the production output rate of each activity in the project. The LOB chart also shows the crew movement from one unit to other.

Line of balance scheduling chart consists of two axes, on the horizontal axis the duration of the project, and on the vertical axis the units repeated through the project. Each activity repeated through the units - using one or more crew - is connected together forming the activity line as shown in Fig. 2.3.

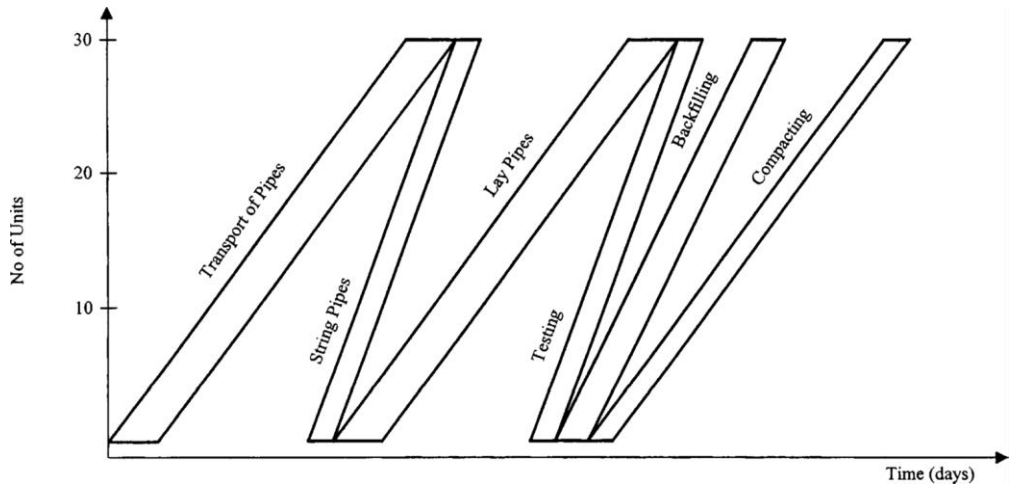


Fig. 2.3: Line of Balance Representation (Arditi et al. 2002)

Lumsden (1968) incorporated the network logic by linearly propagating the CPM start and finish dates for the activities of the first repetitive unit with a rate that would achieve the project completion date. The schedule is obtained through three steps:

1. Calculate the start and finish dates of the first repetitive unit using the CPM technique.
2. Calculate the required progress rate based on the competition date of the first unit and project deadline as given in Eq. (2.2);

$$R = \text{units} - 1 / (\text{Deadline-CPM Duration}) \quad (2.2)$$

3. Apply the same progress rate to all activities.

To achieve the required progress rate for each activity through the different units, it is required to determine the number of crews needed to be assigned to these activities. The number of crews (*C*) is determined by multiplying the progress rate (*R*) by the units' duration (*d*) as presented in Eq. 2.3;

$$C = R \text{ X } d \quad (2.3)$$

However, on calculating the number of crews required for each activity type, the number might needs to be rounded up and thus creating a new progress rate that may interfere with the project required rates. Lumsden suggests that each project has a "Natural Rhythm, and that any deviation from that rhythm results in a less than full utilization of resources. Also, in the previous Eq. (2.2), it was assumed

that the quantities through the units are the same and the duration would be the same, while in practice, the quantities in each repetitive activity is not the same and the production rate of each crew is not equal, and this assumption limits the applicability of the LOB method (Huang and Sun 2006).

Moselhi and Hassanein (2003) presented a model to optimize repetitive activities projects and overcoming the regular LOB drawbacks by employing a resource driven and traditional network scheduling techniques. The proposed model calculates quantities of each unit and the duration corresponding to the assigned crew's production rate.

The proposed model is flexible with the number of predecessors of each activity and their relationships. As well as it calculates the transportation duration and cost of each crew moving from one unit to the other.

The proposed model consists of four main steps:

1. Dividing the project into sections based on the location of units and the possible start time of each unit.
2. Determining the quantities of each section.
3. Determining the optimum crew to be assigned to this location based on a series of pre-entered indices.
4. Calculation of the section duration.

The model is said to be very efficient, yet it's a single objective model, and deals with the multi-objective requirements as a function in the duration criteria, which may give local optimum solutions.

Hegazy et al. (2004) developed a LOB model for scheduling infrastructure projects. The model created in this work used Genetic Algorithms as an optimization tool. The model allows for up to three construction methods, each consists of different crew, material and sub-contractors formation.

The model allows for site order change, which can improve the overall project duration concerning the transportation time, as well as decreasing the crews'

interruptions to increase production rate. It takes into consideration the seasonal changes and its effects on the production rate of the crews at each construction site.

However, the model objective function is to minimize the total construction cost, thus it's a single objective model, even though the total construction cost includes direct cost, indirect cost, liquidated damage, incentive for early completion and crew moving costs but it neglected the project total duration as a main objective. This may give a local optimum or a vague output and will not give the project managers and planners the flexibility of choosing a plan that meets their project's requirements.

Hyari and El-Rayes (2006) introduced a model based on linear scheduling technique (LSM) and Genetic Algorithms as an optimization tool to find the optimum set of plans in a tradeoff between total project duration and work continuity. The model used the date entered by the user to determine the logical relationship between repetitive activities, the quantities of work for each unit and the available crew formations and their corresponding production rates.

The model consists of four main steps:

1. Calculating the duration of the activity depending on the quantities of the unit and the production rate of the crew.
2. Determination of the earliest possible start time for a crew to be assigned to a unit.
3. Determination of the activities earliest possible start time depending on the crew start time and logical start time.
4. Determination of project total duration and project total interruptions

The model can be easily implemented to most of the repetitive activities projects, however it didn't consider transportation duration of crews from one unit to the other. Also, due to the research scope, the model doesn't represent all the parameters a project manager requires like cost, delivery dates…etc.

Huang and Sun (2006) introduced a practical non-unit based repetitive activities scheduling model, as it is very rare to have identical repetitive units in a repetitive activities project. They organized the project into activity groups and resources to be assigned to each group.

Huang and Sun identified, as well, the need to reconsider the hard relationship between activities in the same activity group. For example, there shouldn't be any hard relationship between foundations activities in different spans, the relationship should be more generalized to give a more realistic schedule. This means that in an activity group there's no specific order to undertake them.

The proposed model also allows assigning more than one resource to the activity group to carry out different units in the activity group. Moreover, the model maintains the work continuity of the resources assigned to decrease the idle time of resources. The model as well takes into consideration the mobilization and demobilization of crews moving from one unit to another unit.

The scheduling model is carried out in three steps:

1. Identify activity groups
2. Development of resources chains
3. Position resources chains for project scheduling.

However, they considered only single objective for the model, accounting only for total project duration and neglecting other objectives that may change the optimum solution.

Ipsilandis (2007) presented a multi-objective programming model for repetitive linear projects. Defining the complex nature of linear repetitive projects and the various objectives a project manager needs for his project. Ipsilandis defined the five objectives needed for this type of construction projects to minimize total project duration, total work-break time, unit completion time, total cost of work-break time, and delay cost in unit completion.

Ipsilandis also defined some general parameters that are common for repetitive linear projects: all tasks are performed in all units; set of dependencies are fixed in all the units, and a task can't be performed in any project unit before the same task is completed in the previous unit. This means that the model doesn't take into consideration assigning more than one resource to more than one unit in an activity group. In addition, the model solves the multi-objective criteria in a single objective cost function.

Elbeltagi and ElKassas (2008) developed a cost optimization scheduling model for repetitive activities projects. The model consists of two modules; a scheduling module and a cost optimization module. the model is capable to (1) schedule repetitive activities projects with typical and non-typical repetitive activities, (2) calculate the total project cost including direct, indirect and interruption costs and (3) generate optimum or near optimum project schedule.

In their work, the scheduling module consists of two stages; an initial stage and a refinement stage. In the initial stage, the scheduling module creates a schedule that respects the logical start time of the activity from the relationship with other activities, and the start time of the earliest possible available crew. After scheduling, it calculates the interruptions of each activity. In the refinement stage, the scheduling module shifts the activities with crew interruptions to minimize the interruptions in an attempt to maximize the work continuity. The cost optimization module used the genetic algorithms to determine a near optimum solution for the project schedule. The variables for the genetic algorithm optimization module are the construction indices assigned to the activities.

One notable effort was developed by Duffy et al. (2011) in creating a linear scheduling technique that takes into consideration the production rate variability based on working windows. The proposed model is an addition to a previous research proposed by Yamin and Harmelink (2001).

Understanding that construction crews have variable production rates depending on the changing conditions of the location of units, an adjustment on the duration

of each unit should be calculated depending on the activity performance index in the corresponding working window.

Working windows are a grid approach to organize the project units based on their locations and time. Each working window has its variables that affect the production rate of the activities. Duffy et al. (2011) classified the variables that affect the production rate into four categories:

- General variables; that are not related to either time or location. For example; number of workers, construction methods.
- Time variables
- Location variables
- Time-location variables: like weather, and site conditions.

Determining the activity performance index depending on the working window variables, the model can not obtain the actual production rate of a crew assigned to an activity as well as determining the time remaining and distance traveled for each activity in the grid.

The model is very easy and can be implanted in commercial software as it is easy for the planners to use. However, it optimizes only a single objective function. Also it only used production rates with linear quantities only (m and feet), while there are other production rates that will give an error or illogic calculations like m^3 and kg etc… Moreover, not all crews for each activity are affected with the same variables (location, time, general or time-location variables) in the same way.

2.5 Genetic Algorithm

Through the researches made in the last decade, it was found the need of an optimization tool to deal with the complex, diverse and conflicting variables and objectives. Different tools are used to find the optimum solutions for the repetitive activities project scheduling. These tools varied from dynamic programming to the use of Artificial Intelligence (AI) models. With the recent implementations of AI in construction management problems, Genetic

Algorithms (GAs) showed great efficiency in searching for global optimum solutions for the complex problems. (Li and Love 1997, Feng et. al. 1997, Hegazy and Ayed 1999, Hyari and El-Rayes 2006, Elbeltagi et. al. 2005).

Genetic Algorithms (GAs) are inspired by biological systems' improved fitness through evolution (Holland 1975). GAs form a set of random solutions that search the solution space for the optimum set of solutions through evaluating the solutions depending on their fitness.

2.5.1 GAs structure

Genetic Algorithm is a metaheuristic that simulates Darwin's theory of evolution and the survival of the fittest. The solutions in GA are subject to evolution like in nature through crossover of inherited genes and mutation. These solutions are called chromosomes, and each chromosome consists of numbers of genes which carries the values of the problem's decision variables. Genes' values can be binary or real number depending on the problem at hand. The chromosomes' length is equal to the number of decision variables in the problem. (Elbeltagi et al. 2005).

Each chromosome is evaluated to calculate its fitness depending on the objective function(s). Good chromosomes are the ones that have high fitness value in case of maximization problems, or low fitness value in case of minimization problems. Those good chromosomes (solutions) have a higher probability to create offspring chromosomes that may have better fitness.

The number of chromosomes generated represents the population size. The population size usually affects both the run time and the precision of the solution. It is determined by trial and errors. A whole complete cycle of creation of chromosomes, their evaluation against the fitness function and finally the selection of the fittest, is referred to as one generation (iteration). The number of generations also affects both the run time and the precision of the solution, and likewise the population size. The appropriate number of generation is determined experimentally (Elbeltagi et al. 2005).

The first generation of chromosome yield a number of offspring generation. Only good ones are selected and the rest are discarded. The offspring generation then undergoes reproduction, crossover and mutation operations in an attempt of enhancing the solution. This process continues until the termination condition satisfied as shown in Fig. 2.4. (Elbeltagi et al. 2005).

Selection utilizes roulette wheel to contain all individuals in the population and their slots in the wheel are sized in proportion to their relative fitness values. When each time a decision has to be made, a simple spin of the wheel yields the selected candidate. In this way, the random nature of GAs is maintained and chromosomes with a high degree of fitness can still have a higher chance to survive in succeeding generations. Once the selection is made, an exact replica of the string is entered into a mating pooling, waiting for further genetic operations (Elbeltagi et al. 2005).

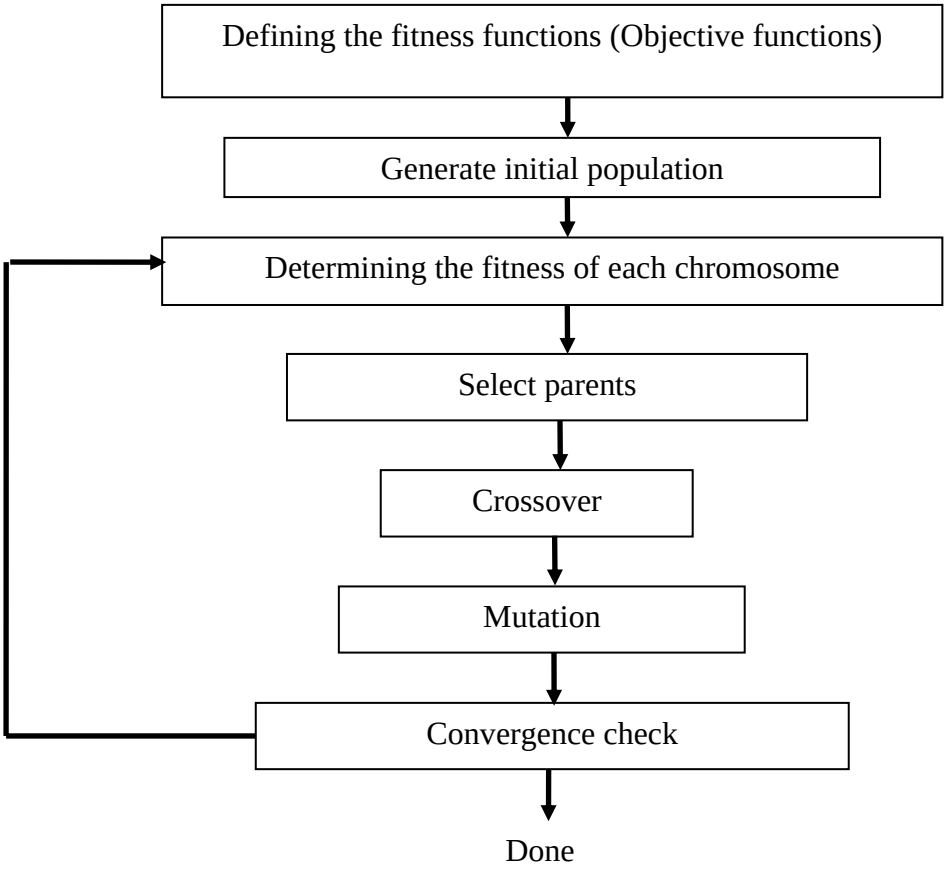


Fig 2.4: Flowchart of GAs Procedure

2.5.2 Genetic operations

Crossover is the main genetic operator in GAs which passes the genes from two parents chromosomes to two new offspring chromosomes mimicking marriage in nature. Randomly choosing the parents as the best of the population will most likely yield to have a better offspring through crossover. The randomizing of selecting the parents chromosomes is guided by their fitness; the fittest chromosomes are more likely to be selected as parents for the next population. However, not all the chromosomes in the population will be subjected to crossover operation; this is controlled by a crossover probability (P_c). This crossover probability defines the number of chromosomes that will undergo crossover (population size X P_c), and the priority is given to the most fit parents chromosomes. However, the more the crossover probability index is the more the time it will take to complete the computations, which might be wasted in discovering undesired solution space. In the other hand, the less the crossover probability index is the more probability the solution will be trapped into local optimum region (Elbeltagi et al. 2005).

Mutation is another genetic operation that helps avoiding local optimum solutions by suddenly and randomly altering the genes value between its upper and lower bounds in some of the chromosomes. Although mutation can be considered a secondary GAs operation, it is essential to ensure discovering more into the solution space. Moreover, mutation prevents premature convergence by increasing the population diversity (Elbeltagi et al. 2005).

Like the crossover process, only a number of chromosomes will undergo the mutation process, these chromosomes are selected depending to the mutation probability (Pm). Thus, the number of chromosomes that will undergo mutation is equal to the multiplication of the population size by the mutation probability (population size X P_m). However, the more the mutation probability index is the more computation time it will take the processor, and also the more likely the process yield into losing vital and fit solutions. In the other hand, the less the mutation probability index, the more likely the

problem will face a premature convergence and local optimum solutions (Elbeltagi et al. 2005). Fig. 2.4 illustrates the overall GA's operations, while Fig. 2.5 illustrates the crossover and mutation process.

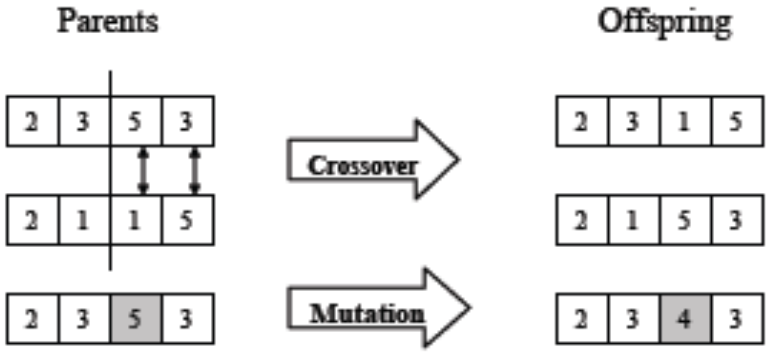


Figure 2.5: Crossover and Mutation

2.6 Multi-Objective Optimization

Single objective problems optimization can easily be achieved through one maximum/minimum objective function. However, for a multi-objective optimization problem, the functions can be conflicting and diverse. Thus, there is no single optimum solution, and instead there are a set of optimum solutions for planner to choose from. Decision makers desire solutions that simultaneously optimize multiple objectives and obtain an acceptable trade-off amongst objectives (Dehuri and Cho 2009).

Conventional mathematical optimization techniques such as dynamic programming and linear programming are not suitable for solving a multi-objective problem. These techniques solve an optimization problem by one point (one answer). Also, conventional mathematical techniques can't discover a disconnected feasible region and would be trapped in a local optimum solution.

To solve a multi-objective optimization problem, different methods have been used such as; simple aggregation, weighted aggregation and Pareto optimal solution set.

2.6.1 Simple aggregation

Simple aggregation methods converts a mutli-objective problem to a single objective one by aggregating all objective in a single objective function as shown in Eq. (2.4)

$$\text{Min (max): } F = f_1 + f_2 + \ldots.. + f_n \tag{2.4}$$

Optimization using simple aggregation gives a single optimal solution. However, the relative importance of the objectives is not considered in this approach. Beside this, one of the objective functions may be, especially if its value is large, the dominant in the over objective function and the smallest objectives may be ignored (Elbeltagi et al. 2010).

2.6.2 Weighted aggregation

Weighted aggregation is a modified simple aggregation method, where a multi-objective problem is transformed into a single objective function by applying a function operator to the objective vector (Baumgartner et al. 2004). These functions are designed by the decision maker to achieve his/her preferences. A linear combination of the objective functions sample is shown below:

$$\text{Min (max): } F = w_1 f_1 + w_2 f_2 + \ldots. + w_n f_n \tag{2.5}$$

Where the weights (w_i) indicate the relative importance of the function vector to the decision maker and the sum of weights are equal to unity. The values of the weights are problematic, and can't be determined without prior information and parameters of the problem at hand.

This method is simple, and achieves an optimum solution for the planner with much less programming effort. Also, achieve the solution that is more likely to be preferred to the planner if the weights are entered properly.

However, both of the above mentioned multi-objective optimization methods have several drawbacks (Elbeltagi et al. 2010):

- Only a single optimal solution can be obtained due to the use of a single objective function.
- Trade-offs between objectives can't be evaluated in aggregation objective functions
- For weighted aggregation, the solution is highly dependent on the weights, and any lack of information may lead to improper weights and undesirable optimal solution.
- If the feasible region is discontinuous, these methods will be trapped in local optimum solution.
- Objectives are usually of different measuring units which may give a vague solution.

2.6.3 Pareto Front

The Pareto Front concept is to find a set of optimum solutions so the decision maker can choose from the most desirable. A solution belongs to a Pareto set (set of non-dominated solutions) if there is no other solution that can improve at least one of the objectives without degradation of any other objective.

Using Pareto Front sets has several advantages; (1) it gives the decision maker a set of optimum and desirable solutions to choose from, (2) unlike the other methods, it doesn't ignore the trade-off between objective functions, (3) it discovers more of the solution space with respect to the objective functions which avoids local optimum solutions, and (4) it can discover a discontinuous feasible solution space.

Figure 2.6 shows the concept of Pareto optimality considering two minimization objective functions; duration and cost. All the solutions in the feasible region space - marked by dashed line - satisfy the problem constraints. But only the Pareto optimum solutions are on the Pareto Front

marked by solid line in the left lower corner. The set of Pareto optimal solutions are called Pareto Front.

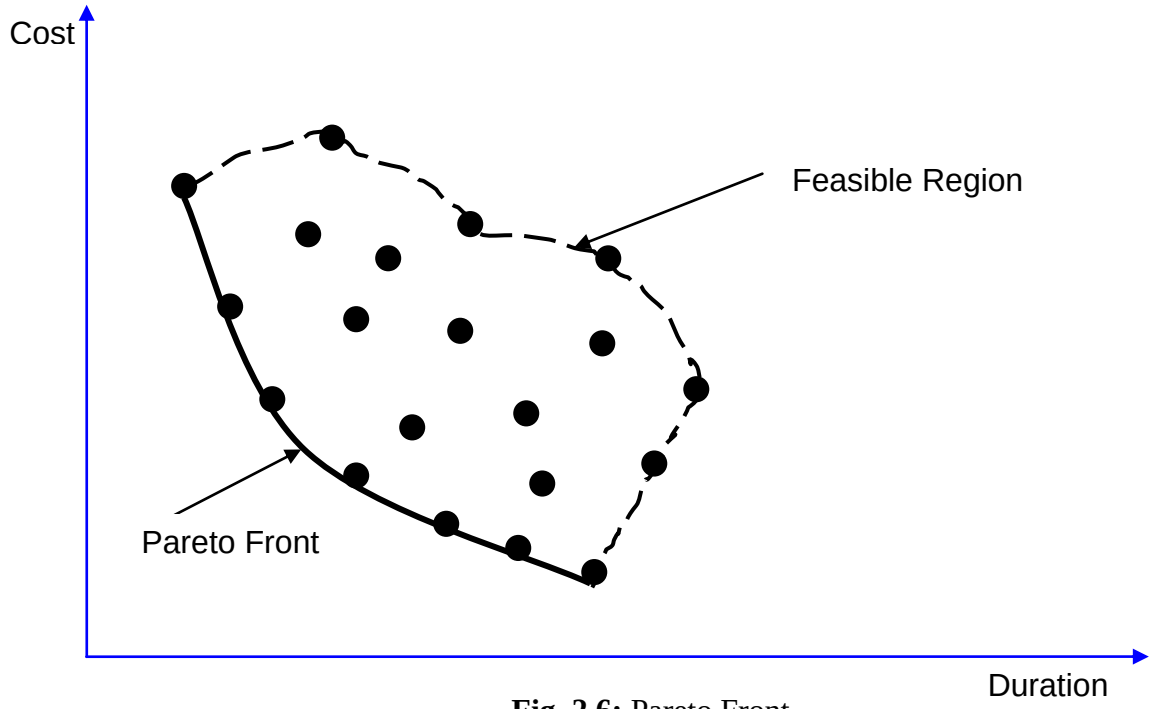


Fig. 2.6: Pareto Front

To measure the fitness of solutions in any GAs iteration, the Pareto Front Sorting may be used efficiently (Fig. 2.7). Pareto Front sorting sorts different feasible solutions into fronts that determine their ranks. In Pareto Front sorting, the set of non-dominated solutions defining the first Pareto Front is identified and assigned a rank of unity. This set is then isolated and the other solutions are compared to determine their non-dominated solutions and this new set is ranked by two. The sorting process is repeated until the entire population is ranked.

