REPETITIVE PROJECT SCHEDULING USING GENETIC ALGORITHMS

TAN HENG WEE

NATIONAL UNIVERSITY OF SINGAPORE

2004

REPETITIVE PROJECT SCHEDULING USING GENETIC ALGORITHMS

TAN HENG WEE
(B.Eng. (Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF CIVIL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

ACKNOWLEDGEMENTS

To complete an academic thesis is always an onerous and sometimes consuming task. It is the following people whom I constantly draw strength from to accomplish this seemingly insurmountable task at times.

I have learnt more than just scheduling and optimization from the three fulfilling years working under Associate Professor Chan Weng Tat. I am most appreciative of his *laissez-faire* philosophy, which allowed me to pursue various non-related interests, resulting in perhaps the most fruitful learning experience I have had thus far. There are so much more that I can learn from him (I know I am still struggling to emulate, amongst others, his clarity in expressions); alas the world is, lamentably, no utopia, and pragmatism invariably reigns.

Two wonderful years were spent with a bunch of great colleagues, with whom ideas were discussed and debated; dreams shared and cherished. How can one forget those delightful drinking sessions, where politics were deliberated at great length? It is indeed no exaggeration to say that I have learnt much from the perceptive insights of these colleagues. Those were two very good years indeed.

Finally, one girl has never failed to amaze me with her magnanimity towards my numerous faults and foibles. To Ms Jasmin Tay, thank you for all the support that you have shown and for putting up with my constantly busy schedule. It has been eleven unbelievable years with you, and counting!

TABLE OF CONTENTS

SUMMARY

Repetitive projects consist of multiple units that are identical or similar in nature, with various activities or trades repeatedly executed from one physical unit to another during the construction process. This repetitive construction process renders it desirable to schedule the execution of these trades as uninterrupted and continuous work sequences by imposing a work continuity constraint whenever necessary. The traditional Critical Path Method (CPM) is cumbersome when used on such projects due to their repetitive nature and the large number of units present. In addition, the CPM equations do not take work continuity into consideration when calculating activity start/finish times. Existing linear scheduling models geared towards work continuity are nevertheless graphical in nature and therefore not easily amenable to computerization. This study seeks to address the drawbacks of these methods.

A new scheduling approach for repetitive projects is proposed. The CPM equations are adapted as a new set of recurrence equations which can ensure work continuity through imposing a scheduling constraint that adjusts the calculations of activity start/finish times. This new representation has two main advantages. Firstly, the mathematical representation lends itself to computerization and preserves CPM's analytical capabilities which are absent in the graphical scheduling approach. Secondly, the equations enable user-specific scheduling considerations encountered in practice, like mandatory delays imposed between units, to be easily incorporated. A case study is used to illustrate the utility of the proposed scheduling approach when work continuity and other scheduling considerations are imposed.

This set of equations also forms the basis for optimizing the schedules of repetitive projects with an evolutionary optimization technique. The Genetic Algorithm is used to search for the best schedule by varying the crew size and work continuity requirements of the project activities using a suitable chromosome representation. The schedules which have lower work continuity and do not meet deadlines are penalized.

A case study benchmarks the performance of the Genetic Algorithms Recurrent-equations Approach (GARA) against the CPM and linear scheduling methods in optimizing the schedules. Results indicate that GARA consistently produces superior schedules.

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

A_i	Task *i*
A_iU_j	Activity *ij*
C_i	Daily basic wages for worker of task *i*
C_T	Daily tardiness penalty
D_i	Effective duration for task *i*
EF_{ij}	Earliest finish date for activity *ij*
K_i	Crew size for task *i*
LF_{ij}	Latest finish date for activity *ij*
M	Total number of tasks in a project
Q	Total number of repetitive units
S	Contractual duration for project completion
T_i	Duration for task *i*
U_j	Unit *j*
$\omega_{i,j}$	Number of days delayed on activity *ij*'s earliest finish date
$\Pi_{i,j}$	Number of days delayed on activity *ij*'s earliest and latest finish dates

CHAPTER 1

INTRODUCTION

Repetitive projects are characterized by the existence of several identical or similar units, where construction activities are sequentially executed from one unit to another in a vertical or horizontal manner, resulting in a linear workflow. As such, repetitive projects are also known as linear projects in some of the literature. Examples of horizontal repetitive projects include highway or pipeline construction, where the units can be stations on the highways or meters of pipelines respectively. Multi-story buildings, where different stories constitute the set of repetitive units, are representative of vertical repetitive projects.