The rank obtained from Pareto Front sorting helps to determine the fitness of the solution. The fitness of each solution *i* is calculated by Eq. (2.6) (Elbeltagi et al. 2010).

$$\text{Fitness}_i = 1 / \text{rank}_i \qquad (2.6)$$

Where $fitness_i$ and $rank_i$ are the new fitness value and rank number for the solution *i*.

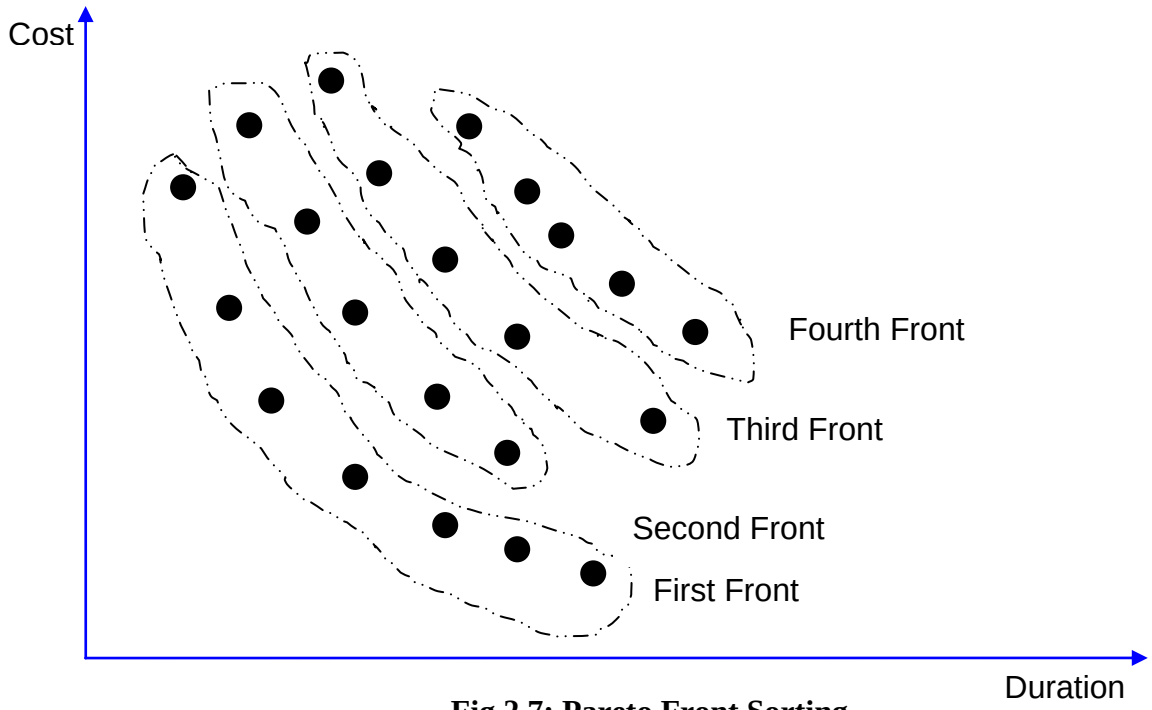


Fig 2.7: Pareto Front Sorting

2.7 Summery and Conclusions

This chapter presented a review of the repetitive activities projects and their need for a scheduling technique that suits their nature. The use of traditional network techniques has been presented and their drawbacks have been annotated in this chapter. The use of line of balance scheduling technique is also presented along with some of the latest researches made in this area to optimize the use of line of balance combined with CPM technique. The use of genetic algorithms has been noticed to be used efficiently in the optimization of the repetitive activities project scheduling. However, the usages of multi-objective evaluation of solutions have been needed.

CHAPTER 3

MULTI-OBJECTIVIES REPTIVITE ACTIVITIES PROJECT SCHEDULING MODEL DEVELOPMENT

3.1 Introduction

This chapter presents the development of a multi-objective scheduling model for repetitive projects. The proposed model consists of two modules; scheduling and optimization modules. In this chapter, an explanation of the proposed model is presented. The mathematical formulation and logic of the scheduling module, as well as the optimization module are explained. The objectives of the model and factors are also shown.

3.2 Model Overview

The proposed model comprises of a scheduling and optimization modules that takes into consideration several factors that affect the schedule. These factors are; assigned construction method, duration, cost, work continuity and delivery dates of units. On meeting the delivery dates of units, the total project duration might decrease, however it might as well create interruptions and affect the work continuity as well as increasing the cost. Also, decreasing project cost might produce a slow progress rate and increase the total project duration. In an attempt to overcome such problems, a new model is proposed to assign appropriate construction methods to different activities to achieve the following objectives:

- Minimization of total project duration.
- Minimization of total project cost.
- Minimization of total project interruptions (maximum work continuity).
- Minimization of units' delivery dates delays.

The scheduling module of the proposed model is a resource driven module that develops a schedule for a repetitive activity project while respecting logical

relationship constraints as well as other practical factors taken into consideration as follow:

- The ability to have different quantities to be undertaken at each unit for the same activity.
- Each activity type can have different construction methods; each construction method has its own production rate and direct cost.
- The ability to start two or more units of the same activity type at the same time using different construction methods.

The optimization module utilizes a multi-objective genetic algorithm through assigning the different construction methods to the project activities.

3.3 Model Development

The proposed model, as mentioned before, consists of scheduling and optimization module. In the scheduling model, each activity (*i*) can have different number of construction methods (*m*) that can be assigned to the activity in any unit (*j*). These construction methods are associated with different direct costs and production rates, thus creating different activity durations when assigning them to different units. Accordingly, by assigning different combination of construction methods to different activities' units, different schedules are developed.

3.3.1 Scheduling Module

The scheduling module consists of four coherent stages that aim to create a schedule depending on the assigned crews' to the different activities in each unit. The scheduling module also calculates:

- The total project duration
- The total project cost
- The total project crews' interruptions
- The total units' delivery delays

1) calculating the activities' scheduling dates

This step determines the activity's start ($S_{i,j}$) and finish ($F_{i,j}$) dates depending on the assigned construction method and the logical relationship with preceding activities.

The start date ($S_{i,j}$) is obtained using Eq. 3.1 by determining the latest of both logical relationship start date ($SL_{i,j}$), calculated from a regular CPM calculations in Eq. 3.2, and the earliest possible start date of the crew ($SC_{i,j}$).

$$S_{i,j} = \text{Max}\,[SL_{i,j}\,,\, SC_{ij}\,] \tag{3.1}$$

$$SL_{i,j} = F_{i-1,\,j} \pm \text{lag} \tag{3.2}$$

In figure 3.1, two activities are carried out using two crews. Crew one is assigned for activity two in the third unit. The start time of this activity ($S_{2,3}$) will be the latest of the predecessor finish date, or the logical start time ($SL_{2,3}$) and the crew start time ($SC_{2,3}$). Although the preceding has finished early, the activity will not start until the assigned crew is available to work on the current activity.

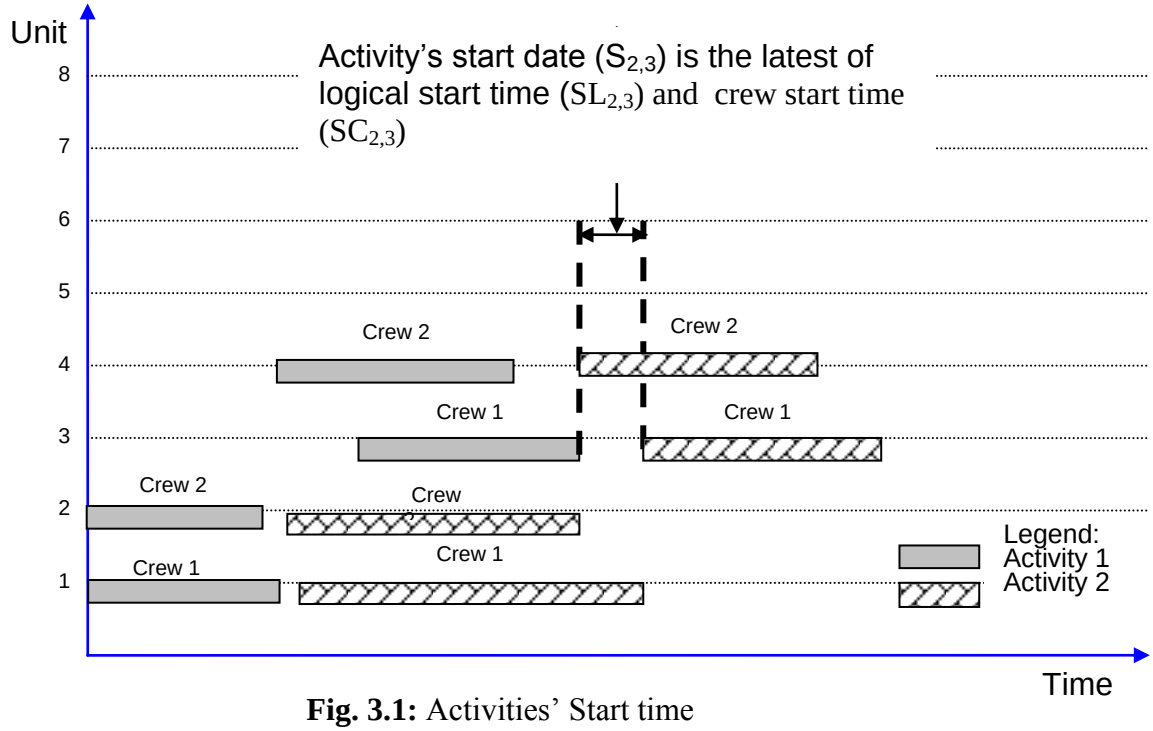


Fig. 3.1: Activities' Start time

The crew's earliest possible start ($SC_{i,j}$) is obtained through determining the previous unit (PU) finish date ($F_{i,PU}$) that the crew have undertook, in addition to its corresponding transportation duration ($TD_{PU,j}$) from the previous unit (PU) to the current one, the crew's earliest possible start date can be obtained using Eq. 3.3.

$$SC_{i,j} = F_{i,PU} + TDm_{Pu,j} \quad (3.3)$$

Depending on the construction method, assigned to activity, production rate and the quantities of work to be undertaken, the duration of the activity can be calculate using Eq. 3.4.

$$D_{i,j} = Q_{i,j} / P_{m,i} \quad (3.4)$$

Where $D_{i,j}$ is the duration of activity (i) in unit (j). $Q_{i,j}$, is the quantity of work of activity (i) in unit (j), $P_{m,i}$ is the production rate for construction method crew (m) that can be assigned to activity (i).

The finish date is calculated by adding the duration of the activity to its start date as shown in Eq. 3.5.

$$F_{i,j} = [S_{i,j} + D_{i,j}] \quad (3.5)$$

2) detection of the previous unit

Determination of the previous unit that the construction method's crew has been working at before the current one aids the module to determine the transportation duration (TD) and cost (TC) for the assigned crew. This step objective is to determine the previous unit (PU), if it exists.

Detection of the previous unit is obtained through the following two steps:

a. determining if the assigned crew have been working at any unit before the current one using a Crew Start Check index (CSC_m)

b. If the previous step equal "Started" then by checking backward through the preceding units determining their finish dates ($F_{i,j}$) and compare them with the current unit's crew's earliest possible start

date ($SC_{i,j}$), if the two dates are equal then the preceding unit will be the previous unit (*PU*) that the crew has been working at.

These steps are illustrated in a flow chart in the following Fig. 3.2.

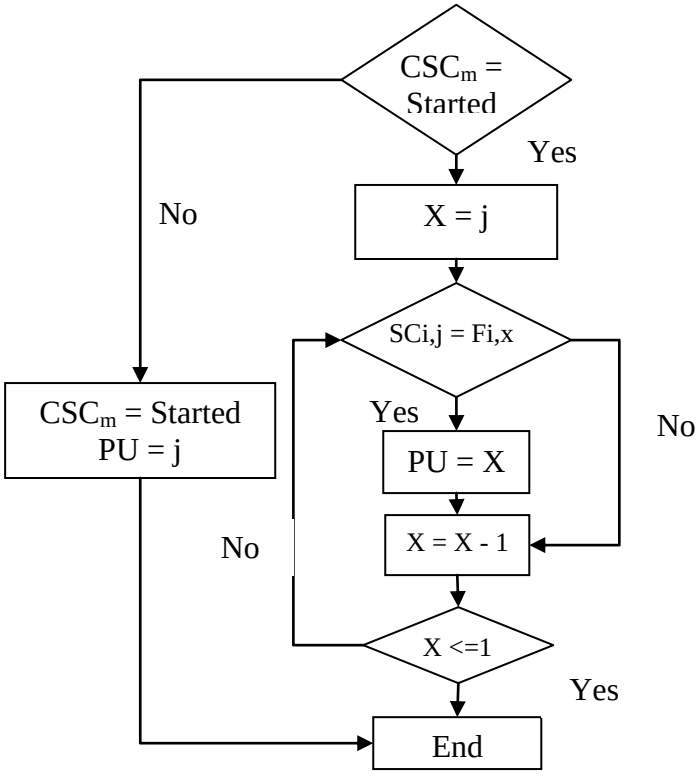


Fig. 3.2: Determination of Previous Unit (PU)

3) Transportation duration and cost

Transportation from one unit to the other is a crucial element in scheduling a repetitive activity project. Thus, it must be calculated according to the assigned construction method's crew type (*m*) and the previous unit (*PU*) it worked at before, for each two units have different distances, so there would be different transportation duration (*TD*) and cost (*TC*).

The transportation duration (Fig. 3.2) of any crew is obtained through Eq. 3.6 by dividing the distance travelled between the two units by the average speed of the crew.

$$TD_{m\ PU,j} = DS_{PU,j} / SP_m \quad (3.6)$$

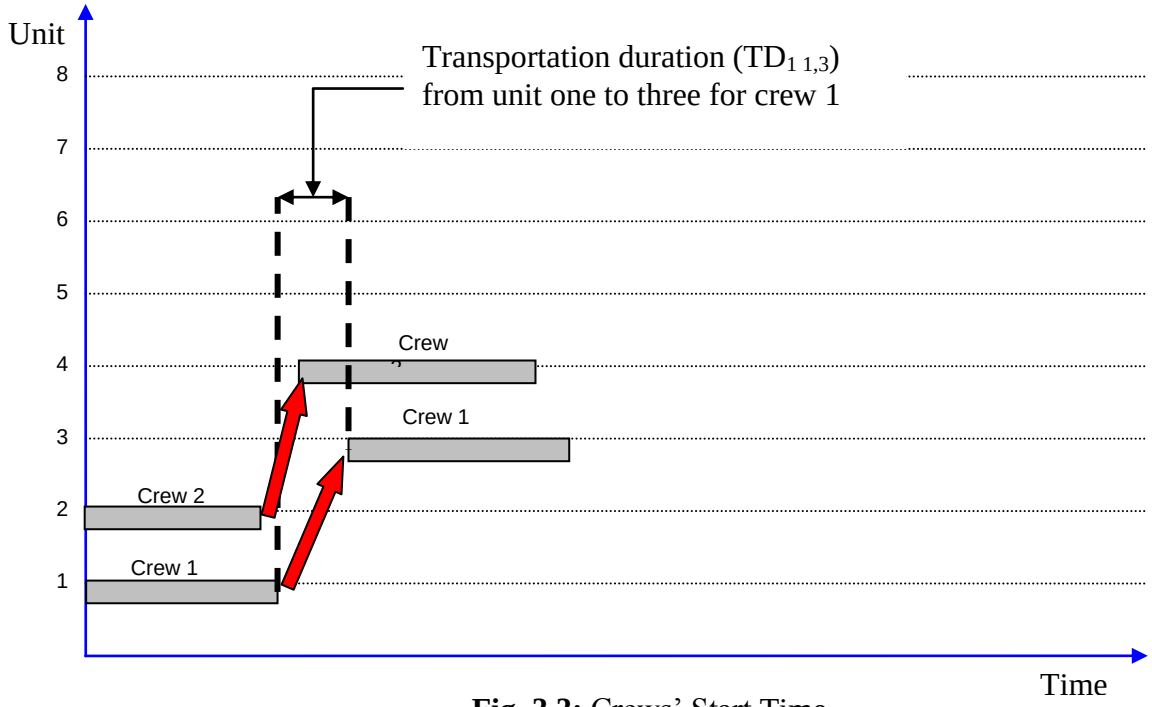


Fig. 3.3: Crews' Start Time

Where, m is the construction method's crew index, j is the current unit, (*PU*) is the previous unit, (*DS*) is the distance to be travelled by the crew from one unit to the other and *(SP*) is the average speed of the construction method's crew (*m*).

The transportation cost ($TC_{m\ PU,j}$) of any crew (*m*) from one unit (*PU*) to the other (*j*) is obtained through multiplying the transportation distance (*TS*) between the two units by the cost of transportation per unit distance as shown in Eq. 3.7.

$$TC_{m\ PU,j} = TS_{PU,j} * CT_m \quad (3.7)$$

Where, (*m)* is the construction method's crew index, (*j*) is the current unit, (*PU*) is the previous unit and (*CT*) is the cost of transportation of the construction method's crew (*m*) per unit distance.

4) Adjusting the crew's available start dates in the upcoming units

After scheduling an activity (*i*) in unit (*j*) using the assigned construction method's crew (*m*), the earliest possible start dates for the following units of the same activity using the same construction method should be adjusted.

The adjustment is obtained through checking the upcoming units of the same activity type and changing their earliest possible start date ($SC_{i,j}$) for the same construction method's crew (*m*) to the finish date of the current activity ($F_{i,j}$) as shown in Fig. 3.4.

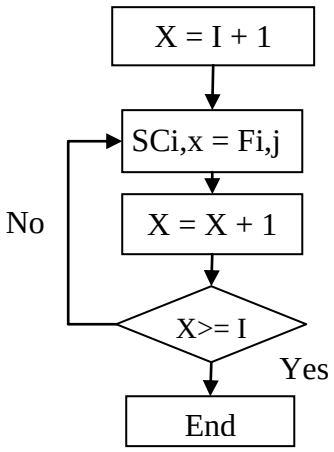


Fig. 3.4: Adjusting The Crew's Earliest Possible Start Time for the Next Units

At this point, the scheduling module have created a practical plan that takes into consideration the production rate of the assigned construction method's crew, its transportation duration and cost, detection of the previous unit that crew have been undertaking and changing the crew's earliest possible dates for the same activity in the next units.

The scheduling module also calculates:

- Total project duration
- Total project cost
- Total project crews' interruptions
- Total units' delivery delays

1) Total project duration

The total project duration equals to the maximum finish date of the last activity in each unit. This can be represented as in Eq. 3.8:

$$TPD = Max\ [F_{i,J}\ ] \tag{3.8}$$

Where, TPD is the total project duration.

2) Total project cost

The total project cost consists of three parts: direct cost, indirect cost and transportation cost. Both the direct and transportation cost are construction method's dependent, while the indirect cost is a duration dependent cost as shown in Eq. 3.9.

$$TPC = \sum[\ CC_{m,i,j}\ + TCm_{PU,j}\ ] + TPD * IC \tag{3.9}$$

Where, (*TPC*) is the total project cost, (*CC*) is the construction methods (*m*) cost assigned to activity (*i*) in unit (*j*), (*IC*) is the indirect cost index per day.

3) Total project interruptions

Interruptions may occur due to the difference between the logical start time ($SL_{i,j}$) and the crew's earliest possible start date ($SC_{i,j}$). It needs to be calculated and minimized (through the optimization module) to increase the utilization of resources.

Interruptions are only found when the resources are idle, and not being used or undertaking activities. This may happen as the logical start date is greater than the crew's earliest possible start date. The module first check if there would be interruptions in the first place, and then calculates the interruptions as shown in Eq. 3.10.

$$If\ SL_{i,j} > SC_{ij}\ then\ Inter_{i,j}\ = [SL_{i,j} - SC_{ij}\ ] \tag{3.10}$$

Where $Inter_{i,j}$ is the interruption in activity (*i*) at unit (*j*).

The total project interruption is calculated using Eq. 3.11.

$$TPI = \sum Inter_{i,j} \tag{3.11}$$

Where, *TPI* is the total project interruptions.

4) Total project units delivery delays

Meeting Delivery dates of project's units is an important key in a successful repetitive construction project. Thus, it is important to calculate any units' delivery delays.

First, the module check for each activity if there's a required finish date. If the check is true, then it checks if there's any delivery delay and calculates it as shown in Fig. 3.5.

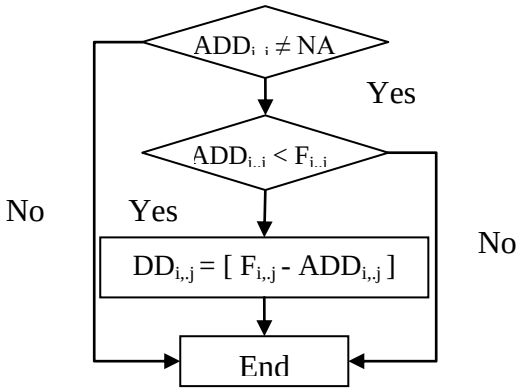


Fig. 3.5: Calculating Unit's Delivery Dates Delays

Where, $ADD_{i,.j}$ is the activity (i) at unit (j) required delivery date, $DD_{i,.j}$ is the delivery delay for activity (i) in unit (j).

The total project delivery delay is then calculated through Eq. 3.12.

$$TPDD = \sum [\, DD_{i,.j} \,] \qquad (3.12)$$

A flow chart showing an over view on the scheduling module is presented Fig. 3.6.

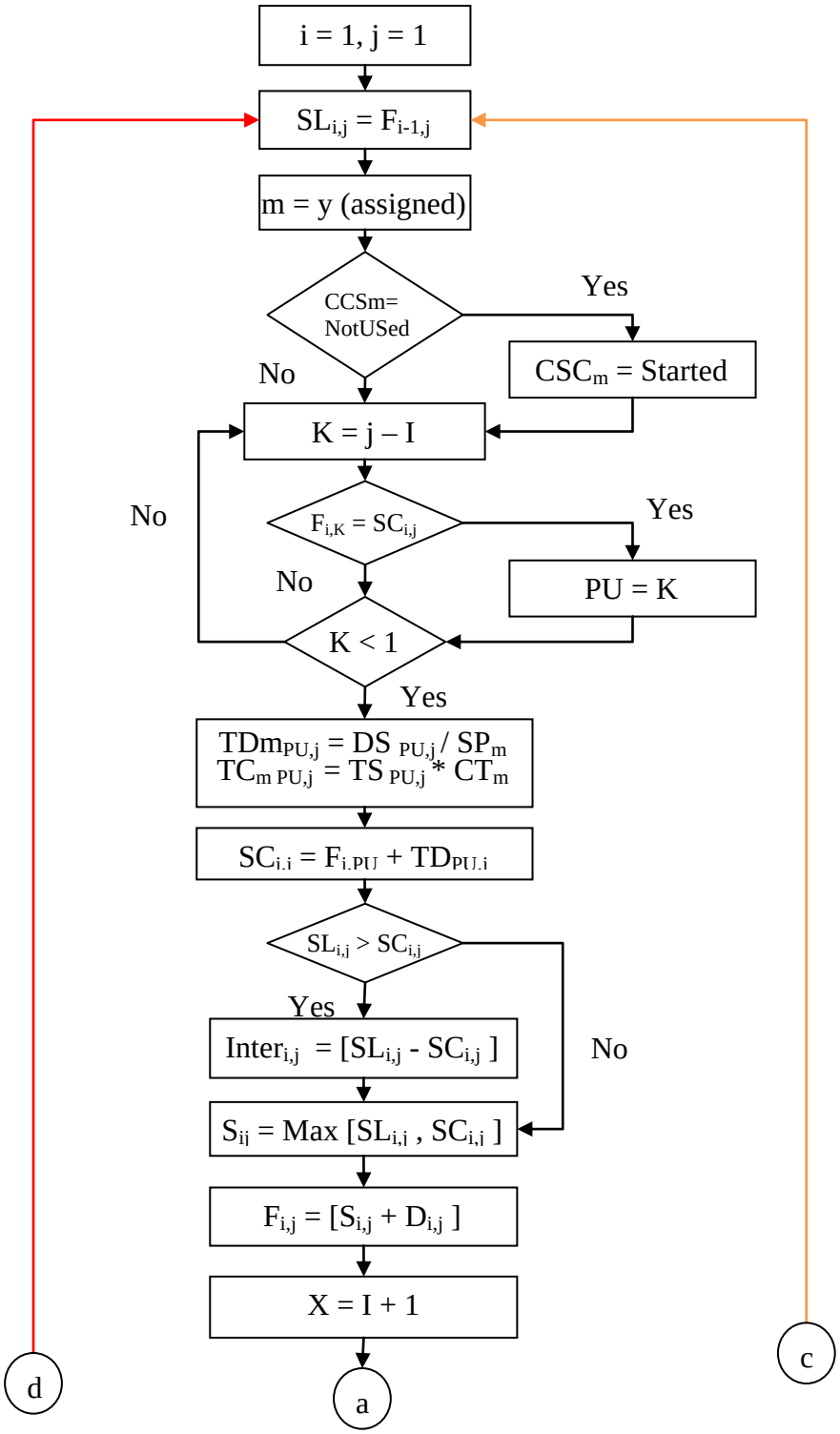


Fig. 3.6: Scheduling Module Overview

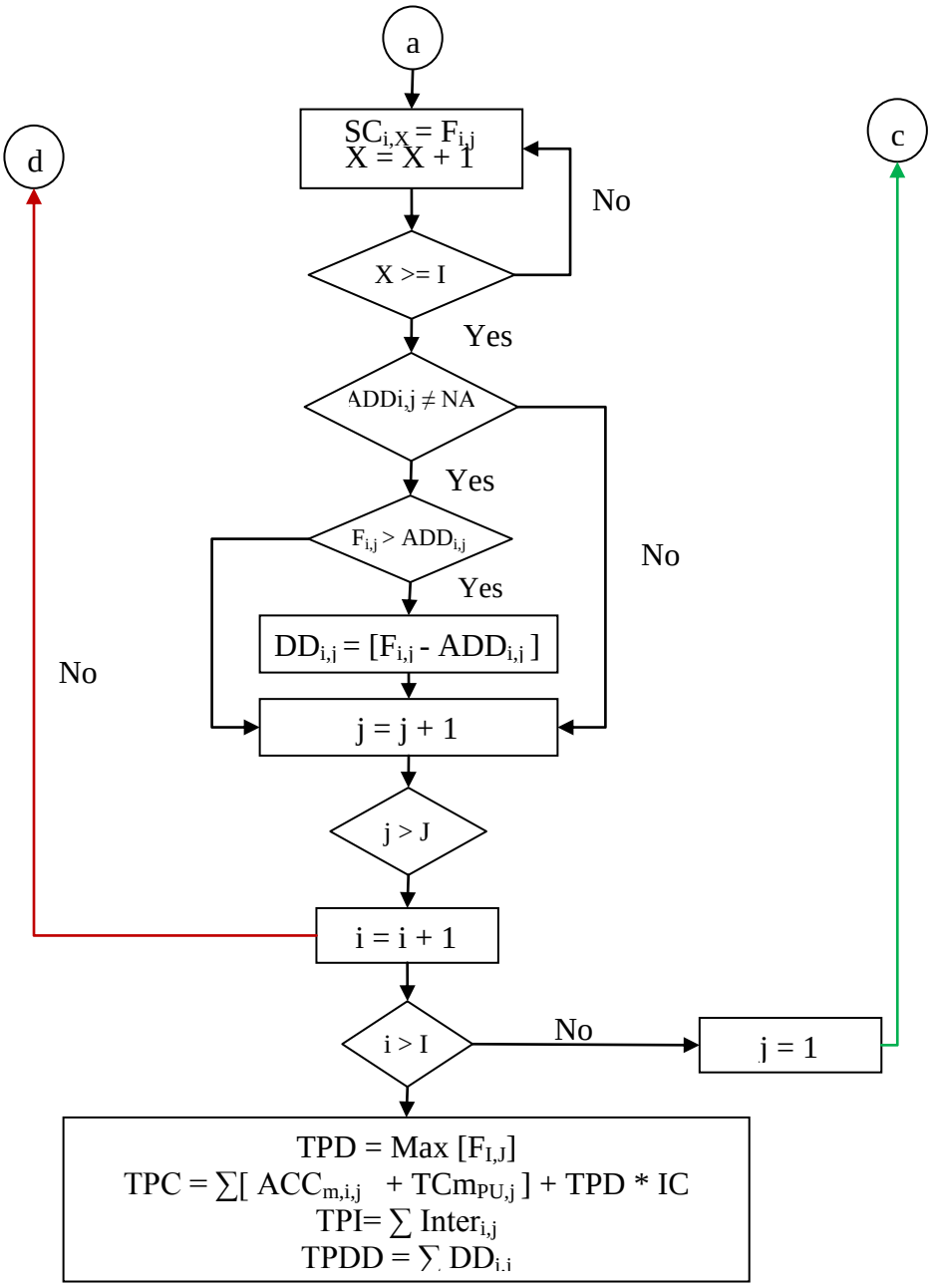


Fig. 3.6: Scheduling Module Overview (Continued)

3.3.2 Multi-Objective Optimization Module

The proposed scheduling module formulation is suitable for generating repetitive activities projects' schedules depending on the indices of the construction methods assigned to the activities in the different units. However, due to the great number of feasible solutions (schedules) depending on the combinations of construction methods assigned to different activities each has its own outputs (duration, cost, interruptions and units delivery delays), An optimization module is necessary to select the optimum set of schedules for the project manager to choose from. The optimization module as such requires identifying the objective function, the optimization variables and the optimization constraints.

3.3.2.1 Objective functions

The optimization of the proposed model is carried out through a four objective functions; (1) minimize total project duration, (2) minimize total project cost, (3) minimize total project interruptions, and (4) minimize total project units' delivery delays. These objective functions are all calculated from the scheduling module for each schedule as mentioned in the previous section. The solutions (schedules) are evaluated through multiple dimensions, multi-objective optimization functions as given in Eq. 3.13.

$$\left.\begin{array}{c} \underline{Min.}\ \text{Total Project Duration: } F_{I,J} \\ \underline{Min}.\ \text{Total Project Cost: } \sum[\ ACC_{m,i,j}\ + TCm_{PU,j}\ ] + TPD * IC \\ \underline{Min}.\ \text{Total Project Interruptions: } \sum Inter_{i,j} \\ \underline{Min}.\ \text{Total Project's Units' Delivery Dates Delays: } \sum DD_{i,j} \end{array}\right\} \quad (3.13)$$

In order to evaluate the solutions (schedules) based on the four objectives, the Pareto-Front sorting concept (Fig. 3.7) is used. The Pareto-Front sorting process starts by identifying a set of non-dominated solutions, which will be ranked as the first Pareto Front. Then the process continues to rank the other schedules to the second Pareto Front and so on till all the solutions are ranked to their fronts. Consequently, the fitness of any solution equals the inverse of its rank (Pareto Front index) (Beradi et al. 2009, Li and Zhang 2009, Elbeltagi et al. 2010).

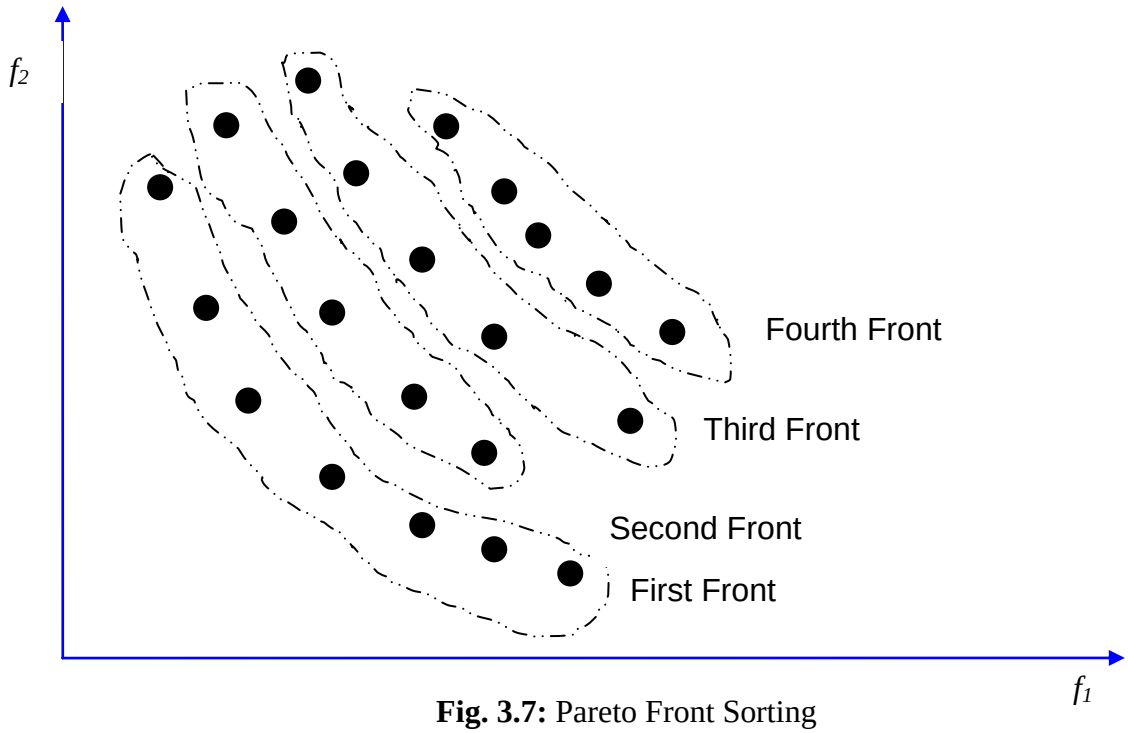


Fig. 3.7: Pareto Front Sorting

A solution (schedule) with a lower-numbered rank is assigned a higher fitness than that for a solution with a higher-numbered rank. Accordingly, for a minimization problem, the fitness of each solution i is calculated by Eq. 3.14 (Elbeltagi et. al. 2010).

$$\text{Fitness}_i = 1 / \text{rank}_i \tag{3.14}$$

Where $fitness_i$ and $rank_i$ are the fitness value and rank number for the solution i.

The main goal of the multi-objective optimization process is to achieve a continuous improvement of the solutions quality within successive iterations.

3.3.2.2 Optimization variables

As earlier mentioned, the independent variables in the proposed model are the construction methods assigned to each activity. Each construction method's crew has its own production rate, direct cost, and available start date. The number of variables for each solution (schedule) is I * J, where I is the number of activities types and J is the number of units. For example, a project consisting of five activities repeated in four units will produce twenty variables. Assuming a four construction methods available for each activity type will result to a solution space of 4^{20} possible schedules (four construction methods assigned to the twenty activities). Therefore, a manual optimization attempt for a small repetitive activity construction project will result to an enormous number of solutions and very tedious work to determine the optimum solution due to the highly dynamic nature of the model. Optimization variables' values are represented in genetic algorithm by the genes' values, and the space where the gene is found is the activity, and the chromosome length is the total number of genes (decision variables) as shown in Fig. 3.8.

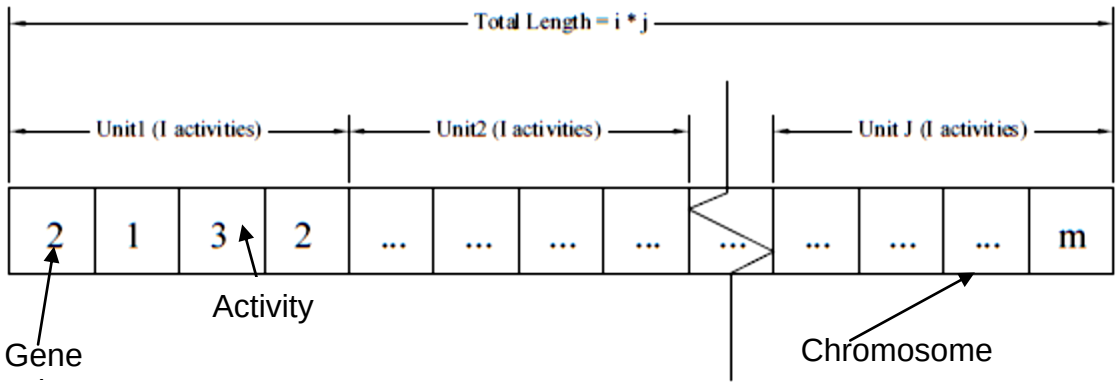


Fig. 3.8: Chromosome Representation Example

3.3.2.3 Optimization constraints

Three constraints have been introduced to the model to keep the solutions (schedules) feasible, these constraints are as follows:

1. The construction methods' indices are limited to the positive integer number of methods available to each activity.
2. The actual number of crews used in each activity is limited to the number of repetitive units, since this model assumes that only one crew works in a single unit.
3. Precedence constraint; the start time of an activity must be greater than or equal to the finish date of the predecessor.

3.3.2.4 Convergence criterion

In order to determine the convergence of the optimization process, the following mechanism is proposed. After sorting all solutions in generation t, the normalized Euclidean norm (NEN) of the non-dominated solutions (First Pareto Front) is calculated to determine the nearest solution to the origin. For each non-dominated solution k in generation t, the NEN value is calculated using Eq. 3.15 and Fig. 3.9 for a two criteria optimization problem (Sanad 2011).

$$NEN_{k}^{t} = \sqrt{\left(\frac{TPC_{k}}{TPC_{\max}}\right)^{2} + \left(\frac{TPD_{k}}{TPD_{\max}}\right)^{2} + \left(\frac{TPI_{k}}{TPI_{\max}}\right)^{2} + \left(\frac{TPDD_{k}}{TPDD_{\max}}\right)^{2}} \quad (3.15)$$

Where: $\underline{TPC_k}$, $\underline{TPD_k}$, $\underline{TPI_k}$, and $\underline{TPDD_k}$ are the cost, duration, interruption, and units delivery dates delays objective function values of solution *k* respectively. TPC_{max}, TPD_{max}, TPI_{max}, and $TPDD_{max}$ are the maximum cost objective, the maximum duration objective, the maximum interruption objective, and the maximum delivery dates delay objective in the non-dominated solutions respectively.

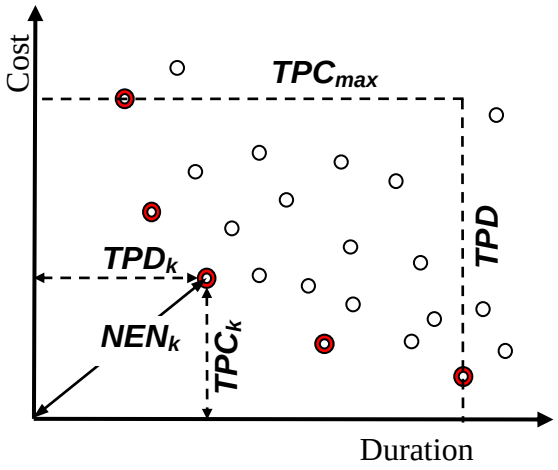


Fig. 3.9: Normalized Euclidean Norm for Non-Dominated Solutions (Sanad 2011)

For each generation, the nearest non-dominated solution to the origin (the non-dominated solution with the minimum NEN) is already identified. Then, the difference between the nearest solution of the current generation and the nearest solution of the previous generation ($Diff_t^{t-1}$) is calculated using Eq. 3.16 (Sanad 2011).

$$Diff_t^{t-1} = \frac{TPC_n^{t-1} - TPC_n^t}{TPC_n^{t-1}} + \frac{TPD_n^{t-1} - TPD_n^t}{TPD_n^{t-1}} + \frac{TPI_n^{t-1} - TPI_n^t}{TPI_n^{t-1}} + \frac{TPDD_n^{t-1} - TPDD_n^t}{TPDD_n^{t-1}} \quad (3.16)$$

where: TPC_n^t, TPD_n^t, TPI_n^t, and $TPDD_n^t$ are the cost, duration, interruption, and delivery dates delays objective function values, respectively, of the nearest solution in generation t. TPC_n^{t-1}, TPD_n^{t-1}, TPI_n^{t-1}, and $TPDD_n^{t-1}$ are the cost, duration, interruption, and delivery dates delays objective function values, respectively, of the nearest solution in the previous generation t-1.

Convergence of the evolutionary process occurs when the $Diff_t^{t-1}$ value has not changed or changes by not more than 1% for a consecutive ten generations.

An overview flow chart of the proposed multi-objective model is presented in Fig. 3.10.

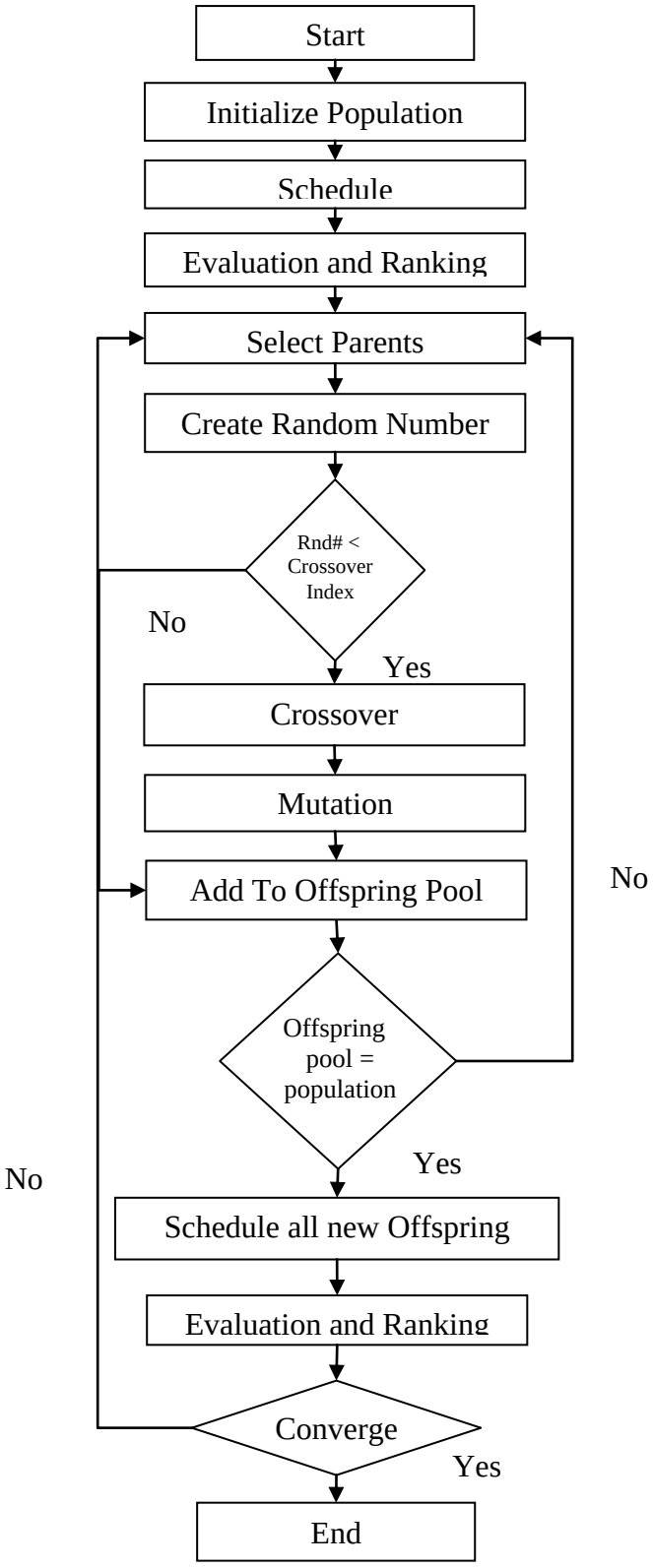


Fig. 3.10: Multi-Objective Optimization Module

3.4 Summary and Conclusions

The development of a multi-objective scheduling and optimization model for a repetitive activity project has been presented in this chapter. The scheduling module takes into consideration the construction methods' crew assigned to each activity and their corresponding production rates, direct costs, transportation durations and costs from one unit to another. The scheduling module handles multi construction methods assignment strategy. The module also calculates the project's total project duration, the project's total cost including the indirect cost, the project's total crews' interruptions and the project's total units' delivery dates delays.

The multi-objective optimization module selects combinations of the appropriate construction methods assigned to the different activities to determine the optimum set of schedules through a four multi-dimensional objective functions.

CHAPTER 4

MODEL IMPLEMENTATION

4.1 Introduction

In this chapter, the implementation of the proposed model is presented along with an example application of repetitive units construction project to validate the model and show its capabilities in scheduling repetitive activities projects. The optimization module using genetic algorithm will also be implemented to determine the optimum sets of plans considering the number of construction methods for each activity type, construction methods' costs, production rate, project's indirect cost, interruptions of working construction methods' crews, and delays in units' delivery dates.

4.2 Implementation Media

The proposed model is implemented on a commercially available and widely spread scheduling software (MS Project 2007) for its ease of use and simple interface. Also, for its wide use by construction practitioners. The software provides the planner with simple data entry of the activities; dependencies, relationships, duration…etc. The software performs CPM calculations on the project as well as representing the project schedule in bar chart and network diagrams. MS Project, also, allows modeling more complex algorithms by implementing the model through Visual Basic Application macro tool (VBA macro). VBA allows dynamically altering the schedule depending on the inputs given to the model. The implementation for scheduling and optimizing repetitive activities projects will follow the same steps as the formulation explained in Chapter three.

4.3 Implementation Details

The implementation of the model on MS Project (2007) starts by creating a regular construction plan through data entry of the activities – repeated according to the number of repeated units – and their relationships through the same unit along with their corresponding available construction methods' crews' duration, direct costs as well as the required delivery date of the unit if needed. An example of five activities repeated through five different units is presented in this and the following sections to illustrate the implementation of the model.

As shown in Figs. 4.1 and 4.2, a data entry of the construction repetitive project is completed through the four units with their dependencies and relationships. Each of the five repetitive activities can have up to four different construction methods. Each construction method has its own duration to undertake the activity as well as its direct cost and the available start time to be assigned to this activity. The required units' delivery dates are also entered in their corresponding column. For the VBA implementation purpose, any unavailable data will be entered as 0 days, NA, and $0 for no construction method's duration, start time, and direct cost, respectively. For example, the first activity - excavation1 - has only one available construction method, thus the duration for the available three other methods are 999days, their corresponding start time are NA, and their costs are $0. As well as the delivery date of this activity is NA as there is no required delivery date.

The data entered are then used by the MS Project regular CPM calculation to calculate the project duration. On starting the proposed model implemented on VBA macros, the user will be asked to enter six inputs that start up both the scheduling and optimization modules as shown in Fig. 4.3:

- Number of activities
- Number of Units
- Project indirect cost per day
- Initial population size

- Crossover probability rate
- Mutation probability rate

These data entered aids the VBA macros to create a data base for the proposed repetitive activity project in addition to guidance for the GA's optimization process. The VBA macros then starts to create an initial random solutions (chromosomes) depending to the population size desired, schedule these solutions, evaluate them and starts optimizing them through evolution process described in chapter three. VBA code of MS Project is provided in Appendix I.