From the perspective of the individual trades on the project, a construction schedule that ensures the uninterrupted flow of resources from one unit to the next is preferred. When the work crew is the critical resource, this uninterrupted flow of resources leads to the concept of work continuity. In order to maintain work continuity, repetitive units must be scheduled to enable the timely movement of crews from one unit to the next. The benefits of this arrangement include the maximization of the learning curve effect for each crew, and the minimization of idle time for each crew. Furthermore, it ensures that the specialist contractors can work straight through a project and leave - a working condition reported to be ideal for them.

Traditionally, network techniques such as the Critical Path Method (CPM) have been used extensively in the construction industry for scheduling and controlling construction projects. However, traditional CPM proves to be cumbersome to apply

for the scheduling of repetitive projects, and critical information is obscured among the details resulting from the way activities are represented. CPM also fails to address the issue of how to ensure work continuity in a particular trade. Several graphical scheduling techniques have been developed to address the weaknesses of the CPM method with respect to repetitive projects. However, as these techniques are graphical in nature, they are not easily amendable to computerisation and thus lack analytical capabilities.

This study proposes a new approach to address the issues identified in scheduling repetitive projects by adapting the CPM equations, to form a new set of recurrence equations, which consider work continuity. An additional work continuity term is added to the CPM equations such that suitable adjustments to the calculations of activity start/finish times can be effected. It is also possible to incorporate specific user-defined scheduling constraints like mandatory time lags between units using the same technique. This proposed approach retains the advantages of using a mathematical representation of the problem, namely ease of computerization and analytical capability. Furthermore, this set of recurrence-equations forms the basis for performing schedule optimization using an evolutionary search algorithm.

This chapter first reviews the characteristics of repetitive projects, and provides an overview of the limitations of current scheduling methods with respect to repetitive projects. This leads to the identification of the objectives and scope of this study, followed by the organization of the thesis itself.

1.1 Characteristics of Repetitive Projects

The essential elements in construction projects are the tasks, their durations, and the *logical* interrelationships among them. A task is an operation or closely related group of operations whose performance contributes to the completion of the overall project. Precedence constraints link tasks together to reflect the intended or natural sequence of the operations. The start of a particular task is permitted by the completion of all preceding tasks or by the start of the project.

Fig. 1.1 depicts part of a typical CPM network for a multi-story building construction project. Each story of the building is identified as a 'unit' in the figure. The square nodes depict the project activities, whereas the directed arrows represent the precedence constraints between the activities. The figure also makes use of two kinds of arrows – solid ones to represent precedence between activities performed on the same story / unit, and dashed ones to represent precedence between activities of the same trade performed on different units.

At this point, it is useful to make a distinction between the 'task' and 'activity' which are often used synonymously. For repetitive projects, we propose to use 'task' to refer to the activity carried out by particular trades / specialists irrespective of the unit on which the task is performed. The word 'activity' retains its traditional usage as referring to the individual elements that make up a project network; in repetitive projects, an activity would mean the performance of a task on a particular unit. For example, task D is defined as the task of erecting the precast walls. Irrespective of which unit the activity is performed on, the precedence constraint ensures that this task will only commence upon the completion of the preceding task – setting of the