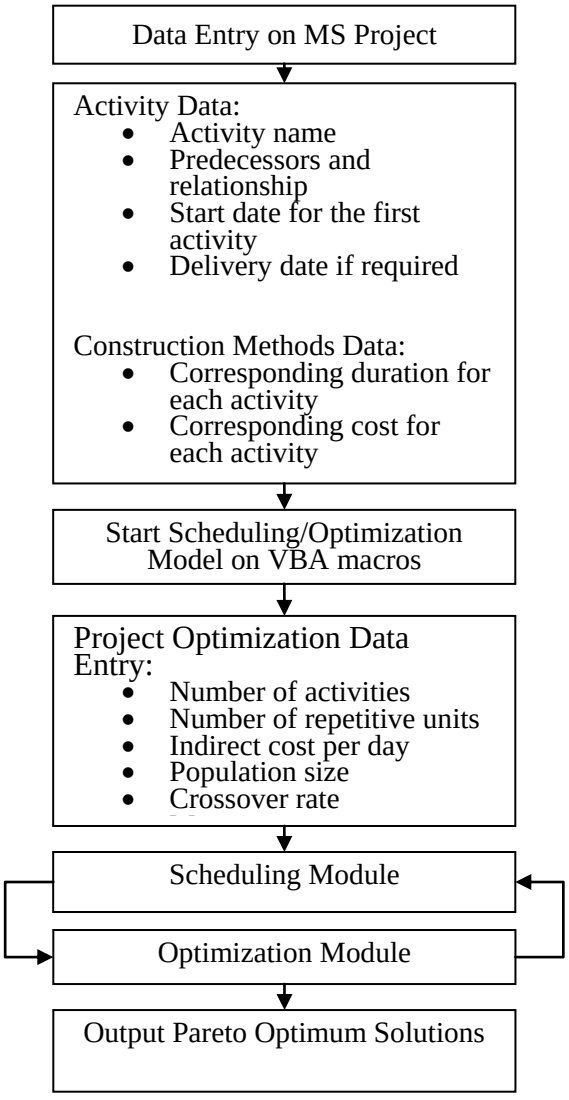


Fig. 4.1: Model Implementation Process

ID	Task Name	Duration (days)	Const. Method 1 (days)	Const. Method 2 (days)	Const. Method 3 (days)	Const. Method 4 (days)	Start	Start1	Start2	Start3	Start4	Finish	Pred.	Delivery Date	Const. Method1 Cost	Const. Method2 Cost	Const. Method3 Cost	Const. Method4 Cost
1	Excavation1	12.5	12.5	999	999	999	1/3/2011	1/3/2011	NA	NA	NA	1/19/2011		NA	$1,000.00	$0.00	$0.00	$0.00
2	Foundation1	11.5	11.5	14.37	19.16	999	1/4/2011	1/3/2011	1/3/2011	1/3/2011	NA	2/3/2011	1	NA	$3,200.00	$3,000.00	$2,500.00	$0.00
3	Columns1	18.15	18.15	15.12	12.95	999	1/5/2011	1/3/2011	1/3/2011	1/3/2011	NA	3/2/2011	2	NA	$3,700.00	$3,900.00	$4,000.00	$0.00
4	Beams1	8.59	8.59	10.01	12.02	15.02	1/6/2011	1/3/2011	1/3/2011	1/3/2011	1/3/2011	3/14/2011	3	NA	$4,500.00	$4,000.00	$3,900.00	$3,700.00
5	Slabs1	0	0	0	999	999	1/7/2011	1/3/2011	1/3/2011	NA	NA	3/14/2011	4	3/15/2011	$0.00	$0.00	$0.00	$0.00
6	Excavation2	15.63	15.63	999	999	999	1/8/2011	1/3/2011	NA	NA	NA	1/24/2011		NA	$1,200.00	$0.00	$0.00	$0.00
7	Foundation2	12	12	15	20	999	1/9/2011	1/3/2011	1/3/2011	1/3/2011	NA	2/9/2011	6	NA	$3,400.00	$3,100.00	$3,000.00	$0.00
8	Columns2	10.71	10.71	12.5	10.71	999	1/10/2011	1/3/2011	1/3/2011	1/3/2011	NA	2/24/2011	7	NA	$3,500.00	$3,750.00	$3,900.00	$0.00
9	Beams2	13.01	13.01	10.84	13.01	16.25	1/11/2011	1/3/2011	1/3/2011	1/3/2011	1/3/2011	3/15/2011	8	NA	$4,300.00	$3,800.00	$3,600.00	$3,500.00
10	Slabs2	4.8	4.8	17.78	999	999	1/12/2011	1/3/2011	1/3/2011	NA	NA	3/22/2011	9	4/22/2011	$8,000.00	$6,000.00	$0.00	$0.00

(a) **(b)** **(c)**

Fig. 4.2: Project Data Entry

a) Activities' and available construction methods' durations

b) Construction methods' start dates

c) Construction methods' costs and units delivery dates

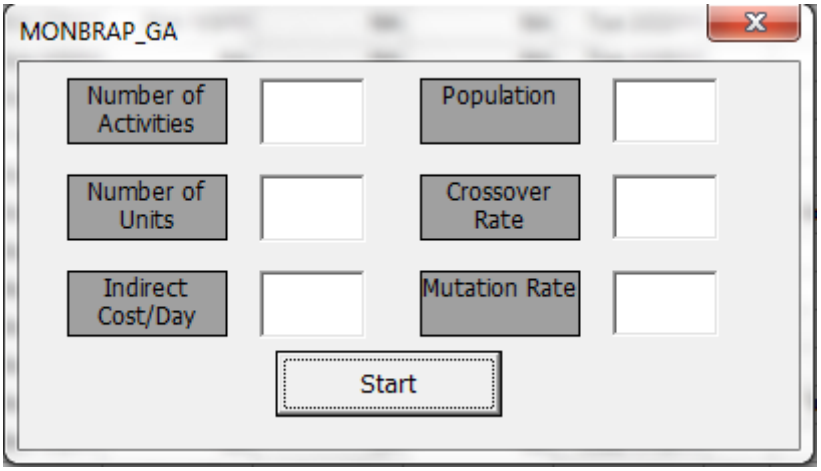


Fig. 4.3: Start up of Scheduling and Optimization modules

4.4 Example Application and Validation

In order to validate the proposed model and demonstrate its capabilities in scheduling and optimizing multi-objective repetitive activities projects, an example of a three span concrete bridge drawn from literature (Hyari and El-Rayes 2006) is analyzed. The project consists of five activities; excavation, foundations, columns, beams and slabs that are repeated in four sections of the project which would generate a solution alternatives of 4^{20} possible schedule. The precedence relationships among these five successive activities are finish to start with no lag time as shown in Table 4.1.

Table 4.1: The Example Application Activities and Predecessors.

ID	Activity Description	Predecessor
1	Excavation	-
2	Foundation	1
3	Column	2
4	Beam	3
5	Slab	4

The activities' quantity of the example project in each unit is presented in Table 4.2 below. For example, the quantities of foundation for units 1, 2, 3 and 4 are 1032, 1077, 943 and 898 m^3 respectively.

Table 4.2: Quantities of Activity in Each Unit

Repetitive Activities					
Repetitive units (j)	**Excavations Quantities (Qij) m^3**	**Foundations Quantities (Qij) m^3**	**Columns Quantities (Qij) m^3**	**Beams Quantities (Qij) m^3**	**Slabs Quantities (Qij) m^3**
1	1147	1032	104	85	0
2	1434	1077	86	92	138
3	994	943	129	101	114
4	1529	898	100	80	145

The module takes into account up to four construction methods per activity type, and as provided from the example. Table 4.3 shows the given production rate for each construction method while Table 4.4 shows the duration needed for each construction method to undertake each activity in the four units depending on their quantities.

Table 4.3: Construction Methods Production Rates

Activity Description	**Construction Method 1 (m^3/day)**	**Construction Method 2 (m^3/day)**	**Construction Method 3 (m^3/day)**	**Construction Method 4 (m^3/day)**
Excavation	91.75	NA	NA	NA
Foundation	89.77	71.81	53.86	NA
Column	5.73	6.88	8.03	NA
Beam	9.9	8.49	7.07	5.66
Slab	28.73	7.76	NA	NA

Table 4.4: Construction Methods' Duration in Each Unit

Activity Description / Construction Method	Unit	Excavation	Foundation	Beam	Column	Slab
Construction Method 1 Duration (days)	**1**	12.5	11.5	8.59	18.15	0
	2	15.63	12	9.29	15.01	4.8
	3	10.83	10.5	10.02	22.51	3.97
	4	16.66	10	8.08	17.45	5.05
Construction Method 2 Duration (days)	**1**	NA	14.37	10.01	15.12	0
	2	NA	15	10.84	12.5	17.78
	3	NA	13.13	11.9	18.75	14.69
	4	NA	12.51	9.42	14.53	18.69
Construction Method 3 Duration (days)	**1**	NA	19.16	12.02	12.95	NA
	2	NA	20	13.01	10.71	NA
	3	NA	17.51	14.29	16.06	NA
	4	NA	16.67	11.32	NA	NA
Construction Method 4 Duration (days)	**1**	NA	NA	15.02	NA	NA
	2	NA	NA	16.25	NA	NA
	3	NA	NA	17.84	NA	NA
	4	NA	NA	14.13	NA	NA

All the data of this example are entered to MS Project plan as earlier stated in section 4.3. The example drawn from the literature doesn't take into consideration the transportation duration of construction methods' crews, also it has only two objectives; (1) minimize project duration and (2) minimize crews' interruptions. For this reason, through the first trial of validation, construction methods' crews' transportation durations and costs as well as two of the proposed objectives; (1) minimize total project cost and (2) minimize the units' delivery delays, are neglected and will be studied in the later sections to compare the results obtained with the results from the literature.

4.4.1 Scheduling module

By pressing the start button, the developed system reads the data entered and then determines the maximum number of construction methods available for each activity. Then, it reads the duration of each activity in respect with the available construction methods and their corresponding costs, the delivery dates required for each activity (if any).

The model, then randomly creates the initial set of chromosomes depending on the population size as predefined by the user. The chromosome length equals the multiplication of number of activities by the number of units. In the current example, the chromosomes length equals to twenty genes.

Each gene value (variable) is the index of the construction method assigned to an activity. These variables are limited to the maximum number of construction methods available for this activity type. For example, the gene number 2 – foundation in the first unit – can have a value from one to three, as there are up to three different construction methods available for this activity type.

Upon creating a given chromosome the scheduling module calculates the total project duration proposed by this chromosome, depending on the genes values (construction method assigned to the activities), as well as the total project interruption.

The scheduling module - as presented in Appendix I - starts by determining the construction method index (gene value), then determines if the assigned construction method's crew has been assigned to any other units before the current one so as to determine the previous unit and the corresponding transportation duration and cost. The module then calculates the assigned crew available start date, and compares it with the logical start date to determine the activity's start date and crew's interruption, if any.

After scheduling an activity, the module compares the finish date with the required delivery date of the activity, and calculate the delivery date delay if any. Then the module adjusts the assigned construction method's crew available start date for the next units to the finish date of the current activity. The scheduling module finishes its role after calculating the chromosome's total project duration and total project interruption as well as total project delivery delays and total project cost. However the last two criteria are neglected for comparison purposes with the example drawn from the literature.

4.4.2 Optimization module

The optimization module starts by evaluating the chromosomes depending on their results achieved from the scheduling module through sorting these chromosomes into Pareto Fronts from 1 to *n*, where *n* is the number of fronts created and 1 is the highest rank. Each chromosome is then given its fitness by calculating the inverse of the rank as mentioned in Chapter three.

To calculate the selection probability of each chromosome, the optimization module first calculates the relative fitness of each chromosome using Eq. 4.1.

$$\text{Rel. Fitness}(i) = \text{Fitness}(i) / \sum_{i=1}^{I} \text{Fitness}(i) \quad (4.1)$$

Where (i) is the number of chromosome.

The relative fitness of each chromosome is then cumulatively added through the whole population to obtain the selection probability for each chromosome as illustrated in Fig. 4.4, creating higher probability for the chromosomes with the

higher fitness. The selection of the parents chromosomes are carried out using a "Roulette Wheel" selection criteria, where a series of randomly created number are compared with the parent chromosomes' selection probabilities to introduce them to GAs operations or else will be inserted to upcoming generation as they are.

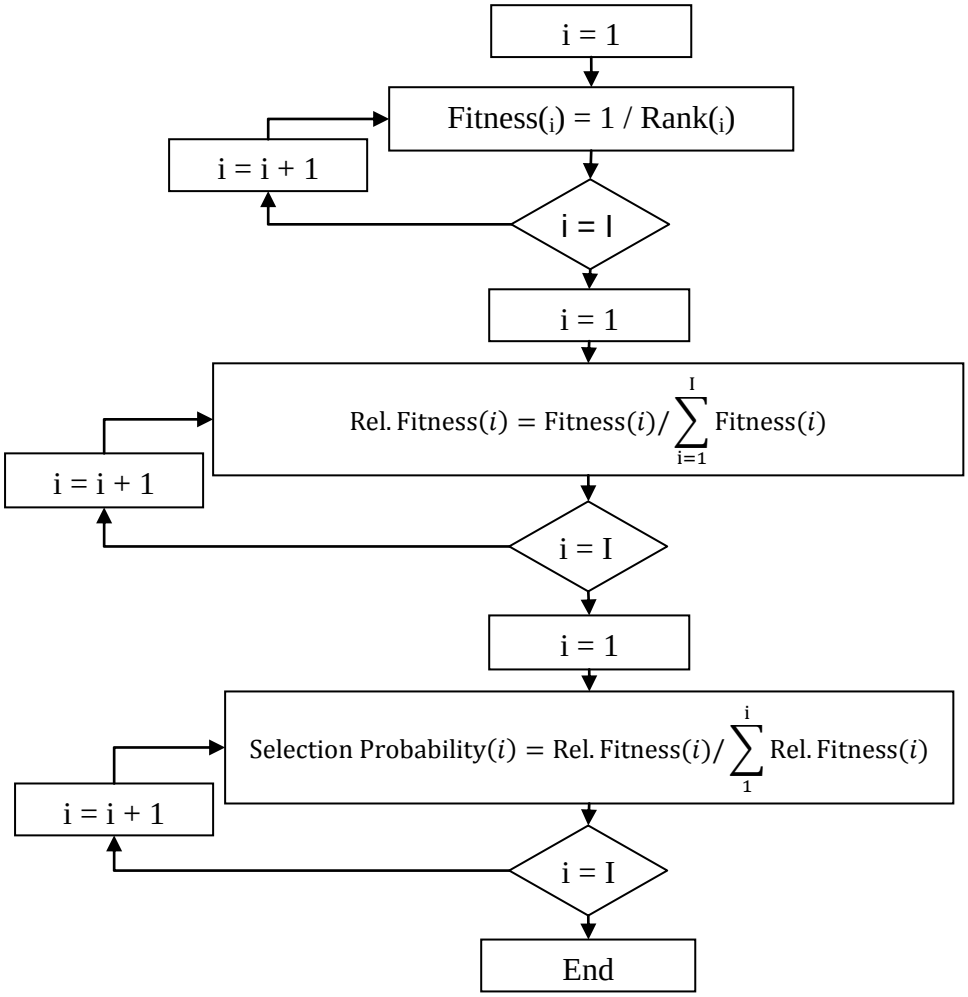


Fig. 4.4: Selection Probability Calculation Process

4.4.3 Analysis of results

Sample solutions of the proposed model results are analyzed and compared to the solutions drawn from the literature. The population proposed is 200 chromosomes and with a crossover and mutation probability indices of 0.85 and 0.2 respectively. A sample of the Pareto Front solutions (schedules) are shown in Table 4.5.

Table 4.5: Assigned Construction Methods' Indices for Each Activity

	Schedule1	Schedule2	Schedule3	Schedule4	Schedule5	Schedule6
Excavation1	1	1	1	1	1	1
Foundation1	3	3	3	3	3	3
Columns1	1	2	1	2	1	1
Beams1	3	4	4	4	4	2
Slabs1	2	2	2	2	2	2
Exavation2	1	1	1	1	1	1
Foundation2	2	1	1	3	1	1
Columns2	1	2	3	1	2	1
Beams2	2	4	2	2	2	4
Slabs2	1	1	2	1	2	1
Excavation3	1	1	1	1	1	1
Foundation3	1	1	1	2	1	1
Columns3	3	3	3	3	2	2
Beams3	3	2	3	1	3	3
Slabs3	1	1	2	1	2	1
Excavation4	1	1	1	1	1	1
Foundation4	1	1	1	1	2	2
Columns4	3	3	3	3	3	3
Beams4	1	1	1	1	1	1
Slabs4	1	1	1	1	1	1
Project Duration	91	92	92.5	93	93.5	94
Project Interruptions	12	6	5	4	3	0

Table 4.6: Results Comparison

Comparison element	Proposed model Sample solutions						Hyari and El-Rayes (2006) Sample solutions					
Project Duration	91	92	92.5	93	93.5	94	106.8	107	108.5	110.9	114.3	117.9
Project Interruption	12	6	5	4	3	0	15	14	11	8	4	0

The gap between the proposed model results and the example drawn from literature results (as shown in Table 4.6) are due to the ability to use more than one construction method for the same activity type creating flexibility when dealing with large number of repetitive units. Through analyzing the results obtained from the proposed model and results by Hyari and El-Rayes (2006), it is observed that the current model's project duration varied between a maximum value of 94 days and minimum value of 91 days, while Hyari and El-Rayes (2006) minimum project duration is 106.8 days and expanded to 117.9 days. However, both models reached a zero project interruption, yet the current model's maximum project interruption is 12 days while Hyari and El-Rayes

(2006) project interruption reached 15 days. Fig. 4.5 shows a comparison between the proposed model's results and the results drawn from the literature.

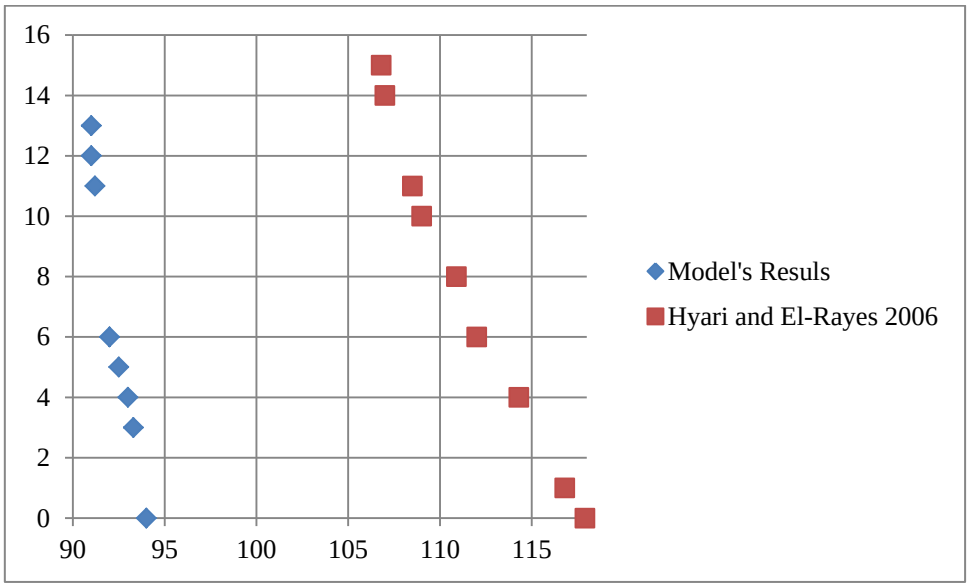


Fig. 4.5: Results Comparison: Proposed Model Vs. Hyari and EL-Rayes (2006)

4.5 Further Experimentations

This section presents the proposed model full ability to optimize the four objectives simultaneously; (1) minimization of project duration, (2) minimization of project cost, (3) minimization of project total interruptions and (4) minimization of units' delivery dates delays.

The example drawn from literature didn't take into account the project cost; direct cost, indirect cost and transportation cost. The example, also, didn't consider the units' delivery dates required. For this purpose, some assumptions have been taken into account to experiment the model by adding costs for the construction methods undertaking each activity, as well as their transportation costs and delivery dates of each activity if required as shown earlier in Fig. 4.1.

Table 4.7 shows the direct cost of each activity with respect to the different construction methods that can be assigned to them. The indirect cost for this experimentation is assumed to be LE200 per day with a population size of 200 and crossover and mutation probability of 0.85 and 0.15 respectively.

Table 4.7: Activities Direct Costs

Activity	Repetitive Unit	Construction Method1 ($)	Construction Method2 ($)	Construction Method3 ($)	Construction Method4 ($)
Excavation	1	1000	0	0	0
	2	1200	0	0	0
	3	4700	0	0	0
	4	4700	0	0	0
Foundation	1	3200	3000	2500	0
	2	3400	3100	3000	0
	3	3500	3150	3000	0
	4	3500	3200	3000	0
Columns	1	3700	3900	4000	0
	2	3500	3750	3900	0
	3	4500	3300	3200	0
	4	5000	5200	5600	0
Beams	1	4500	4000	3900	3700
	2	4300	3800	3600	3500
	3	4500	4300	4000	3800
	4	4700	4500	4100	3900
Slabs	1	0	0	0	0
	2	8000	6000	0	0
	3	8100	6500	0	0
	4	8000	6100	0	0

Table 4.8 illustrates the assumed distances between repetitive units in KM. Table 4.9 shows the assumed speeds for each construction method's crew as well as transportation cost for each crew in Table 4.10 to experiment the impact of crews' speeds and transportation on a repetitive activities project. These data are introduced to the VBA macros using data entry windows as shown in Fig. 4.6 (A, B, and C).

Table 4.8: Distance between Repetitive Units

From \ To	Unit2	Unit3	Unit4
Unit 1	0.5	1	1.25
Unit 2	--	0.5	.75
Unit 3	--	--	0.25
Unit 4	--	--	--

Table 4.9: Construction Methods' Crews' Transportation Speeds

Crews / Activities	Construction Method1(Crew1)	Construction Method2(Crew2)	Construction Method3(Crew3)	Construction Method4(Crew4)
Excavations	1km/day	NA	NA	NA
Foundations	3km/day	5km/day	5km/day	NA
Columns	5km/Day	5km/day	6km/day	NA
Beams	3km/day	5km/day	5km/day	6km/day
Slabs	3km/day	5km/day	NA	NA

Table 4.10: Construction Methods' Crews' Transportation Cost

Crews / Activities	Construction Method1(Crew1)	Construction Method2(Crew2)	Construction Method3(Crew3)	Construction Method4(Crew4)
Excavations	$1000/day	NA	NA	NA
Foundations	$1500/day	$1500/day	$1300/day	NA
Columns	$1400/day	$1300/day	$1300/day	NA
Beams	$1600/day	$1500/day	$1500/day	$1300/day
Slabs	$1300/day	$1300/day	NA	NA

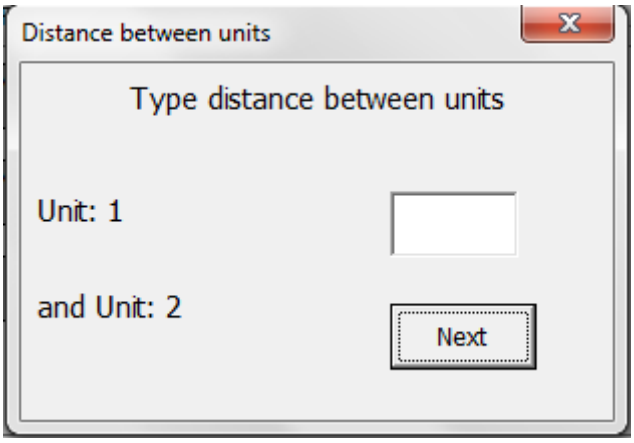


Fig. 4.6-A: Distance between units

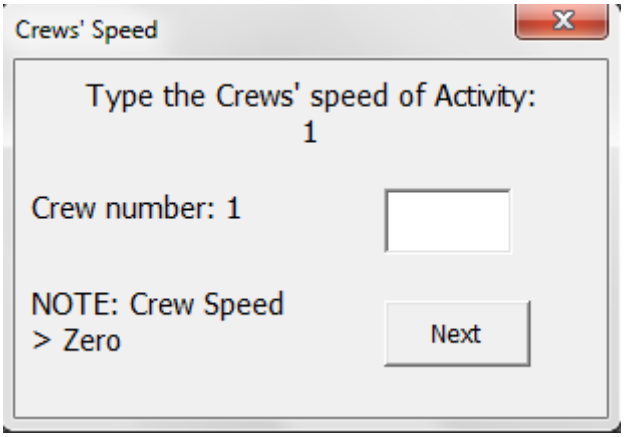


Fig. 4.6-B: Crew's Transportation Speed

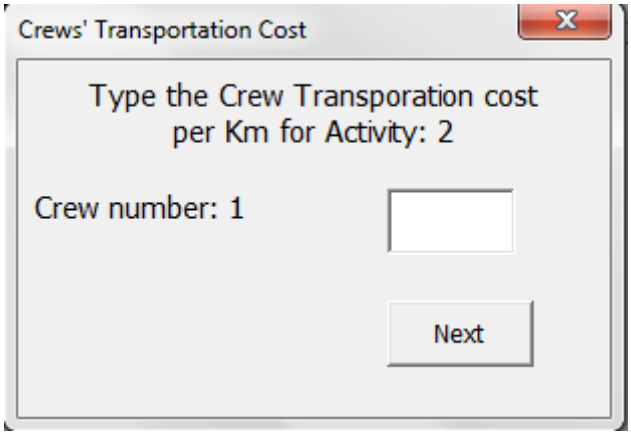


Fig. 4.6-C: Crew's Transportation cost/day

The Pareto optimum solution set expanded to 121 optimum solutions. Selected results from the optimum set obtained from the model are shown in the following Table 4.11.