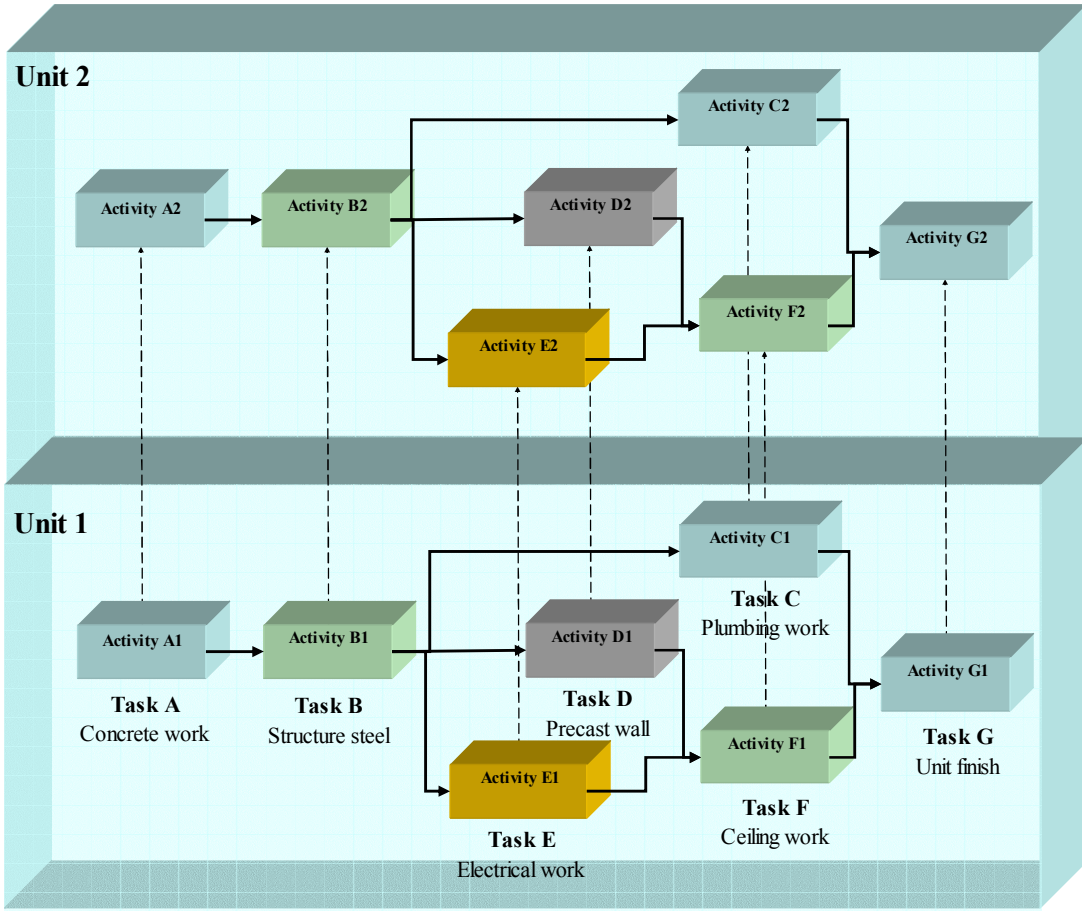


Figure 2.1 Network representation of a repetitive project

necessary structural steels in place. The solid arrows linking the different tasks represent all the activity-to-activity precedence constraints between tasks performed on the same unit. For instance, tasks C, D and E cannot commence until the completion of task A.

Given the information on these elements, the activity start / finish times and total float can be computed. These computations also yield the total expected duration for the project, and identify the most critical activities and hence the critical path for the

project. This is a powerful concept that great aids management in setting its priorities for allocating resources to operations.

Fig. 1.1 also illustrates a characteristic peculiar to repetitive projects. A repetitive project consists of multiple instances of identical CPM projects, each of which represents tasks for a repetitive unit. The first characteristic is the large number of activities required to represent the repetitive project as the number of units increase. This creates a problem in using networks to represent repetitive projects as Fig. 1.1 illustrates where a crowded network results from a project that consists of only seven tasks over two repetitive units. It is therefore necessary to be able to refer to activities grouped by unit or task.

The second characteristic pertains to the repetitive execution of a task between successive units. When there is only one work crew available for each task, the work activity on one unit cannot commence until the work crew is available again, usually through the completion of the preceding unit. The dashed arrows linking the same task from one unit to the next represent the resource availability constraint. The general contractor recognizes the desire of the different specialist contractors / trades for a continuous and uninterrupted movement of work crews from one unit to another without unnecessary crew idle time. Such continuity provides for an efficient resource utilization strategy that leads to the maximized learning curve benefits, minimized crew idle time and reduced the off-on movement of crews on a project once work has begun (Ashley 1980). The traditional CPM scheduling algorithm cannot ensure such work continuity.

1.2 Drawbacks of Existing Scheduling Methods

Both the CPM and the Graphical Scheduling Models (GSM) can be used to schedule repetitive projects. They have their own strengths and weaknesses, but neither is adequate on its own to address the scheduling needs of repetitive projects.

Although CPM provides a well-established logic in analyzing and scheduling networks, the two characteristics discussed renders the utilization of CPM in repetitive projects unsuitable. The network diagram in CPM cannot effectively communicate vital schedule information to the end-users as the complex network needed to represent repetitive projects obscures this information. In comparison, the line representation depicted in Fig. 1.2 provides a more effective representation of the progression of work and work sequences in a repetitive project; the breaks in the lines intuitively convey the presence of work discontinuities.

In addition, CPM calculation assumes that activities commence as soon as all the precedence constraints are satisfied. On repetitive projects, this causes tasks progressing at a faster rate to be "held back" by preceding tasks that are progressing at a slower pace. This results in a time lapse in a task's finish-to-start dates between two consecutive activities, creating undesirable work discontinuities. For example, in Fig. 1.2 the time to complete one unit for tasks A and B are 10 and 5 days respectively. Under CPM computation, the longer amounts of time taken to complete a unit by task A delays the start time for all the activities in task B by 5 days.

The limitations of traditional CPM led to the development of several graphical approaches such as the Line-of-Balance (LoB), and recent methods like Linear

Scheduling Model (Harmelink and Rowing, 1998) and Repetitive Scheduling Model (Harris and Ioannou, 1998). In these graphical methods, the tasks are plotted as lines over the axes of units versus time. In this case, the slope represents the work rate for the task. The advantage of this form of representation is its simplicity and the ease with which to visualize the whole project schedule. Fig 1.2 illustrates the line representation for the network schedule of Fig 1.1 but with five repetitive units.

These models also account directly for work continuity to ensure effective resource utilization. While there are minor variations in the method of ensuring work continuity, it generally involves introducing a delay in the first activity of particular tasks. However, the graphical models are not easily amenable to computerization and lack CPM's analytical capabilities. In addition, work continuity is uniformly imposed on all the activities; this reduces flexibility in determining good schedules where a mix of work continuous and discontinuous activities may lead to lower project costs.