Table 4.11: Model Results for Full Scale Experimentation

	Sample Schedules Outputs									
	1	2	3	4	5	6	7	8	9	10
Total Cost	96807	97129.8	97109.8	97799	97513	96918	99055	96744	96799	99813
Total Delays	0.93	0	0.93	0	6.3	3.9	4	2.8	3.2	0
Total Duration	94.3	98.1	94	93.24	91	94	93.27	97	94	91
Total Interruption	28	20	13	34	36	10	10	57	35	13

A quick comparison between the optimum 10 solutions is shown in the following figure (Fig 4.7) on a radar chart. A wider look on the First Pareto Front (optimum set) is shown in Fig. 4.8

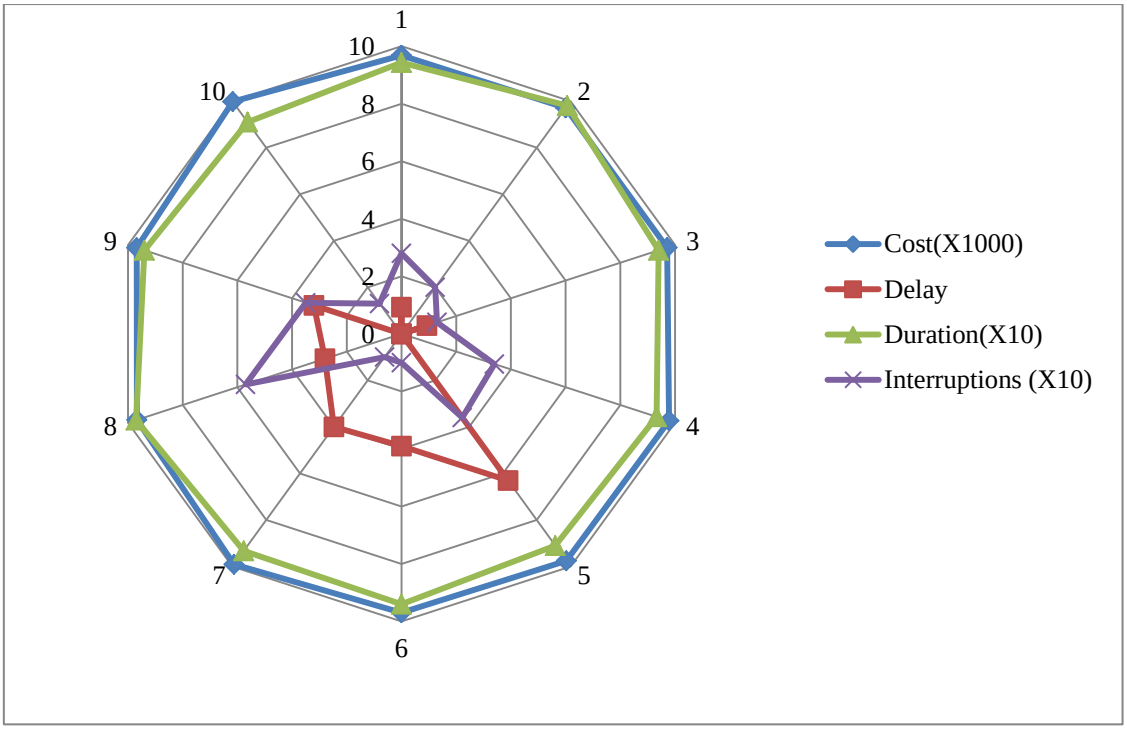


Fig 4.7: Results Comparison

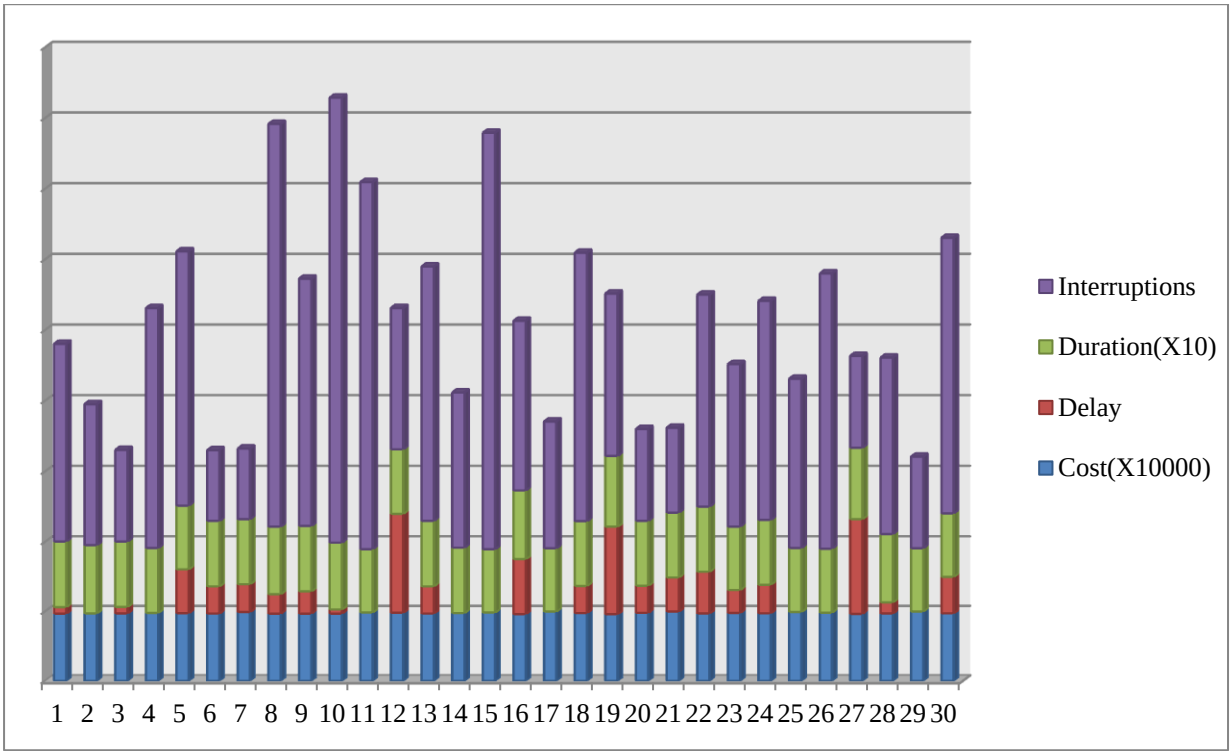


Fig. 4.8: Optimum Set Wider View

4.6 Compromise Solution

On reaching convergence in a GA problem and determining the set of optimized solutions, the decision maker will have to choose the preferable solution for the current project. However, decision makers, in some cases, have no certain criterion facilitates the selection of their preferred solution among the optimized solution set. A "compromise solution" is then preferred for the decision maker to determine the best solution of the optimized set of solutions. In order to determine the compromise solution among the set of Pareto optimal solutions, the procedure introduced by Elbeltagi et al. (2010) is adopted.

Elbeltagi et al. (2010) introduced the following mechanism to determine a single Pareto-compromise solution of the optimization problem with multiple objectives. Concisely presented in the following are the step-by-step mathematical and graphical details of the selection procedure to find the mutually agreeable objective criteria values defining a unique Pareto-compromise solution to a given multi-objective optimization problem with n objectives:

Step 1: Determine the solutions of the Pareto optimization problem defining the Pareto front, and identify extreme objective values $f_i^{\max}$ and $f_i^{\min}$ (i=1, n). For example, suppose n=2 so that the problem is,

$$\text{Minimize } \{f_1(z), f_2(z)\} \tag{4.2}$$

Further suppose f_1 = - (criterion 1) and f_2 = criterion 2, and that the solution to Eq. (4.2) is represented by two criteria vectors f_1^* and f_2^* having m=10 entries and extreme values $f_1^{\max}$, $f_1^{\min}$ and $f_2^{\max}$, $f_2^{\min}$, defining the original Pareto front shown in Fig. 4.9.

Step 2: Normalize the objective criteria vectors f_i^* (i=1, n) defining the Pareto front to find the normalized objective criteria vectors,

$$X_i = \left(f_i^* - f_i^{\min}\right) / \left(f_i^{\max} - f_i^{\min}\right); \quad \text{(i=1, n)} \tag{4.3}$$

having extreme entry values $X_i^{\max} = 1$ and $X_i^{\min} = 0$ (i=1, n). For n=2 and the Pareto front in Fig. 4.10, for example, the corresponding normalized vectors X1 and X2 from Eq. (4.3) define the normalized Pareto front in Fig. 4.10.

Step 3: Reorder the normalized vectors X_i from Eq. (4.3) to form the primary objective criteria vector,

$$x_i = \left[x_i^{\min}, \ldots, x_i^{\max}\right]^T = [0, \ldots, 1]^T; \quad \text{(i=1, n)} \tag{4.4}$$

$$y_i = reordered \left[\left(\sum_{k=1}^{n} x_k - x_i\right) \Big/ (n-1)\right]^T$$

$$= \left[y_i^{\max}, \ldots, y_i^{\min}\right]^T = [1, \ldots, 0]^T; \quad \text{(i=1, n)} \tag{4.5}$$

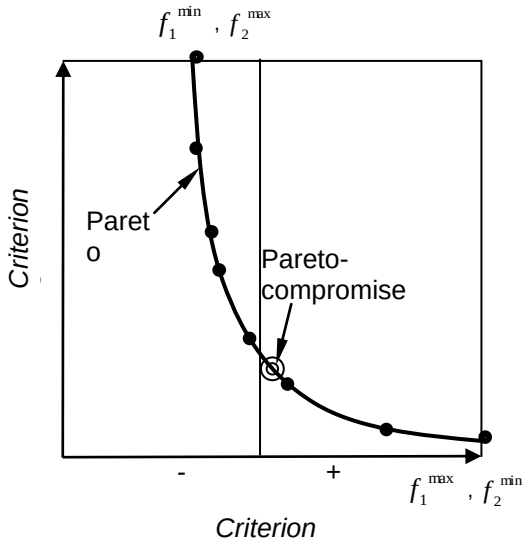


Fig. 4.9: Original Pareto Front (n=2) (Elbeltagi et al. 2010)

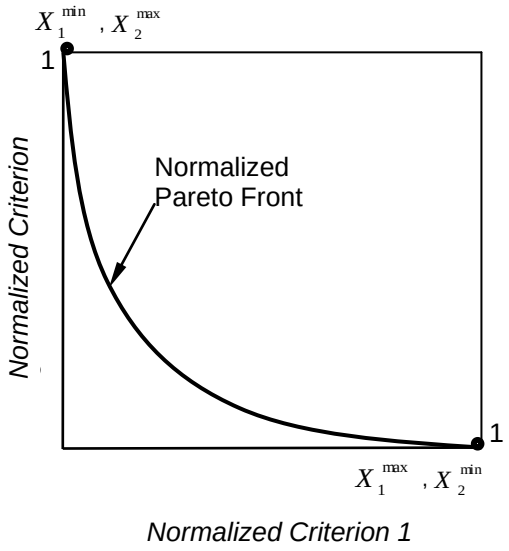


Fig. 4.10: Normalized Pareto Front (n=2) (Elbeltagi et al. 2010)

so that each pair of primary-aggregate criteria vectors (x_i , y_i) defines a 2-dimentional subspace of the n-dimensional objective space, where (x_1 , y_1) = (x_2 , y_2) = (X_1 , X_2) when n=2. From Fig. 4.9, for example, the aggregate vectors are, y_1 = reordered [(x_2 + x_3)/2], y_2= reordered [(x_1 + x_3)/2], and y_3 = reordered [(x_1 + x_2)/2].

Step 4: Uniformly translate and re-normalize each pair of primary-aggregate objective criteria vectors (xi , yi), to create a corresponding translated Pareto front defined by the vectors,

$$x_i^* = (x_i - \delta x)/(1 + \delta x);\ y_i^* = (y_i - \delta y)/(1 + \delta y);\quad (i=1, n) \tag{4.6}$$

where δx and δy are m-dimensional vectors of translation parameters δx=δy=√2-1. For n=2, for example, the normalized Pareto front in Fig. 4.10 is translated to the Pareto front indicated by the dashed curve in Fig. 4.11.

Step 5: For the translated Pareto front defined by each pair of vectors (x_i^*, y_i^*) from Eq. (4.6) determine the 45° radial distance to the centre point Ei(0.5,0.5) of the corresponding 2D-subspace as,

$$\Delta r_i = \sqrt{2}(0.5-(x_j^*+x_j+1^*)(y_j^*+y_j+1^*))/(x_j^*+x_j+1^*+y_j^*+y_j+1^*); \quad (i=1, n) \tag{4.7}$$

where vector index j is such that $x_j^*/y_j^* \leq 1$ whereas $x_j+1^*/y_j+1^* \geq 1$, and $\Delta r1 = \Delta r_2$ ($=\Delta r_0$) when n=2. As shown in Fig. 4.11 for n=2, for example, Eq. (4.7) determines the 45° radial distance from the Pareto front (x^*, y^*) to the centre point $E_i(0.5,0.5)$ to be Δr_0, where the radial-shifted Pareto front (x° , y°) passing through point E_o is circular.

Step 6: For the extreme vector f_i^{max} , f_i^{min} from step 1, and the radial distance Δr_i, from step 5, evaluate the function,

$$f_i^\circ = f_i^{max} - \left(f_i^{max} - f_i^{min}\right)\left(\Delta r_i + \sqrt{2/2}\right); \qquad (i=1, n) \tag{4.8}$$

To find n criteria values f_i° (i=1, n) that collectively define a unique compromise solution that represents a mutually agreeable Pareto-tradeoff between all n criteria.

As shown in Fig. 4.9 for n=2, for example, Eq. (4.8) determines the Pareto-compromise solution to be at point 0 on the original Pareto front.

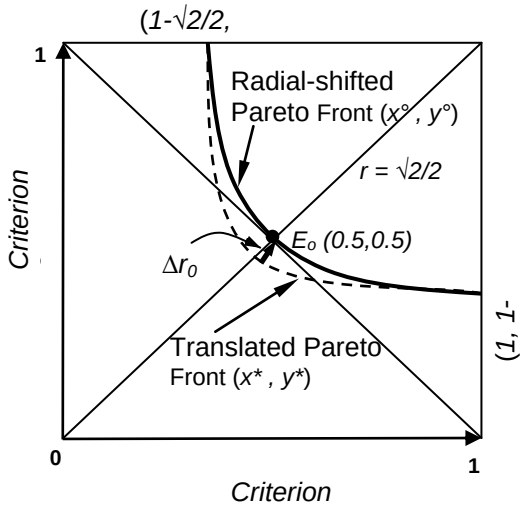


Fig. 4.11: Unique Pareto Trade-off Point Eo (n=2) (Elbeltagi et al. 2010)

4.6.1 Experimental example's compromise solution

The experimental example's Pareto optimum solution set expanded to 121 solutions as previously mentioned in section 4.4. Selecting a single solution from the presented solutions is a difficult task. In order to help the decision maker to select a single solution among the set of Pareto-optimal solutions, the procedure introduced by Elbeltagi et al. (2010) is applied. The resulted Pareto-compromise solution and its theoretical objective values are listed in Table 4.12. The best-alternative solution among the Pareto-optimal solutions is found to be the solution number 18 in Fig. 4.8.

Table 4.12: Pareto-Compromise and Best-Alternative Solutions

Solution	Project Duration (days)	Project Cost ($)	Project Interruptions (days)	Delivery Dates Delays (days)
Pareto-Compromise Solution	91.96	97529.63	24.56	12.96
Best-Alternative Solution	100	95953	23	12.58

4.7 Another Example Application and Validation

In order to increase the proposed model's credibility, an example of highway project drawn from the literature (Moselhi and Hassanain 2003) is analyzed for validation. The project involves the construction of a three-lane-highway of stretch of 15 Km and consists of five activities as follows:

Table 4.13: Second Example's Activities

ID	Activity Description	Predecessor
1	Cut and Chip Trees	-
2	Grub and Remove Stumps	1
3	Earthmoving	2
4	Base	3
5	Paving	4

The project is divided into 15 repetitive units each of length 1 Km and each of the five activities is repeated at each of the 15 segments or units of the project. The precedence relationships among these sequential activities are finish to start with no lag time as given in Table 4.13. The data of quantities for each activity is shown in Table 4.14

Three alternative construction methods were introduced into the model, every construction method may represents different production rates and/or different cost. This approach gives the planner flexibility in creating alternative construction methods by changing one of the three parameters mentioned above and check which schedule is suitable to the project or the planner can use the optimization part of this model to optimize the schedule taking into consideration the relationship between time, cost and interruption. The methods of construction data are shown in Table 4.14.

Table 4.14: Quantities of Activities s in Each Unit

	Cut and Chip Trees	**Grub and Remove Stumps**	**Earthmoving**	**Base**	**Paving**
Repetitive units (Km)	**Quantities m^2**	**Quantities m^2**	**Quantities m^3**	**Quantities m^2**	**Quantities m^2**
1	12000	12000	7000	32000	32000
2	12000	12000	6000	32000	32000
3	18000	18000	6000	32000	32000
4	12000	12000	6000	32000	32000
5	18000	18000	8600	32000	32000
6	30000	30000	7000	32000	32000
7	36000	36000	6500	32000	32000
8	30000	30000	6000	32000	32000
9	24000	24000	6000	32000	32000
10	24000	24000	6000	32000	32000
11	18000	18000	6000	32000	32000
12	12000	12000	6000	32000	32000
13	12000	12000	6000	32000	32000
14	12000	12000	6000	32000	32000
15	12000	12000	6000	32000	32000

.

Table 4.15: Construction Methods Details

ID	Activity Description	Method 1		Method 2		Method 3	
		Rate (unit/day)	**Cost (LE/day)**	**Rate (unit/day)**	**Cost (LE/day)**	**Rate (unit/day)**	**Cost (LE/day)**
1	Cut and Chip trees	3000	2000	2500	1250	3500	1500
2	Grub and remove stumps	4000	2000	3000	1500	3500	1750
3	Earthmoving	1000	1700	1000	2500	900	1600
4	Base	3200	3000	3200	3000	3000	3800
5	Paving	4000	3000	4000	3000	4000	3500

In order to investigate the models validation the indirect cost was estimated for LE3000/day as well as some delivery dates were required for the "Grub and Remove Trees" activity at 9^{th} and 15^{th} unit to finish after 35 and 87 days respectively. The model reached to a Pareto optimum solution set of 16 schedules. Selected schedules from the Pareto optimum solution set is shown in Table 4.15.

Table 4.16: Model Results for Second Validation Example

	Sample Schedules Outputs								
	1	2	3	4	5	6	7	8	9
Total Cost	1351503	1381597	1332308	1383930	1355062	1380989	1348082	1351142	1378152
Total Duration	99	97	89	90	89	95	91	100	88
Total Delays	5	7.9	0	0.9	0	1	8.7	0	8.8

Through the analysis of the model's results in comparison to the proposed model by Moselhi and Hassanein (2003), it was observed that the model minimum duration and maximum total project duration was 88 and 100 days respectively, while the model proposed by Moselhi and Hassanein (2003) minimum and maximum project duration were 87 and 97 days respectively. However the

proposed model gave variant schedules with respect to total project cost and respecting the units' delivery dates required.

Table 4.16 illustrates the Pareto compromise solution and best alternative for the Pareto optimum scheduling set.

Table 4.17: Pareto-Compromise and Best-Alternative Solutions for Second Validation Example

Solution	Project Cost ($)	Project Duration (days)	Delivery Dates Delays (days)
Pareto-Compromise Solution	1355230	88.5	4.4
Best-Alternative Solution	1332308	89	0

4.8 Summary and Conclusions

The implementation of the multi-objective non-unit based repetitive activities project scheduling model have been presented in this chapter. The model is implemented on a MS Project with VBA macros to enable flexible data entry and efficient calculations. The model is capable to generate a number of schedules as predefined by the user and optimize them through the application of genetic algorithm to determine a set of optimum schedule for the user to choose from. Compromise solution has also been presented to an experimental example to aid the decision maker to choose a solution (schedule) from a large number of optimum solutions.

CHAPTER 5

REAL LIFE CASE STUDY

5.1 Introduction

In the previous chapter, the model ability was compared to previous work drawn from the literature review. It has been proved the flexibility of the model in scheduling and optimizing repetitive activities projects. In this chapter, real case study shall be presented to prove the applicability of the model for real projects. The case study is representing a project has five serial repetitive activities for constructing water channel in Riyadh city in King of Saudi Arabia.

5.2 Case Study

5.2.1 Project Overview

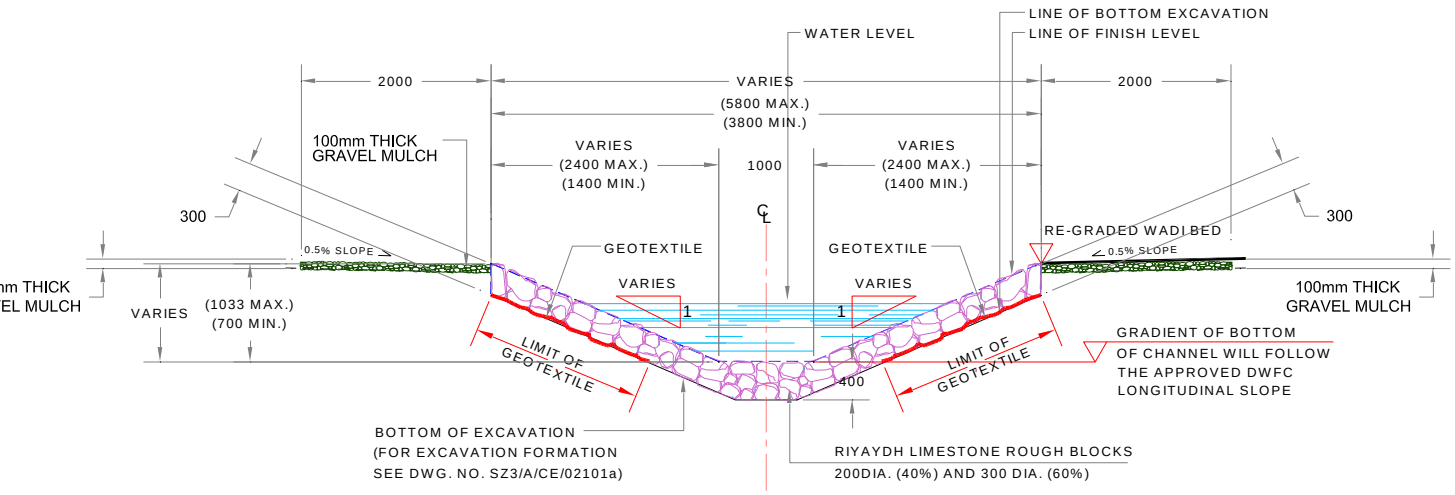
The case study has been implemented in this research is "Wadi Hanifa Restoration Project" in Saudi Arabia in Riyadh extracted from Masters of Science thesis by Al-Taweel, S. (2007). The project client is ArRiyadh Development Authority and the designer is Buro Happold from Brittan and the architecture from Canada "Moriyama and Teshima ". Wadi Hanifah is deemed as the most important natural landmark in Riyadh area. It spreads over an area exceeded 120 sq-km penetrating Riyadh-city. The valley is flowing down from northwestern towards southeastern. More than forty rivers are pouring on this Wadi which is still contains the remains of traditional environment features at the area, such as villages, parks, and farms. This valley is overflow with agricultural, inheriting, entertaining ingredients that assist in its developing as entertaining, agricultural and cultural center for the city-dwellers.