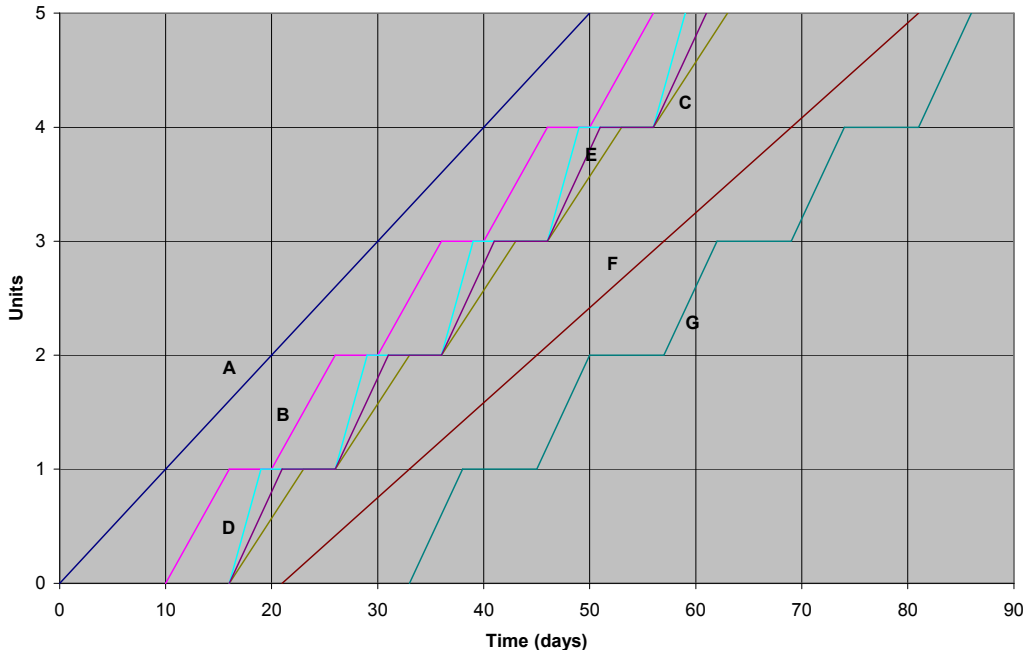


Figure 1.2 Line representation of a network schedule over five repetitive units

In summary, neither of the two methods mentioned is adequate on its own for scheduling repetitive projects. It is possible that a customized scheduling technique for repetitive projects can be developed by combining the strengths of the individual methods. An improved repetitive scheduling method will be of great benefit as repetitive projects are very common in civil engineering works.

1.3 Objectives and Scope of the Thesis

This study has two objectives, namely to:

(1) Develop a new scheduling technique for repetitive projects by combining the strengths of the CPM and GSM. The new scheduling technique retains the ability to numerically compute schedule attributes like early / late starts and finishes of individual activities, presents the schedule information in a manner that is easy to comprehend, and takes discretionary work continuity for tasks into consideration. Recurrence equations that link the schedule attributes form the basis of the proposed scheduling method.

(2) Extend the functionality of these recurrence equations by using them as a means of effecting schedule optimization using Genetic Algorithms (GA). This GA Recurrence-equations Approach (GARA) searches for the best schedule in terms of crew size and work continuity status using a suitable chromosome representation. The objective function is formulated in such a way that schedules with work discontinuity and which do not meet completion deadlines are penalized.

The scope of work for the research includes:

(1) Developing a set of recurrence equations suitable for performing network scheduling calculations for repetitive projects; the recurrence equations incorporate the work continuity constraint when calculating activity start / finish dates;

(2) Developing means of incorporating specific user scheduling requirements into the scheduling calculations;

(3) Developing a schedule optimization model using the proposed recurrence equations with GA;

(4) Evaluating the effectiveness of the proposed optimization model by performing a series of comparisons with CPM and GSM

1.4 Organization of the Thesis

This thesis is organised into six chapters, beginning with this chapter. Chapter 2 reviews the existing methods and identifies the inadequacies of these methods with respect to scheduling repetitive projects. This chapter also reviews the use of genetic algorithms in scheduling problems.

Chapter 3 describes the development of a set of recurrence equations used in the proposed repetitive scheduling method. This chapter also describes how user-specified scheduling constraints can be incorporated into these equations. A case study illustrates the application of the proposed repetitive scheduling method when work continuity constraints and specific scheduling requirements are imposed.

Chapter 4 introduces the model setup for schedule optimization using the genetic algorithms. The two types of chromosome representations used and the formulation of

a suitable objective function that reflects actual conflict of interests in practice are discussed.

Chapter 5 describes the experimental setup and presents the analysis of results with respect to the effects of increasing repetitive units and different due date constraints. The times of convergence for networks with various complexities are also compared.

Finally, chapter 6 summarizes the significant findings and observations in this thesis before concluding with the limitations and suggestions for future study.

CHAPTER 2
LITERATURE REVIEW

This chapter reviews the existing literature that is relevant to scheduling repetitive projects. The chapter begins with a summary of important terms used by authors in the literature. This is followed by a review of the strengths and weaknesses of various scheduling methods when applied on repetitive projects. The chapter closes with a review of the use of Genetic Algorithms (GA) in construction scheduling problems.