The higher commission for Riyadh development has established a strategy for investing and developing Wadi Hanifah since 1407H for the sake of maintaining its natural environment and prohibiting human destructive activity and preparing it as a natural drainage for water. It could also be used as an entertaining summer resort. The commission considers the valley as an environmental protectorate as well as developing area under its supervision. According to above mentioned, the commission performed visibility study including water resources, land, wildlife, land ownership and its employment, the existing farms in the valley, inheriting and entertaining ingredients,

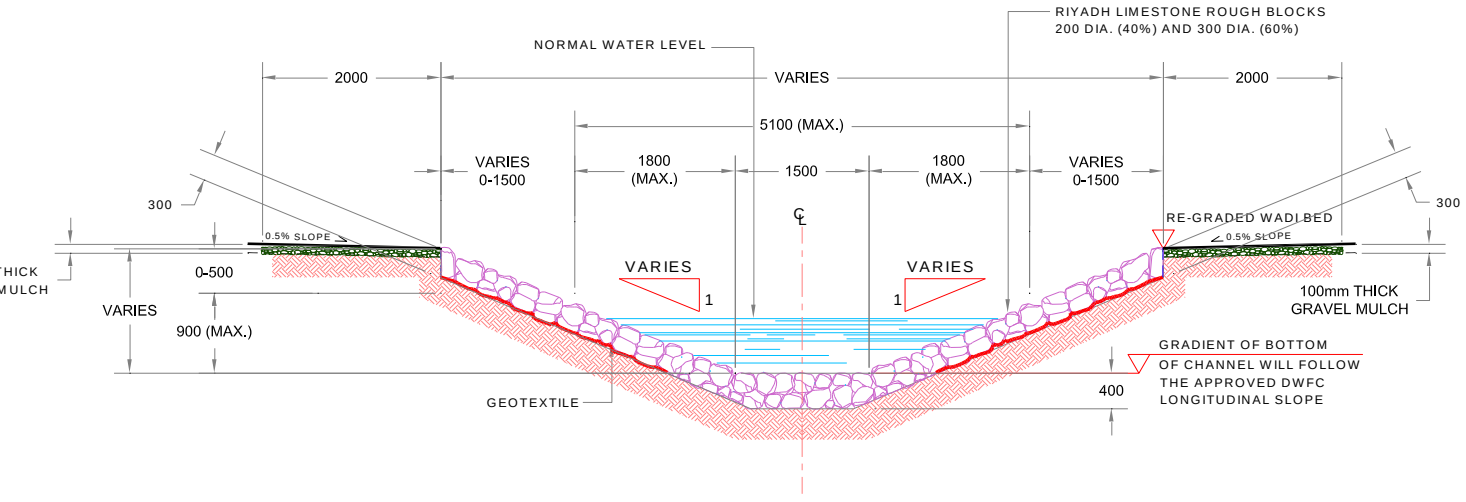
traffic, air and water pollution and water livings. The study resulted in creation of a comprehensive strategy for developing the valley. Finally this strategy approved by the commission in 1415H. The strategy is based on a number of policies, organizations, procedures and works in order to achieve the goals above mentioned. The strategy aims to cease environmental destruction to the valley by maintaining and developing its resources as well as establishing organizational diagrams for the usage of the land in the valley, providing entertaining facilities and completing the basic structure in the valley area in order to encourage investment in developmental projects. Accordingly, HRH, the chief of higher commission for developing Riyadh-city order to constitute several committees for studying the destructive factors that affect the environment of the valley and suggesting adequate solutions. Now, most of the committees has achieved their successfully. In 1419H the commission has decided to form a committee to set up a joint venture to develop the valley. The project consist of four zones, zone 3 was awarded to Al-Mashric Company. Zone 3 has been divided to three Areas (Area A, Area B and Area C). Area A and C consist of Excavation of water channel and Area B represent the bioremediation structure. In this research Area A and B water channel stage shall be scheduled as repetitive project.

5.2.2 Project Data

Project activities for water channel stage consist of repetitive activities that are repeated for all units. The channel is divided into four types relating to their dimension, (type 1, type 2, type 3 and type 4). The cross sections details for all channels are shown in Figs. 5.1 & 5.2. In channel construction stage, there are five activities repeated in 21 units as shown in Table 5.1. The quantities for activities in all units are shown in Table 5.2.

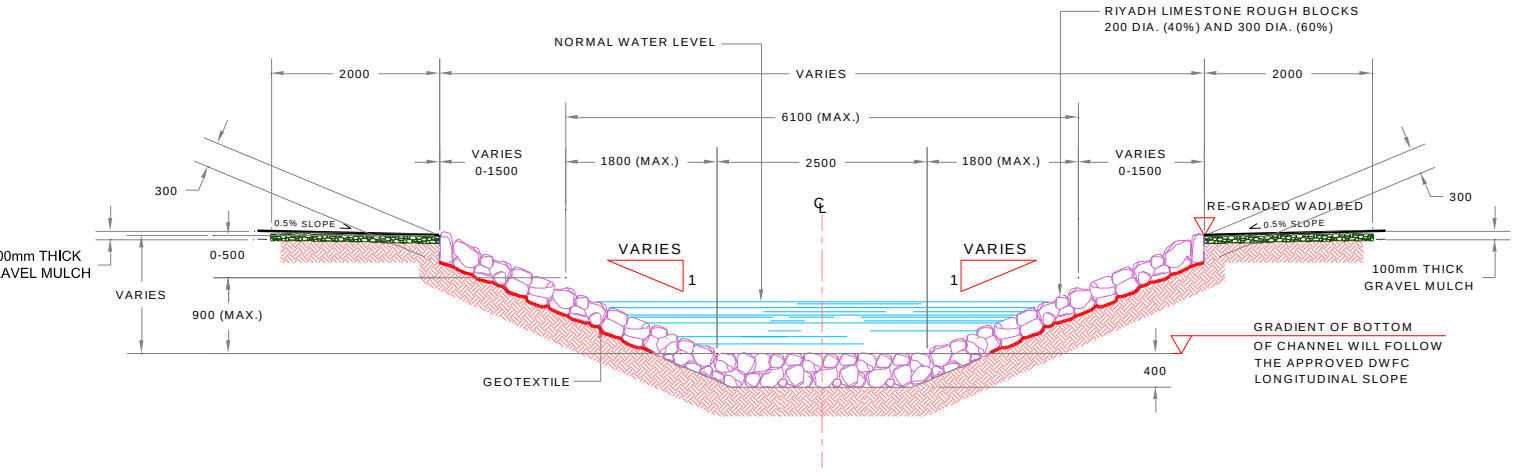


CHANNEL TYPE 1

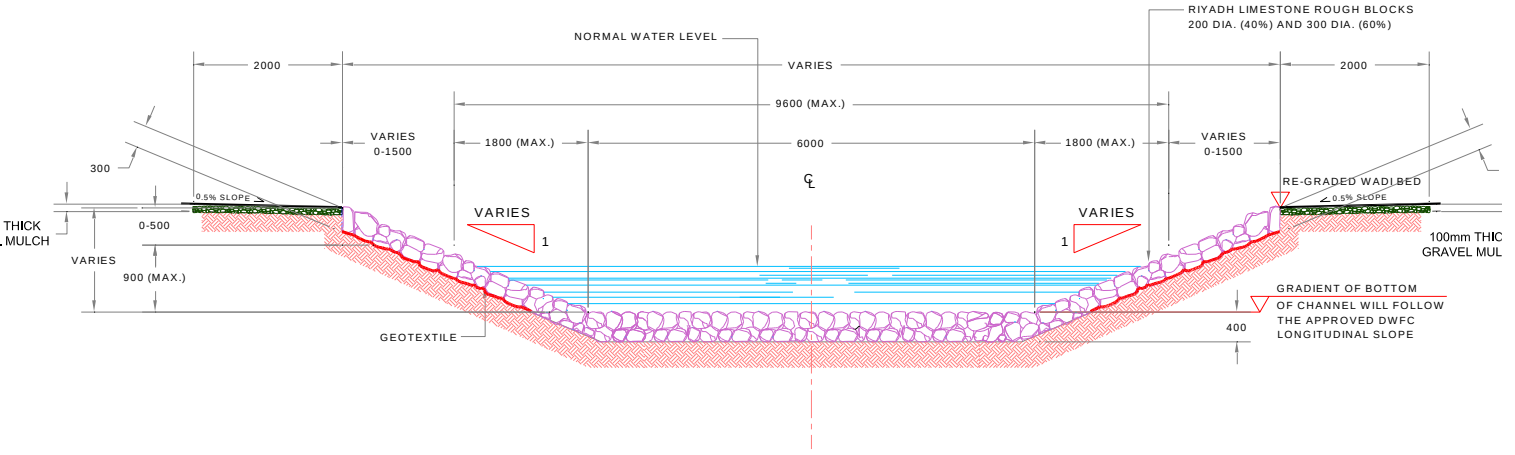


CHANNEL TYPE 2

Figure 5.1: Cross Section of Water Channel for Type 1 and 2.



CHANNEL TYPE 3



CHANNEL TYPE 4

Figure 5.2: Cross Section of Water Channel for Type 3 and 4

Table 5.1: Repetitive Activities of the Case Study

ID	Activity Description	Predecessor
1	Site Clearance	-
2	Channel Excavation	1
3	Geotextile Laying	2
4	Stone Laying	3
5	Mulshstone Laying	4

Table 5.2: Project Data

			Repetitive Activities					
Repetitive Unit Data			Site Clearance		Channel Excavation		Geotextile laying	
Chainage Stations	Repetitive Units No.	Channel	Quantities	Units	Quantities	Units	Quantities	Units
Ch.36.38 to Ch.37.00	1	Type 1	*19000*	M^2	*7500*	M^3	*6200*	M^2
Ch.37.00 to Ch.38.00	2	Type 2	*20000*		*7600*		*6200*	
Ch.38.00 to Ch.39.00	3	Type 2	*20000*		*7000*		*6200*	
Ch.39.00 to Ch.40.00	4	Type 2	*19000*		*7600*		*6200*	
Ch.40.00 to Ch.41.00	5	Type 2	*19000*		*7500*		*6200*	
Ch.41.00 to Ch.42.00	6	Type 2	*20000*		*7500*		*6300*	
Ch.43.00 to Ch.44.00	7	Type 3	*19000*		*7500*		*6200*	
Ch.44.00 to Ch.45.00	8	Type 3	*20000*		*7500*		*6200*	
Ch.45.00 to Ch.46.00	9	Type 3	*19000*		*7500*		*6200*	
Ch.46.00 to Ch.47.00	10	Type 3	*19000*		*7500*		*6200*	
Ch.47.00 to Ch.48.00	11	Type 3	*19000*		*7400*		*6300*	
Ch.48.00 to Ch.49.00	12	Type 3	*18000*		*7500*		*6200*	
Ch.49.00 to Ch.49.50	13	Type 3	*19000*		*7500*		*6200*	
Ch.49.50 to Ch.50.00	14	Type 4	*19000*		*7500*		*6200*	
Ch.50.00 to Ch.50.50	15	Type 4	*20000*		*7500*		*6200*	
Ch.50.50 to Ch.51.00	16	Type 4	*19000*		*7600*		*6200*	
Ch.51.00 to Ch.51.50	17	Type 4	*19000*		*7500*		*6200*	
Ch.51.50 to Ch.52.00	18	Type 4	*19000*		*7500*		*6200*	
Ch.52.00 to Ch.52.50	19	Type 4	*19000*		*7600*		*6200*	
Ch.52.50 to Ch.53.00	20	Type 4	*19000*		*7500*		*6300*	
Ch.53.00 to Ch.53.50	21	Type 4	*20000*		*7600*		*6300*	

Table 5.2: Project Data (Continued)

			Repetitive Activities			
Repetitive unit data			Stone laying		Mulshstone laying	
Chainage Station	**Repetitive units No.**	**Channel Type**	**Quantities**	**Units**	**Quantities**	**Units**
Ch.36.38 to Ch.37.00	1	Type 1	***3300***	M^3	***3500***	M^3
Ch.37.00 to Ch.38.00	2	Type 2	***3400***		***3600***	
Ch.38.00 to Ch.39.00	3	Type 2	***3400***		***3600***	
Ch.39.00 to Ch.40.00	4	Type 2	***3300***		***3500***	
Ch.40.00 to Ch.41.00	5	Type 2	***3300***		***3500***	
Ch.41.00 to Ch.42.00	6	Type 2	***3300***		***3600***	
Ch.43.00 to Ch.44.00	7	Type 3	***3300***		***3500***	
Ch.44.00 to Ch.45.00	8	Type 3	***3400***		***3500***	
Ch.45.00 to Ch.46.00	9	Type 3	***3300***		***3500***	
Ch.46.00 to Ch.47.00	10	Type 3	***3300***		***3400***	
Ch.47.00 to Ch.48.00	11	Type 3	***3300***		***3500***	
Ch.48.00 to Ch.49.00	12	Type 3	***3300***		***3500***	
Ch.49.00 to Ch.49.50	13	Type 3	***3300***		***3400***	
Ch.49.50 to Ch.50.00	14	Type 4	***3400***		***3500***	
Ch.50.00 to Ch.50.50	15	Type 4	***3300***		***3500***	
Ch.50.50 to Ch.51.00	16	Type 4	***3300***		***3500***	
Ch.51.00 to Ch.51.50	17	Type 4	***3300***		***3500***	
Ch.51.50 to Ch.52.00	18	Type 4	***3300***		***3500***	
Ch.52.00 to Ch.52.50	19	Type 4	***3300***		***3500***	
Ch.52.50 to Ch.53.00	20	Type 4	***3300***		***3500***	
Ch.53.00 to Ch.53.50	21	Type 4	***3300***		***3500***	

5.2.3 Project Scheduling

To implement the case study, four construction methods was applied that can be assigned to each activity. All construction methods have the different production rates and direct costs as shown in Table 5.3. Through implementing the model in this research, best combination of construction methods which are assigned to the different activities shall be obtained achieving minimum project duration, cost, interruptions and units delivery dates.

Table 5.3: Construction Methods Production Rates and Costs

ID	Activity description	Method 1		Method 2		Method 3		Method 4	
		P.R.	Cost ($)	P.R.	Cost ($)	P.R.	Cost ($)	P.R.	Cost ($)
1	Site clearance	2000	1850	1900	1650	1900	1600	1800	1500
2	Channel excavation	700	3900	700	3800	700	3700	700	3300
3	Geotextile laying	600	1100	650	1000	550	900	550	850
4	Riyadh lime Stone laying	300	2500	300	2300	300	2200	300	2000
5	Gravel mulch laying	300	1650	300	1500	280	1400	270	1300

Table 5.3 illustrates the transportation speed (Km/day) and cost per Km for each construction method.

Table 5.4: Construction Methods Transportation Speed and Costs

ID	Activity description	Method 1		Method 2		Method 3		Method 4	
		Speed	Cost ($)	Speed	Cost ($)	Speed	Cost ($)	Speed	Cost ($)
1	Site clearance	8	10	10	11	8	9	9	10
2	Channel excavation	5	10	5	9	6	10	6	8
3	Geotextile laying	10	5	10	6	8	5	9	7
4	Riyadh lime Stone laying	9	7	10	7	10	6	8	6
5	Gravel mulch laying	8	10	10	10	9	7	10	8

The project indirect cost was 3000 $ / day and the data have been entered to the model by the same way as example in chapter 4. Then scheduling and optimization model was applied to the project. Selected results obtained are shown in Table 5.5. The model proposed various schedules with maximum and minimum project total duration of 211 and 194 days respectively. Also with maximum and minimum project total cost of LE3237487 and LE3164646 respectively. Finally with maximum and minimum total project interruptions of 130 and 60 days respectively.

Table 5.5: Case Study Selected Results

	1	2	3	4	5	6	7	8	9
Total Cost (LEX1000)	3179	3216	3614	3218	3221	3216	3217	3209	3237
Total Duration	211	194	207	204	189	207	200	200	208
Total Interruption	91	109	130	88	84	106	102	113	60

The total crews' interruptions for the proposed solution are relatively high; this is due to the large number of possible schedules with different crews assigned to different units

minimizing the total project schedule as much as possible. Table 5.6 shows the Pareto-compromise solution and best alternative for the case study.

Table 5.6: Case Study Pareto-Compromise and Best-Alternative Solutions

Solution	Project Duration (days)	Project Cost ($)	Project Interruptions (days)
Pareto-Compromise Solution	202	3216573	94
Best-Alternative Solution	211	3179204	91

It is significantly important to annotate that the computer model calculations took more than 96hours; this is due to the large solution space for the case study which expands to 4^105 possible solution. This vast feasible solutions space takes more time for the GA's calculation to reach convergence and define the Pareto optimum scheduling set. This can be overcome by using parallel computation or coarse grain computation approaches in larger projects.

Moreover, the optimum solution set outcome depends on the initial population and the mutation probability, thus it may reaches a local optimum solution in larger problems with low mutation probability. To avoid local optimum solutions, the mutation probability was increased to 0.85-0.9. However, some other related techniques (i.e. Ant Colony or Particle Swarm) can be used to reach the optimum set more effectively avoiding local optimum solutions and with much less calculation time.

5.3 Summary and Conclusions

In this chapter, real case study was implemented by the model for the two parts of the model, scheduling and optimization. It was observed the flexibility of the model in dealing with the data of the case study. It is noted that optimization process is very important help decision maker to choose between different schedules that minimize total project duration, total project cost and total project interruptions.

CHAPTER 6

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

6.1 Summary

In this research, a flexible and dynamic model for scheduling and optimizing repetitive activities project has been developed. The model has been implemented on MS Project with VBA macro program. The model provides a schedule complies with precedence relationship using different relationship types and assigned construction method constraints. In addition, it considers the impact of following practical factors:

- Assigned construction methods transportation duration.
- Assigned construction methods transportation cost.
- Multiple crews assigned to work simultaneously.
- Accounting for project total cost and duration.
- Accounting for interruption for crews assigned to repetitive activities.
- Accounting for units' delivery dates delays.
- Optimization among several alternatives construction methods for each activity.

For each activity in a repetitive unit, the model identifies the scheduled start and finish times depending on the assigned construction methods' crew for each unit in every activity and interruption days for each crew, if any, as well as transportation duration and costs from the previous unit to the current one. The model also identifies the project duration, total direct cost and total indirect cost. The model has been developed on a MS Project by using Visual Basic Application (VBA) macros. The model employs non-traditional optimization technique, Genetic-Algorithm, that has powerful random search capabilities. The model sorts the output possible schedules to Pareto Fronts to determine the optimum set of schedules for the decision maker to choose from. The optimization process identifies the combination of construction methods that achieves the following objectives:

1- Minimize the total project cost.
2- Minimize the total project duration.
3- Minimize the total project interruptions.
4- Minimize the total units' delivery date delays.

6.2 Conclusions

In this research, a scheduling and optimization model for repetitive activities projects was developed. The developed model can be used to optimize construction schedules based on multiple objectives. Some remarks were concluded and listed below:

- Using different construction methods for the same activity type strategy creates a more flexible schedule that respects resource and precedence relationship, especially in large number of repetitive units.
- Transportation duration and cost of construction methods can affect the selection of the optimum schedule significantly.
- Multi-objective optimization for repetitive activities projects can be highly beneficial for the construction industry, avoiding future problems such as project's cost, duration and units' delivery dates delays.
- Genetic Algorithms technique is efficient in solving multi-objective optimization problems when it is integrated with Pareto Front sorting. It showed in the research its capability to search for the optimum solution set in a moderate large solution space in an efficient manner.

6.3 Recommendations and Future Work

The Proposed model can be implemented to different construction repetitive activities projects effectively. It can aid the decision maker with proposing different repetitive activities project schedules to select the most appropriate one with respect to the project at hand.

The presented model in this research could be improved considering the following points:

- Integrate the non-repetitive activities and repetitive activities in one Scheduling and Optimization model.
- Integrate the ability to change the work order among repetitive units.
- Experiment different evolutionary algorithms optimization techniques that discover the solution space more effectively and efficiently (Particle Swarm Optimization, Ant Colony and Shuffled Frog Leaping algorithms).

Reference list

- Al-Taweel, S. (2007), "*Repetitive Projects Scheduling and Optimization Using Genetic Algorithms", Masters of Science thesis, Arab Academy for Science and Technology, Alexandria, Egypt*.

- Ammar, M. and Elbeltagi, E. (2001), "Algorithm for Determining Controlling Path Considering Resource Continuity", *Journal of Computing in Civil Engineering*, 15(4), 292-298.

- Arditi, D., Tokedmir, O. and Suh, K. (2002), "Challenges in Line-of-Balance Scheduling", *Journal of Construction Engineering and Management*, 128(6), 545-556.

- Baumgartner, U., Magele, C. and Renhart, W. (2004), "Pareto Optimality and Particle Swarm Optimization", *IEEE Transactions on Magnetics*, 40, 1172-1175.

- Beradi, L., Giustolisi, O., Savic, D. and kapelan, Z. (2009), "An effective mulit-objective approach to prioritization of sewer pipe inspection", *Journal of Water Science & Technology*, 60, 841-850.

- Chrzanowski E. and Johnston, D. (1988), "Application of Linear Scheduling", *Journal of Construction Engineering and Management*, 112, 476-491.

- Cole, L. J. R. (1991), "Construction Scheduling: Principles, practices, and six case studies", *Journal of Construction Engineering and Management*, 117(4), 579-588.

- Dehuri, S. and Cho, S. (2009), "Multi-criterion Pareto base particle swarm optimized polynominal neural network for classification: A review and state-of-the-are projects", *Computer Science review*, 3, 19-40.

- Duffy, G., Oberlender, G. and Jeong, D. (2011), "Linear Scheduling Model with Varying Production Rates", *Journal of Construction Engineering and Management*, 137(8), 574-582.

- Elbeltagi, E. Hegazy, T. and Grierson, D. (2005), "Comparison among five evolutionary-based optimization algorithms", *Advanced Engineering Informatics, 1*9, 43-53.

- Elbelatgi, E., and ElKassas E.M. (2008), "Cost Optimization of Projects with Repetitive Activities Using Genetic Algorithms", *The Sixth International Conference on Engineering Computational Technology*, Paper 66.

- Elbeltagi, E. Hegazy, T. and Grierson, D. (2010)," A New Evolutionary Strategy for Pareto Multi-Objective Optimization", *The seventh International Conference on Engineering Computional Technology*, Paper 99.

- El-Rayes, K., and Moselhi, S. (1998), "Resource-driven scheduling of repetitive activities", *Construction Management Economics*, 16, 433-446.

- Feng, C., Liu, L., and Burns, S. (1997),"Using genetic algorithms to solve construction time-cost trade-off problems", *Journal of Computation in Civil Engineering, 11(3), 184–189.*

- Harris, R. and Ioannou, P. (1998), "Scheduling Projects with Repeating Activities", *Journal of Construction Engineering and Management. 124(4). 269-278.*

- Harmelink, D.J. (1995), "Linear Scheduling Model: The development of a linear scheduling model with micro computer applications for highway construction control", *PhD Thesis*, Iowa State Univ., Ames, Iowa.

- Harmelink, D. and Rowings, J. (1998), "Linear Scheduling Model: Development of Controlling Activity Path", *Journal of Construction Engineering and Management. 124(4), 263-268.*

- Hegazy, T., and Ayed, A. (1999), *"*Simplified spreadsheet solutions: Models for critical path method and time–cost trade-off analysis", *Cost Engineering, 417, 26–33.*

- Hegazy, T., Elhakeem, A. and Elbeltagi, E. (2004). *"*Distributed Scheduling Model for Infrastructure Networks", *Journal of Construction Engineering and Management, 130(2), 160-167.*

- Hegazy, T. and Kamarah, E. (2008), "Efficient repetitive scheduling for high-rise construction", *Journal of Construction Engineering and Management, 134(4), 253-264.*

- Holland, J. H. (1975), *"Adaptation in natural and artificial systems*", *University of Michigan Press, Ann Arbor, Mich.*

- Huang, R. and Sun, K. (2006), "Non-Unit-Based Planning and Scheduling of Repetitive Construction Projects". *Journal of Construction Engineering And Management*, 132(6), 585-597.

- Hyari, K. and El-Rayes, K. (2006), "Optimal Planning and Scheduling for Repetitive Construction Projects", *Journal of Management in Engineering, ASCE.* 22(1). 11-19.

- Ipsilandis, P. G. (2007), "Multiobjective Linear Programming Model for Scheduling Linear Repetitive Projects", *Journal of Construction Engineering and Management.* 133(6), 417-424.

- Johnosn, D. (1981), "Linear Scheduling Method for Highway Construction", *Journal of Construction Division. ASCE,*107, 247-261.

- Li, H., and Love, P. (1997*) "*Using improved genetic algorithms to facilitate time-cost optimization", *Journal of Construction Engineering and Management, 123(3), 233–237.*

- Li, H. and Zhang, Q. (2009), "Multiobjective Optimization Problems with Complicated Pareto Sets, MOEA/D and NSGA-IIO", *IEEE Transactions on Evolutionary Computation*, 13, 284-302.

- Lumsden, P. (1968), "Using Machine Learning and GA to Solve Time-Cost Trade-Off Problems", *Journal of Construction Engineering and Management,* 123, 233-237.

- Mattila, K. G. (1997), *"Resource Leveling of Linear Schedules; A Mathematical Approach using Integer Linear Programming", PhD thesis. Purdue Univ., West Lafayette, Ind.*

- Mattila, K. G. and Park, A. (2003), "Comparison of Linear Model and repetitive Scheduling Method", *Journal of Construction Engineering and Management*, 129(1), 56-64.

- Moselhi, O. and Hassanein, A. (2003), "Optimized Scheduling of Linear Projects", *Journal of Construction Engineering and Management*, 129(6), 664-673.

- MS Project (2007), MS Project Reference Manual, Microsoft Corporation.

- Rahbar, F.F., and Rowings, J.E. (1992), "Repetitive activity scheduling process", *Trans., Am. Assn. Cost Eng.*, 2, O.5.1-O.5.8.

- Reda, R. M. (1990), "RPM: Repetitive project modeling", *Journal of Construction Engineering and Management, ASCE.* 116(2), 316-330.

- Sanad, H. (2011), "*Optimum Analysis of Construction Projects With Nonlinear Cash Flow*", *Doctor of Philosophy Thesis, Tanta University, Egypt.*

- Stradel, O. and Cacha, J. (1982), "Time space scheduling method", *Journal of Construction Division. ASCE,* 108, 445-457.

- Suhail, S.A., and Neale, R.H. (1994), "CPM/LOB: New methodology to integrate CPM and line of balance", *Journal of Construction Engineering and Management,* 120(3), 667-684.

- Vorster, M., Beliveau, Y. and Bafna, T. (1992), "Linear Scheduling and Visualization". *Transportation Research Record.* 1351, 32-39.

- Wassef, N. and Hegazy, T. (2001), "Cost Optimization in Projects with Repetitive Non-Serial Activities", *Journal of Construction Engineering and Management,* 127, 183-191.

- Yamin, R. and Harmelink, D. (2001), “Comparison of Linear Scheduling Model (LSM) And Criticalpath Method (CPM)”, *Journal of Construction Engineering and Management*, 127, 374-381.