2.1 Review of Terms Used in Existing Scheduling Methods

Various researchers in the literature on repetitive project scheduling have defined some commonly occurring terms differently. The purpose of this section is to review the usage of some of these terms and propose a definition to be adopted in this thesis in order to avoid ambiguity.

Generally, repetitive projects are made up of a series of construction tasks that are repeatedly performed from one unit to another (Kang *et al.*, 2001). Examples of repetitive projects include high-rise buildings, housing project with identical model houses, pipeline network constructions, or highway projects (Moselhi and El-Rayes, 1993). Although Harris and Ioannou (1998) use "activity" to refer to the same concept, we prefer to keep "activity" for instances of the same construction task when performed on individual units.

Repetitive projects are also known as "linear projects" to convey the idea of a linear progression of construction activities from the first to the last unit (Selinger, 1980).

However, Moselhi and Hassanein (2003) categorized different repetitive projects into either linear or non-linear projects according to the nature of the units on which the construction tasks are executed. Where tasks are repeatedly executed over *non-identical* sections, such as highway, railroad and pipeline constructions, they are known as linear projects. On the other hand, high-rise building constructions are termed as non-linear projects because the tasks are repeated over *identical* units. This study adopts the general term "repetitive projects" because the proposed scheduling method is able to account for tasks that are executed over non-identical sections by varying the durations for each section.

It is widely recognized that it is necessary to maintain continuous and uninterrupted movements of work crews in repetitive projects due to the nature in which the tasks in repetitive projects are carried out (Reda, 1990; Harris and Ioannou, 1998). However, Reda (1990) called this desired workflow "work continuity" whilst Harris and Ioannou (1998) used to term "resource continuity" to describe the same concept. In this study, resource continuity is construed as a broader term that encompasses the continuous utilization of all the resources, including the work crew, specialized equipment and perhaps, even the working space required. When there is only a single instance of a resource available to perform a particular task (and thus becomes the limiting resource for each task), resource continuity becomes work continuity for the resource involved. This study adopts work continuity as being operative for the scheduling model that is the subject of this study.

Finally, Harris and Ioannou (1998) defined a critical activity in a CPM network as "one that, if delayed, will delay the project". In addition, they also defined the critical

path as “a chain of critical activities extending from project start to project finish”. It is implied that a critical activity is one with no total float. However, Barrie and Paulson (1992) offer a more general definition of a critical path to mean “a continuous chain of activities from the beginning to the end of the network with the minimum total float value”. Whilst seemingly trivial, the distinction is important for resource critical repetitive projects where activities may be delayed because of the non-availability of a critical resource, and there is no time-continuous chain of activities from the beginning to the end of the network. This thesis adopts the more general definition of a critical path by Barrie and Paulson (1992).

2.2 Existing Scheduling Methods for Repetitive Projects

2.2.1 Critical path method

The critical path method (CPM) is a network-based scheduling technique that evolved from a research effort initiated in late 1956 by the Engineering Services Department of Du Pont Company. Their objective was to explore the use of computer-aided systems in planning, scheduling, monitoring, and controlling Du Pont’s engineering projects.

Mattila and Abraham (1998) stated that the process of planning, scheduling and control in construction projects is typically accomplished using CPM. This is no surprise as the advantages of CPM networks are well documented. Barrie and Paulson (1992) discussed three advantages of CPM networks in comparison with bar charts. Firstly, the logical interrelationships and dependencies among activities are inherent in the networks, but cannot be readily shown on bar charts. This makes networks much more useful for forecasting and control as the impact on the whole project of a

delay in an activity is readily transmitted by network logic and computations through the whole schedule. Secondly, networks also provide a more powerful means for documenting and communicating project plans, schedules and performance with its standardized notation and diagramming. Finally, networks, in contrast to other techniques, identify the most critical elements in the project schedule by identifying the activity floats and recognizing the critical path, thus providing management with the necessary information to set priorities for action.