Appendix I – MS Project VBA Scheduling and Optimization Model

```
Dim AllTasks(1 To 2000) As ProjectTask
Dim AllChromosomes As Collection
Dim AllChr(1 To 1001) As ChromosomInfo
Dim Paretoset(1 To 20000) As ChromosomInfo
Dim AllChild(1 To 1000) As ChromosomInfo
Dim PrevParetoSet(1 To 20000) As ChromosomInfo
Dim ParetoFrist(1 To 20000) As ChromosomInfo
Public NChild As Integer
Public ts As Tasks
Public Acrew As CrewInfo
Public X As Integer
Public Y As Integer
Public CN As Integer
Public i As Integer
Public Agene As GeneInfo
Public GN As Integer 'Gene number
Public GC As Double 'Gene cost
Public GCrewCost As Double 'Crew Cost
Public TotalGenesCrewCost As Double 'total genes crew cost
Public ChrTotalCost As Double
Public K As Integer
Public Interruption As Variant
Public DeliveryDelay As Date
Public Act As Integer
Public Unit As Integer
Public MinCost As Double
Public MinInt As Double
Public DefStartDate As Date
Public Trail As Integer
Public TrailMax As Integer
Public NoSolutionS As Collection
Public Indirect_cost As Double
Public ActivityIndex As Integer
Public TotalTransportationCost As Double
Public PrevUnit As Integer
Public MaxDD As Double
Public MaxDur As Double
Public MaxInter As Double
Public MaxCost As Double
Public NENCurrent As Double
Public NENPrev As Double
Public Best1 As Integer
Public Best2 As Integer
Public MC As Integer
Public MD As Integer
Public MI As Integer
Public MDD As Integer
Public CrossoverValue As Integer
Public ParetoOpt As Integer
Public paretoValue As Double
Public paretoCheck As Double
Public NoParetoSol As Integer
```

```
        'CROSSOVER
Public SinglePoint As Integer
Public Parent1 As Integer
Public Parent2 As Integer
Public Parent3 As Integer
Public Parent4 As Integer
Public Offspring1 As Integer
Public Offspring2 As Integer
Public population As Integer
Public CopyChromo As Integer
Public PasteChromo As Integer
Public MO As Double
Public CrossIndex As Double
Public A As Double
Public B As Double
Public Identical As String
Public Identical_ChecK As String
Public mutationindex As Double
Public ConverIndex As Double
Public ParetoIndex1 As Double
Public Paretoindex2 As Double
Public ParetoCount As Integer
Public PrevParetoCount As Integer
Public PGC As Double
Public PDD As Double
Public PD As Double
Public PI As Double
Public PGN As Integer
Public AllSolutions As Integer
Public Count1 As Integer
Public Count2 As Integer
Public Count3 As Integer
Public Count4 As Integer
Public ParetoFrontIndex As Integer
Public Rel_ParetoFrontIndex As Double
Public Distance As Long
Public CostDistance As Double
Public CDistanceCheck As Double
Public InterDistance As Double
Public IDistanceCheck As Double
Public DurDistance As Double
Public DDistanceCheck As Double
Public DelayDistance As Double
Public DDDistanceCheck As Double
Public CounterStop As Integer
Public checkCounter As Long
Public Max_Reached As String
'selection probabilities
Public totmerit As Double
Public summerit As Double
Public Deadline As Integer
Public R As Variant
```

```
Public Sub CommandButton1_Click()

Randomize
Dim PauseTime, Start, Finish, TotalTime
Dim CrewStart(1 To 5, 1 To 4) As Date
Dim CrewStartCheck(1 To 5, 1 To 4) As String
ParetoCount = 1
Dim M As Integer
Dim N_Gener As Integer
NChild = 0
checkCounter = 0
MinCost = 1E+16
MinDur = 1E+17
MinDD = 1E+16
MinInt = 1E+16

Start = Timer

Act = TextBox1.Value
Unit = TextBox2.Value
population = TextBox4.Value
mutationindex = TextBox6.Value
CrossIndex = TextBox5.Value
Indirect_cost = TextBox3.Value


TrailMax = 500

Set ts = ActiveProject.Tasks

Application.SelectAll
Application.SaveSheetSelection
DefStartDate = ts(1).Start1

ReDim TransportationDuration(1 To Act, 1 To Unit, 1 To 21, 1 To 21) As Double
' (activity, crew#, current Unit, Prev Unit)
ReDim TransportationCost(1 To Act, 1 To Unit, 1 To 21, 1 To 21) As Integer


X = 1
Do
'read aTask (copy of project data base)
Set AllTasks(X) = New ProjectTask
AllTasks(X).Name = ts(X).Name
AllTasks(X).FixedCost = ts(X).Cost
If ts(X).Finish1 <> "NA" Then
AllTasks(X).RequiredEndDate = ts(X).Finish1
End If
Set AllTasks(X).CrewData = New Collection
If ts(X).Duration1 <> "0" Then
Set Acrew = New CrewInfo
Acrew.Duration = ts(X).Duration1
Acrew.Cost = ts(X).Cost1
AllTasks(X).CrewData.Add Acrew
End If
If ts(X).Duration2 <> "0" Then
Set Acrew = New CrewInfo
```

```
Acrew.Duration = ts(X).Duration2
Acrew.Cost = ts(X).Cost2
AllTasks(X).CrewData.Add Acrew
End If
If ts(X).Duration3 <> "0" Then
Set Acrew = New CrewInfo
Acrew.Duration = ts(X).Duration3
Acrew.Cost = ts(X).Cost3
AllTasks(X).CrewData.Add Acrew
End If
If ts(X).Duration4 <> "0" Then
Set Acrew = New CrewInfo
Acrew.Duration = ts(X).Duration4
Acrew.Cost = ts(X).Cost4
AllTasks(X).CrewData.Add Acrew
End If
AllTasks(X).CrewNumber = AllTasks(X).CrewData.Count
X = X + 1
Loop Until X > Act * Unit


‘Determine transportation distance, duration and cost
Dim UnitDistance(1 To 21, 1 To 21) As Double
Dim CrewSpeed(1 To 5, 1 To 4) As Double
Dim CrewTransCost(1 To 5, 1 To 4) As Double

For Y = 1 To Unit
For X = Y + 1 To Unit
Call Get_Distance(UnitDistance)
Next X
Next Y

For Y = 1 To Act
For X = 1 To AllTasks(Y).CrewNumber
Call Get_Speed(CrewSpeed)
Next X
Next Y


For Y = 1 To Act
For X = 1 To AllTasks(Y).CrewNumber
Call Get_TransCost(CrewTransCost)
Next X
Next Y



For Y = 1 To Act
For X = 1 To AllTasks(Y).CrewNumber
For K = 1 To Unit
For i = K + 1 To Unit
TransportationDuration(Y, X, K, i) = UnitDistance(K, i) / CrewSpeed(Y, X)
Next i
Next K
Next X
Next Y
```

```
For Y = 1 To Act
For X = 1 To AllTasks(Y).CrewNumber
For K = 1 To Unit
For i = K + 1 To Unit
TransportationCost(Y, X, K, i) = TransportationDuration(Y, X, K, i) * CrewTransCost(Y,
X)
Next i
Next K
Next X
Next Y


        'Creating new Chromosmes :)
X = 1

Do
Set AllChr(X) = New ChromosomInfo
Set AllChr(X).Genes = New Collection
ChrTotalCost = 0
GCrewCost = 0
TotalGenesCrewCost = 0
Y = 1
i = 1
ActivityIndex = 1
Do Until Y > Act * Unit
If ActivityIndex > Act Then ActivityIndex = 1

i = Alltasks(x).Crewnumber

Set Agene = New GeneInfo
GN = Int((i * Rnd) + 1)
Agene.GeneValue = GN
GCrewCost = AllTasks(Y).CrewData(GN).Cost
Agene.CrewCost = GCrewCost
AllChr(X).Genes.Add Agene

TotalGenesCrewCost = TotalGenesCrewCost + GCrewCost
ActivityIndex = ActivityIndex + 1
Y = Y + 1
Loop

AllChr(X).TotalCost = TotalGenesCrewCost
X = X + 1
Loop Until X > population



' Creating dummy child pool

X = 1
Do
Set AllChild(X) = New ChromosomInfo
Set AllChild(X).Genes = New Collection
ChrTotalCost = 0
GCrewCost = 0
TotalGenesCrewCost = 0
Y = 1
```

```
i = 1
Do Until Y > Act * Unit
i = AllTasks(Y).CrewNumber
Set Agene = New GeneInfo
GN = Int((i * Rnd) + 1)
Agene.GeneValue = GN
GCrewCost = AllTasks(Y).CrewData(GN).Cost
Agene.CrewCost = GCrewCost
AllChild(X).Genes.Add Agene
TotalGenesCrewCost = TotalGenesCrewCost + GCrewCost
Y = Y + 1
Loop
AllChild(X).TotalCost = TotalGenesCrewCost

X = X + 1
Loop Until X > population

Label4.Caption = "Scheduling"

' Initial schedule

X = 1
Do
Call Schedule
Call Restore_Data
X = X + 1
Loop Until X > population



'pareto set

ParetoCount = 0
Call get_pareto_optimal(population)

  For X = 1 To population
     If AllChr(X).ParetoFront = 1 Then
        If ParetoCount > 0 Then   'check identical paretoset
     For i = 1 To ParetoCount
     GN = 1
        For Y = 1 To Act * Unit
        If Paretoset(i).Genes(Y).GeneValue = AllChr(X).Genes(Y).GeneValue Then GN
= GN + 1
        Next Y
     If GN = Act * Unit Then GoTo 121
     Next i
     GoTo 111
     End If

111     ParetoCount = ParetoCount + 1
        Set Paretoset(ParetoCount) = New ChromosomInfo
        Set Paretoset(ParetoCount).Genes = New Collection
          Y = 1
          Do Until Y > Act * Unit
          Set Agene = New GeneInfo
          GN = AllChr(X).Genes(Y).GeneValue
          Agene.GeneValue = GN
          GC = AllChr(X).Genes(Y).CrewCost
```

```
            Agene.CrewCost = GC
            Paretoset(ParetoCount).Genes.Add Agene
            Paretoset(ParetoCount).PretoSol = Pareto
            Y = Y + 1
            Loop
         Paretoset(ParetoCount).TotalCost = AllChr(X).TotalCost
         Paretoset(ParetoCount).TotalDelays = AllChr(X).TotalDelays
         Paretoset(ParetoCount).TotalDuration = AllChr(X).TotalDuration
         Paretoset(ParetoCount).TotalIntrruptions = AllChr(X).TotalIntrruptions
         Paretoset(ParetoCount).Note = AllChr(X).Note
         Paretoset(ParetoCount).ParetoFront = 1

      End If
121   Next X

Call selection_prob

M = 0
N_Gener = M

Trail = 1

Label4.Caption = "Optimizing"


'Optimization Module

Do While Trail < TrailMax
NChild = 0
M = M + 1
120     Call chro_select(Parent1, Parent2)
NChild = NChild + 1
MO = Rnd
If MO > CrossIndex Then
   If AllChr(Parent1).InvParetoFront > AllChr(Parent2).InvParetoFront Then
      Set AllChild(NChild) = New ChromosomInfo
      Set AllChild(NChild).Genes = New Collection

         For Y = 1 To Act * Unit
            Set Agene = New GeneInfo
            GN = AllChr(Parent1).Genes(Y).GeneValue
            Agene.GeneValue = GN
            GC = AllChr(Parent1).Genes(Y).CrewCost
            Agene.CrewCost = GC
            AllChild(NChild).Genes.Add Agene
         Next Y
      AllChild(NChild).TotalCost = AllChr(Parent1).TotalCost
      AllChild(NChild).Note = "None"
   Else
      Set AllChild(NChild) = New ChromosomInfo
      Set AllChild(NChild).Genes = New Collection

         For Y = 1 To Act * Unit
            Set Agene = New GeneInfo
            GN = AllChr(Parent2).Genes(Y).GeneValue
            Agene.GeneValue = GN
            GC = AllChr(Parent2).Genes(Y).CrewCost
            Agene.CrewCost = GC
```

```
            AllChild(NChild).Genes.Add Agene
          Next Y
       AllChild(NChild).TotalCost = AllChr(Parent2).TotalCost
       AllChild(NChild).Note = None
   End If
If NChild < population Then GoTo 120
Else

Call CrossOver(Parent1, Parent2, NChild)
Call Mutation(mutationindex, NChild)
If NChild < population Then GoTo 120
End If


For X = 1 To population              'empty parents
   For Y = 1 To Act * Unit
      AllChr(X).Genes(Y).GeneValue = 0
   Next Y
Next X

For X = 1 To population              'copy offspring

   For Y = 1 To Act * Unit
      AllChr(X).Genes(Y).GeneValue = AllChild(X).Genes(Y).GeneValue
      AllChr(X).Genes(Y).CrewCost = AllChild(X).Genes(Y).CrewCost
   Next Y

   AllChr(X).TotalCost = AllChild(X).TotalCost
   AllChr(X).Note = AllChild(X).Note
Next X

For X = 1 To population                              'Empty children pool
         For Y = 1 To Act * Unit
         AllChild(X).Genes(Y).GeneValue = 0
         Next Y
Next X

For X = 1 To population
Call Schedule
Call Restore_Data
Next X
Call get_pareto_optimal(population)
Call selection_prob


   For X = 1 To population                        ' add pareto front 1 to pareto set
      If AllChr(X).ParetoFront = 1 Then
      If ParetoCount > 0 Then   'check identical paretoset
      For i = 1 To ParetoCount
      GN = 0
      For Y = 1 To Act * Unit
      If Paretoset(i).Genes(Y).GeneValue = AllChr(X).Genes(Y).GeneValue Then GN =
GN + 1
      Next Y
      If GN = Act * Unit Then GoTo 122
      Next i
      GoTo 112
      End If
```

```
112          ParetoCount = ParetoCount + 1
         If ParetoCount > 20000 Then
         ParetoCount = 1
         Max_Reached = True
         End If

         Set Paretoset(ParetoCount) = New ChromosomInfo
         Set Paretoset(ParetoCount).Genes = New Collection
            Y = 1
            Do Until Y > Act * Unit
            Set Agene = New GeneInfo
            GN = AllChr(X).Genes(Y).GeneValue
            Agene.GeneValue = GN
            GC = AllChr(X).Genes(Y).CrewCost
            Agene.CrewCost = GC
            Paretoset(ParetoCount).Genes.Add Agene
            Paretoset(ParetoCount).PretoSol = Pareto

            Y = Y + 1
            Loop

         Paretoset(ParetoCount).TotalCost = AllChr(X).TotalCost
         Paretoset(ParetoCount).TotalDelays = AllChr(X).TotalDelays
         Paretoset(ParetoCount).TotalDuration = AllChr(X).TotalDuration
         Paretoset(ParetoCount).TotalIntrruptions = AllChr(X).TotalIntrruptions
         Paretoset(ParetoCount).Note = AllChr(X).Note
         Paretoset(ParetoCount).ParetoFront = 1

      End If
122    Next X


Call get_pareto_optimal_final(population, ParetoCount)


'Concervgence Check

If M = 1 Then
PrevParetoCount = 0        'Create Prev. Pareto Set

For X = 1 To ParetoCount


      If PrevParetoCount > 0 Then   'check identical
      For i = 1 To PrevParetoCount
      GN = 0
      For Y = 1 To Act * Unit
      If PrevParetoSet(i).Genes(Y).GeneValue = Paretoset(X).Genes(Y).GeneValue Then
GN = GN + 1
      Next Y
      If GN = Act * Unit Then GoTo 123
      Next i
      GoTo 113
      End If

113       PrevParetoCount = PrevParetoCount + 1
         Set PrevParetoSet(PrevParetoCount) = New ChromosomInfo
```

```
        Set PrevParetoSet(PrevParetoCount).Genes = New Collection
           Y = 1
           Do Until Y > Act * Unit
           Set Agene = New GeneInfo

           GN = Paretoset(X).Genes(Y).GeneValue
           Agene.GeneValue = GN
           GC = Paretoset(X).Genes(Y).CrewCost
           Agene.CrewCost = GC
           PrevParetoSet(PrevParetoCount).Genes.Add Agene
           PrevParetoSet(PrevParetoCount).PretoSol = Pareto


           Y = Y + 1
           Loop
        PrevParetoSet(PrevParetoCount).TotalCost = Paretoset(X).TotalCost
        PrevParetoSet(PrevParetoCount).TotalDelays = Paretoset(X).TotalDelays
        PrevParetoSet(PrevParetoCount).TotalDuration = Paretoset(X).TotalDuration
        PrevParetoSet(PrevParetoCount).TotalIntrruptions =
Paretoset(X).TotalIntrruptions
        PrevParetoSet(PrevParetoCount).Note = Paretoset(X).Note
        PrevParetoSet(PrevParetoCount).ParetoFront = Paretoset(X).ParetoFront

123 Next X

GoTo 130
End If



Distance = 9999          'Distance calculations
CostDistance = 0
DelayDistance = 0
InterDistance = 0
DurDistance = 0
checkCounter = 0

MaxCost = 0
MaxDur = 0
MaxInter = 0
MaxDD = 0

NENCurrent = 999999
NENPrev = 999999

For X = 1 To ParetoCount     'determine the max value
If Paretoset(X).ParetoFront <> 1 Then GoTo 610

If Paretoset(X).TotalCost > MaxCost Then MaxCost = Paretoset(X).TotalCost
If Paretoset(X).TotalDelays > MaxDD Then MaxDD = Paretoset(X).TotalDelays
If Paretoset(X).TotalDuration > MaxDur Then MaxDur = Paretoset(X).TotalDuration
If Paretoset(X).TotalIntrruptions > MaxInter Then MaxInter =
Paretoset(X).TotalIntrruptions

610 Next X

For X = 1 To ParetoCount  'determine current NEN
If Paretoset(X).ParetoFront <> 1 Then GoTo 710
```

```
If MaxCost > 0 Then CostDistance = (Paretoset(X).TotalCost / MaxCost) ^ 2
If MaxDD > 0 Then DelayDistance = (Paretoset(X).TotalDelays / MaxDD) ^ 2
If MaxInter > 0 Then InterDistance = (Paretoset(X).TotalIntrruptions / MaxInter) ^ 2
If MaxDur > 0 Then DurDistance = (Paretoset(X).TotalDuration / MaxDur) ^ 2

If Sqr(CostDistance + DelayDistance + InterDistance + DurDistance) < NENCurrent
Then NENCurrent = Sqr(CostDistance + DelayDistance + InterDistance + DurDistance)
Best1 = X
710 Next X


For Y = 1 To PrevParetoCount    'determine max value for prev. pareto
If PrevParetoSet(Y).ParetoFront <> 1 Then GoTo 810

If PrevParetoSet(Y).TotalCost > MaxCost Then MaxCost = PrevParetoSet(Y).TotalCost
If PrevParetoSet(Y).TotalDelays > MaxDD Then MaxDD =
PrevParetoSet(Y).TotalDelays
If PrevParetoSet(Y).TotalDuration > MaxDur Then MaxDur =
PrevParetoSet(Y).TotalDuration
If PrevParetoSet(Y).TotalIntrruptions > MaxInter Then MaxInter =
PrevParetoSet(Y).TotalIntrruptions

810 Next Y

For Y = 1 To PrevParetoCount  'determine prev NEN
If PrevParetoSet(Y).ParetoFront <> 1 Then GoTo 910

If MaxCost > 0 Then CostDistance = (PrevParetoSet(Y).TotalCost / MaxCost) ^ 2
If MaxDD > 0 Then DelayDistance = (PrevParetoSet(Y).TotalDelays / MaxDD) ^ 2
If MaxInter > 0 Then InterDistance = (PrevParetoSet(Y).TotalIntrruptions / MaxInter) ^ 2
If MaxDur > 0 Then DurDistance = (PrevParetoSet(Y).TotalDuration / MaxDur) ^ 2

If Sqr(CostDistance + DelayDistance + InterDistance + DurDistance) < NENPrev Then
NENPrev = Sqr(CostDistance + DelayDistance + InterDistance + DurDistance)
Best2 = Y

910 Next Y

CostDistance = 0
DurDistance = 0
InterDistance = 0
DelayDistance = 0

If PrevParetoSet(Best2).TotalCost > 0 Then CostDistance =
((PrevParetoSet(Best2).TotalCost - Paretoset(Best1).TotalCost) /
PrevParetoSet(Best2).TotalCost)
If PrevParetoSet(Best2).TotalDelays > 0 Then DelayDistance =
((PrevParetoSet(Best2).TotalDelays - Paretoset(Best1).TotalDelays) /
PrevParetoSet(Best2).TotalDelays)
If PrevParetoSet(Best2).TotalDuration > 0 Then DurDistance =
((PrevParetoSet(Best2).TotalDuration - Paretoset(Best1).TotalDuration) /
PrevParetoSet(Best2).TotalDuration)
If PrevParetoSet(Best2).TotalIntrruptions > 0 Then InterDistance =
((PrevParetoSet(Best2).TotalIntrruptions - Paretoset(Best1).TotalIntrruptions) /
PrevParetoSet(Best2).TotalIntrruptions)

Distance = CostDistance + DelayDistance + DurDistance + InterDistance
```

```
If Distance < 0.0001 Then CounterStop = CounterStop + 1 Else CounterStop = 0
If CounterStop = 10 Then Trail = TrailMax

i = 1
For X = 1 To ParetoCount                        'new prev. paretoset
      For i = 1 To PrevParetoCount               'check identical
      GN = 0
      For Y = 1 To Act * Unit
      If PrevParetoSet(i).Genes(Y).GeneValue = Paretoset(X).Genes(Y).GeneValue Then
GN = GN + 1
      Next Y
      If GN = Act * Unit Then GoTo 124
      Next i
      GoTo 114

114       PrevParetoCount = PrevParetoCount + 1
         Set PrevParetoSet(PrevParetoCount) = New ChromosomInfo
         Set PrevParetoSet(PrevParetoCount).Genes = New Collection
            Y = 1
            Do Until Y > Act * Unit
            Set Agene = New GeneInfo

            GN = Paretoset(X).Genes(Y).GeneValue
            Agene.GeneValue = GN
            GC = Paretoset(X).Genes(Y).CrewCost
            Agene.CrewCost = GC
            PrevParetoSet(PrevParetoCount).Genes.Add Agene
            PrevParetoSet(PrevParetoCount).PretoSol = Pareto


            Y = Y + 1
            Loop
         PrevParetoSet(PrevParetoCount).TotalCost = Paretoset(X).TotalCost
         PrevParetoSet(PrevParetoCount).TotalDelays = Paretoset(X).TotalDelays
         PrevParetoSet(PrevParetoCount).TotalDuration = Paretoset(X).TotalDuration
         PrevParetoSet(PrevParetoCount).TotalIntrruptions =
Paretoset(X).TotalIntrruptions
         PrevParetoSet(PrevParetoCount).Note = Paretoset(X).Note
         PrevParetoSet(PrevParetoCount).ParetoFront = Paretoset(X).ParetoFront

124 Next X



130 Trail = Trail + 1

Loop
N_Gener = M

X = 1
For Y = 1 To ParetoCount
If Paretoset(Y).ParetoFront = 1 Then
          Set ParetoFrist(X) = New ChromosomInfo
         Set ParetoFrist(X).Genes = New Collection
            K = 1
            Do Until K > Act * Unit
            Set Agene = New GeneInfo
```

```
            GN = Paretoset(Y).Genes(K).GeneValue
            Agene.GeneValue = GN
            GC = Paretoset(Y).Genes(K).CrewCost
            Agene.GeneCost = GC
            ParetoFrist(X).Genes.Add Agene
            ParetoFrist(X).PretoSol = Pareto
            K = K + 1
            Loop
        ParetoFrist(X).TotalCost = Paretoset(Y).TotalCost
        ParetoFrist(X).TotalDelays = Paretoset(Y).TotalDelays
        ParetoFrist(X).TotalDuration = Paretoset(Y).TotalDuration
        ParetoFrist(X).TotalIntrruptions = Paretoset(Y).TotalIntrruptions
        ParetoFrist(X).Note = Paretoset(Y).Note
        ParetoFrist(X).ParetoFront = Paretoset(Y).ParetoFront
X = X + 1
End If
 Next Y
Finish = Timer
TotalTime = Finish - Start
Label4.Caption = "Run Time = " & TotalTime / 60
End Sub
```

```
Public Sub Schedule()

' Scheduling

i = 1
Y = 1
K = 1
ActivityIndex = 1
TotalTransportationCost = 0
ReDim CrewStart(1 To 5, 1 To 4) As Date
ReDim CrewStartCheck(1 To 5, 1 To 4) As String
ReDim TransportationDuration(1 To 5, 1 To 4, 1 To 21, 1 To 21) As Double                '
(activity, crew#, current Unit, Prev Unit)
ReDim TransportationCost(1 To 5, 1 To 4, 1 To 21, 1 To 21) As Integer

   ' Restart crew start time check
   Do Until ActivityIndex > Act
      For i = 1 To 4
      CrewStartCheck(ActivityIndex, i) = "no"
      Next i
      ActivityIndex = ActivityIndex + 1
   Loop

i = 1
ActivityIndex = 1
Interruption = 0
DeliveryDelay = 0
```

```
Do Until Y > Act * Unit
      i = AllChr(X).Genes(Y).GeneValue
   If i = 1 Then

   'Check if the crew started before or not
   If CrewStartCheck(ActivityIndex, i) = "started" Then
      K = Y - Act
         'Getting the previous unit and transporation cost and duration
         Do Until K < 1
         If ts(Y).Start1 = ts(K).Finish Then PrevUnit = K
         K = K - Act
         Loop
      ts(Y).Start1 = ts(Y).Start1 + TransportationDuration(ActivityIndex, i, Int((PrevUnit /
Act) + 0.99), Int((Y / Act) + 0.99))
      TotalTransportationCost = TotalTransportationCost +
TransportationCost(ActivityIndex, i, Int((PrevUnit / Act) + 0.99), Int((Y / Act) + 0.99))

      If ts(Y).Start > ts(Y).Start1 Then
      Interruption = Interruption + (ts(Y).Start - ts(Y).Start1) - 0.63

      ts(Y).Start1 = ts(Y).Start
      Else
      ts(Y).Start = ts(Y).Start1
      End If
   Else
   CrewStartCheck(ActivityIndex, i) = "started"
   ts(Y).Start1 = ts(Y).Start
   End If
   ts(Y).Duration = AllTasks(Y).CrewData(i).Duration



   'Changing the crews start time in the rest of the activities
   K = Y + Act
   Do Until K > Act * Unit
      ts(K).Start1 = ts(Y).Finish
      K = K + Act
   Loop


   ElseIf i = 2 Then

      'Check if the crew started before or not
   If CrewStartCheck(ActivityIndex, i) = "started" Then
    K = Y - Act
         'Getting the previous unit and transporation cost and duration
         Do Until K < 1
         If ts(Y).Start2 = ts(K).Finish Then PrevUnit = K
         K = K - Act
         Loop
      ts(Y).Start2 = ts(Y).Start2 + TransportationDuration(ActivityIndex, i, Int((PrevUnit /
Act) + 0.99), Int((Y / Act) + 0.99))
      TotalTransportationCost = TotalTransportationCost +
TransportationCost(ActivityIndex, i, Int((PrevUnit / Act) + 0.99), Int((Y / Act) + 0.99))

      If ts(Y).Start > ts(Y).Start2 Then
      Interruption = Interruption + (ts(Y).Start - ts(Y).Start2) - 0.63
```

```
    ts(Y).Start2 = ts(Y).Start
    Else
    ts(Y).Start = ts(Y).Start2
    End If
  Else
  CrewStartCheck(ActivityIndex, i) = "started"
  ts(Y).Start2 = ts(Y).Start
  End If
  ts(Y).Duration = AllTasks(Y).CrewData(i).Duration

  'Changing the crew start time in the rest of the activities
  K = Y + Act
  Do Until K > Act * Unit
    ts(K).Start2 = ts(Y).Finish
    K = K + Act
  Loop


  ElseIf i = 3 Then

    'Check if the crew started before or not
  If CrewStartCheck(ActivityIndex, i) = "started" Then
     K = Y - Act
       'Getting the previous unit and transporation cost and duration
       Do Until K < 1
       If ts(Y).Start3 = ts(K).Finish Then PrevUnit = K
       K = K - Act
       Loop
    ts(Y).Start3 = ts(Y).Start3 + TransportationDuration(ActivityIndex, i, Int((PrevUnit /
Act) + 0.99), Int((Y / Act) + 0.99))
    TotalTransportationCost = TotalTransportationCost +
TransportationCost(ActivityIndex, i, Int((PrevUnit / Act) + 0.99), Int((Y / Act) + 0.99))


    If ts(Y).Start > ts(Y).Start3 Then
    Interruption = Interruption + (ts(Y).Start - ts(Y).Start3) - 0.63
    ts(Y).Start3 = ts(Y).Start
    Else
    ts(Y).Start = ts(Y).Start3
    End If
  Else
  CrewStartCheck(ActivityIndex, i) = "started"
  ts(Y).Start3 = ts(Y).Start
  End If
  ts(Y).Duration = AllTasks(Y).CrewData(i).Duration

  'Changing the crew start time in the rest of the activities
  K = Y + Act
  Do Until K > Act * Unit
    ts(K).Start3 = ts(Y).Finish
    K = K + Act
  Loop

  Else


    'Check if the crew started before or not
  If CrewStartCheck(ActivityIndex, i) = "started" Then
```

```
        K = Y - Act
          'Getting the previous unit and transporation cost and duration
          Do Until K < 1
          If ts(Y).Start4 = ts(K).Finish Then PrevUnit = K
          K = K - Act
          Loop
       ts(Y).Start4 = ts(Y).Start4 + TransportationDuration(ActivityIndex, i, Int((PrevUnit /
Act) + 0.99), Int((Y / Act) + 0.99))
       TotalTransportationCost = TotalTransportationCost +
TransportationCost(ActivityIndex, i, Int((PrevUnit / Act) + 0.99), Int((Y / Act) + 0.99))

       If ts(Y).Start > ts(Y).Start4 Then
       Interruption = Interruption + (ts(Y).Start - ts(Y).Start4) - 0.63

       ts(Y).Start4 = ts(Y).Start
       Else
       ts(Y).Start = ts(Y).Start4
       End If
    Else
    CrewStartCheck(ActivityIndex, i) = "started"
    ts(Y).Start4 = ts(Y).Start
    End If
    ts(Y).Duration = AllTasks(Y).CrewData(i).Duration

    'Changing the crew start time in the rest of the activities
    K = Y + Act
    Do Until K > Act * Unit
       ts(K).Start4 = ts(Y).Finish
       K = K + Act
    Loop

    End If

ActivityIndex = ActivityIndex + 1
If ActivityIndex > Act Then ActivityIndex = 1
Y = Y + 1
Loop


‘Calculating Delivery Dates Delays
For Y = 1 To Act * Unit
  If ts(Y).Finish1 = NA Then GoTo 510
    If ts(Y).Finish > ts(Y).Finish1 Then DeliveryDelay = DeliveryDelay + (ts(Y).Finish -
ts(Y).Finish1)
510 Next Y

AllChr(X).TotalDuration = ActiveProject.ProjectFinish - ActiveProject.ProjectStart
AllChr(X).TotalIntrruptions = Interruption
AllChr(X).TotalDelays = DeliveryDelay
AllChr(X).TotalCost = AllChr(X).TotalCost + TotalTransportationCost + (Indirect_cost *
AllChr(X).TotalDuration)

End Sub
```

```
Public Sub Restore_Data()
Application.RestoreSheetSelection
Y = 1
Do Until Y > Act * Unit
If AllTasks(Y).CrewNumber = 1 Then
ts(Y).Start1 = DefStartDate
ElseIf AllTasks(Y).CrewNumber = 2 Then
ts(Y).Start1 = DefStartDate
ts(Y).Start2 = DefStartDate
ElseIf AllTasks(Y).CrewNumber = 3 Then
ts(Y).Start1 = DefStartDate
ts(Y).Start2 = DefStartDate
ts(Y).Start3 = DefStartDate
Else
ts(Y).Start1 = DefStartDate
ts(Y).Start2 = DefStartDate
ts(Y).Start3 = DefStartDate
ts(Y).Start4 = DefStartDate
End If
Y = Y + 1
Loop

Y = 1
Do Until Y > Act * Unit
ts(Y).ConstraintDate = "NA"
Y = Y + 1
Loop
End Sub
```

```
Public Sub CrossOver(Parent1, Parent2, NChild)
Randomize
                                        'perform crossover
 SinglePoint = Int(Rnd * (Act * Unit)) + 1                    'Select two crossover
points
GCrewCost = 0

 X = 1
Do Until X > SinglePoint
AllChild(NChild).Genes(X).GeneValue = AllChr(Parent1).Genes(X).GeneValue
AllChild(NChild).Genes(X).CrewCost = AllChr(Parent1).Genes(X).CrewCost
GCrewCost = GCrewCost + AllChr(Parent1).Genes(X).CrewCost
X = X + 1
Loop

Do Until X > Act * Unit
AllChild(NChild).Genes(X).GeneValue = AllChr(Parent2).Genes(X).GeneValue
AllChild(NChild).Genes(X).CrewCost = AllChr(Parent2).Genes(X).CrewCost
GCrewCost = GCrewCost + AllChr(Parent2).Genes(X).CrewCost
X = X + 1
Loop

AllChild(NChild).TotalCost = GCrewCost
AllChild(NChild).Note = "CrossOver"

End Sub
```

```
Sub get_pareto_optimal(population)
   'ReDim NoSolutionS(1) As Integer
   AllSolutions = False
   ParetoFrontIndex = 0
   For X = 1 To population
      AllChr(X).ParetoFront = 0
   Next X
   Count3 = 0: Count4 = 0
   Do
      ParetoFrontIndex = ParetoFrontIndex + 1
      'ReDim Preserve NoSolutionS(ParetoFrontIndex)
      For X = 1 To population
         If AllChr(X).ParetoFront = 0 Then AllChr(X).ParetoFront = ParetoFrontIndex
      Next X
      For X = 1 To population
         If AllChr(X).ParetoFront < ParetoFrontIndex Then GoTo 110
         For Y = 1 To population
            Count1 = 0: Count2 = 0
            If X = Y Then GoTo 100
            If AllChr(Y).ParetoFront < ParetoFrontIndex Then GoTo 100

               If AllChr(X).TotalCost >= AllChr(Y).TotalCost Then Count1 = Count1 + 1
               If AllChr(X).TotalDelays >= AllChr(Y).TotalDelays Then Count1 = Count1
+ 1
               If AllChr(X).TotalDuration >= AllChr(Y).TotalDuration Then Count1 =
Count1 + 1
               If AllChr(X).TotalIntrruptions >= AllChr(Y).TotalIntrruptions Then Count1
= Count1 + 1

               If AllChr(X).TotalCost > AllChr(Y).TotalCost Then Count2 = Count2 + 1
               If AllChr(X).TotalDelays > AllChr(Y).TotalDelays Then Count2 = Count2 +
1
               If AllChr(X).TotalDuration > AllChr(Y).TotalDuration Then Count2 =
Count2 + 1
            If AllChr(X).TotalIntrruptions > AllChr(Y).TotalIntrruptions Then Count2 =
Count2 + 1

            If Count2 = 4 Then AllChr(X).ParetoFront = 0: Exit For
            If Count1 = 0 And Count2 = 0 Then AllChr(Y).ParetoFront = 0
            If Count1 = 4 And Count2 > 0 Then AllChr(X).ParetoFront = 0: Exit For
100      Next Y
110   Next X
      For X = 1 To population
         If AllChr(X).ParetoFront = ParetoFrontIndex Then Count3 = Count3 + 1
      Next X
      'NoSolutionS(ParetoFrontIndex) = Count3 - Count4              'Counts number of
solutions in each Pareto level
      Count4 = Count3
      If Count3 = population Then AllSolutions = True
   Loop While AllSolutions = False
   For X = 1 To population
      AllChr(X).InvParetoFront = (1 / AllChr(X).ParetoFront)
   Next X
End Sub
```

```
Sub selection_prob()
   totmerit = 0
   summerit = 0                          'Calculate probability of selection
   For X = 1 To population                          'Calculate chromosomes relative
fitness
      totmerit = totmerit + AllChr(X).InvParetoFront
   Next X
   For X = 1 To population
      AllChr(X).Rel_ParetoFront = AllChr(X).InvParetoFront / totmerit
   Next X
   For X = 1 To population                          'Calculate probability of selection
      AllChr(X).SumParetoFront = summerit + AllChr(X).Rel_ParetoFront
      summerit = AllChr(X).SumParetoFront
   Next X
End Sub
```

```
Sub chro_select(Parent1, Parent2)
Randomize
'Parent1 = 0
'Parent2 = 0
'Y = 0
   Do                                    'Select two chromosomes randomly
      A = Rnd
      For X = 1 To population                       'Select first chromosome
         If A < AllChr(X).SumParetoFront Then
            Parent1 = X
            Exit For
         End If
      Next X

      B = Rnd
      For X = 1 To population                       'Select second chromosome
         If B < AllChr(X).SumParetoFront Then
            Parent2 = X
            Exit For
         End If
      Next X

       Identical = True                     'Check if chromosomes are identical

      For Y = 1 To Act * Unit
         If Parent1 = 0 Then Exit For
         If Parent2 = 0 Then Exit For

         If AllChr(Parent1).Genes(Y).GeneValue <>
AllChr(Parent2).Genes(Y).GeneValue Then Identical = False: Exit For
      Next Y

   Loop While Identical = True

    End Sub
```

```
Sub Mutation(mutationindex, NChild)
Randomize                                   'Perform mutation
GCrewCost = 0
   For X = 1 To Act * Unit
      MO = Rnd
      If MO < mutationindex Then
      GN = Int(((AllTasks(X).CrewNumber) - 1 + 1) * Rnd + 1)

      AllChild(NChild).Genes(X).GeneValue = GN
      AllChild(NChild).Genes(X).CrewCost = AllTasks(X).CrewData(GN).Cost
      AllChild(NChild).Note = "Mutation"
      End If
  Next X

   For X = 1 To Act * Unit
 GCrewCost = GCrewCost + AllChild(NChild).Genes(X).CrewCost
   Next X
AllChild(NChild).TotalCost = GCrewCost
End Sub
```

```
Public Sub get_pareto_optimal_final(population, ParetoCount)
   ParetoFrontIndex = 1
   For X = 1 To ParetoCount
    Paretoset(X).ParetoFront = 1
   Next X
 i = 1
 Do
   For X = 1 To ParetoCount
      If Paretoset(X).ParetoFront < ParetoFrontIndex Then GoTo 110
      For Y = 1 To ParetoCount
         Count1 = 0: Count2 = 0
         If X = Y Then GoTo 100
         If Paretoset(Y).ParetoFront < ParetoFrontIndex Then GoTo 100
            If Paretoset(X).TotalCost >= Paretoset(Y).TotalCost Then Count1 = Count1 +
1
            If Paretoset(X).TotalDelays >= Paretoset(Y).TotalDelays Then Count1 =
Count1 + 1
            If Paretoset(X).TotalDuration >= Paretoset(Y).TotalDuration Then Count1 =
Count1 + 1
            If Paretoset(X).TotalIntrruptions >= Paretoset(Y).TotalIntrruptions Then
Count1 = Count1 + 1

            If Paretoset(X).TotalCost > Paretoset(Y).TotalCost Then Count2 = Count2 + 1
            If Paretoset(X).TotalDelays > Paretoset(Y).TotalDelays Then Count2 = Count2
+ 1
            If Paretoset(X).TotalDuration > Paretoset(Y).TotalDuration Then Count2 =
Count2 + 1
            If Paretoset(X).TotalIntrruptions > Paretoset(Y).TotalIntrruptions Then Count2
= Count2 + 1

         If Count2 = 4 Then Paretoset(X).ParetoFront = 0: Exit For
         If Count1 = 0 And Count2 = 0 Then Paretoset(Y).ParetoFront = 0
         If Count1 = 4 And Count2 > 0 Then Paretoset(X).ParetoFront = 0: Exit For
100     Next Y
```

```
110 Next X
i = i + 1
ParetoFrontIndex = ParetoFrontIndex + 1
Loop Until i > ParetoCount
   NoParetoSol = 0
   For X = 1 To ParetoCount
      If Paretoset(X).ParetoFront = 1 Then NoParetoSol = NoParetoSol + 1
   Next X
End Sub
```

```
Sub selec_prob_pareto(population, ParetoCount)
totmerit = 0
   summerit = 0                          'Calculate probability of selection

   For X = 1 To ParetoCount
      If Paretoset(X).ParetoFront < 1 Then GoTo 200
      Paretoset(X).InvParetoFront = (1 / Paretoset(X).ParetoFront)
200   Next X

   For X = 1 To ParetoCount                                  'Calculate chromosomes
relative fitness
      totmerit = totmerit + Paretoset(X).InvParetoFront
   Next X
   For X = 1 To ParetoCount
      Paretoset(X).Rel_ParetoFront = Paretoset(X).InvParetoFront / totmerit
   Next X
   For X = 1 To ParetoCount                                  'Calculate probability of
selection
      Paretoset(X).SumParetoFront = summerit + Paretoset(X).Rel_ParetoFront
      summerit = Paretoset(X).SumParetoFront
   Next X
End Sub
```

```
Public Sub Get_Distance(UnitDistance)
UserForm2.Caption = "Distance between units"
UserForm2.Label1.Caption = "Type distance between units"
UserForm2.Label2.Caption = "Unit: " & Y
UserForm2.Label3.Caption = "and Unit: " & X
UserForm2.Show
UnitDistance(Y, X) = UserForm2.TextBox1.Value
End Sub
```

```
Public Sub Get_Speed(CrewSpeed)
UserForm2.Caption = "Crews' Speed"
UserForm2.Label1.Caption = "Type the Crews' speed of Activity: " & Y
UserForm2.Label2.Caption = "Crew number: " & X
UserForm2.Label3.Caption = "NOTE: Crew Speed > Zero"
UserForm2.Show
CrewSpeed(Y, X) = UserForm2.TextBox1.Value
End Sub
```

```
Public Sub Get_TransCost(CrewTransCost)
UserForm2.Caption = "Crews' Transportation Cost"
UserForm2.Label1.Caption = "Type the Crew Transporation cost perday for Activity: " &
Y
UserForm2.Label2.Caption = "Crew number: " & X
UserForm2.Label3.Caption = ""
UserForm2.Show
CrewTransCost(Y, X) = UserForm2.TextBox1.Value
End Sub
```