While CPM is generally held to be powerful analytical tool for scheduling construction projects, its limitations when applied to repetitive projects are well-documented (Reda, 1990; Suhail and Neale, 1994). The first drawback is that networks tend to becomes "crowded" due to the numerous repetitive units present in repetitive projects. A study by Yamin and Harmelink (2001) found that project managers place a very high priority on the effective communication of scheduling information. Since the network gets increasingly complex as the number of repetitive units increase, the usefulness of CPM as a communication tool diminishes with increasing repetitive units. This reduces the effectiveness of both the graphical and textual means of disseminating vital scheduling information among the parties involved.

The inability to guarantee work continuity for the construction tasks is the second drawback of CPM techniques. In a survey conducted by O'Brien and Fischer (2000), it is reported that most subcontractors maintain a core group of workers they take care not to lay off unnecessarily. This means that the subcontractors incur additional costs in terms of lost wages paid to the workers during the idle periods when they are

deployed on projects where workflow is discontinuous. Therefore, all the subcontractors in the survey reported that the ideal conditions are when their workers can "get on a job, work straight through, and leave". CPM makes no special provision to ensure work continuity in tasks on repetitive projects since it does not distinguish between tasks and activities.

2.2.2 Graphical linear scheduling methods

Various linear scheduling techniques, including the Line-of-Balance (Lumsden 1968), Vertical Production Method (O'Brien, 1975) and Linear Scheduling Method (Johnston, 1981) were proposed to address the limitations of CPM techniques when applied to repetitive project scheduling. The advantages of these graphical approaches lie in the ease of visually comprehending the whole project and the ability to maintain work continuity for tasks.

However, Neale and Neale (1989) found that the Line-of-Balance (LoB) method can only show schedules with limited complexity and information, and beyond which the schedules degenerate into diagrams with incomprehensible "masses of flow lines". Furthermore, these graphical approaches are not easily amenable to computerization (Chrzanowski and Johnston, 1986) and they lack the analytical capabilities that underpin the popularity of the CPM techniques.

In response to the lack of analytical capabilities for the graphical approaches, Harmelink and Rowings (1998) introduced the Linear Schedule Model (LSM). The LSM produced a Controlling Activity Path (CAP) comprising of a sequence of activities that must be completed as planned to finish the project within the overall

planned duration. Similarly, Harris and Ioannou (1998) introduced the Repetitive Scheduling Method (RSM) that derives a Controlling Sequence (CS) containing both critical and non-critical activities but excludes resource critical activities.

While Mattila and Park (2003) reported that these two methods identified the same path for a simple configuration of two activities, subsequent discussion (Kallantzis and Lambropoulos, 2004) showed that the two methods do not always identify the same controlling path in more complex activity configurations, nor are they synonymous with the critical path identified in CPM techniques. Notwithstanding the two recent methods of CAP and CS, the critical path as defined by Barrie and Paulson (1992) is derived to demonstrate the analytical capability of the recurrence equations in this study.

2.3 Genetic Algorithms and Its Applications in Scheduling

2.3.1 Overview of genetic algorithms

The Genetic Algorithms (GA) proposed by Holland (1975) are stochastic search methods that have been successfully applied in many types of problems, including process scheduling and resource allocation (Gen and Cheng 1997). GA differs from conventional search techniques and starts with an initial set of random solutions called a population. A single string called a chromosome, which consists of a linear string of genes, represents each individual solution. When applied to scheduling, each individual chromosome in the population corresponds to one possible schedule through decoding the genes.

The chromosomes evolve through successive iterations known as generations. The crossover and mutation operators are applied to selected chromosomes to create the next generation. During each generation, the chromosomes are evaluated using some measure of fitness defined by the end-users. Fitter chromosomes generally have higher probabilities of being selected to form the next generation. After several generations, the algorithm should converge to the best chromosome, which hopefully represents the optimum or near-optimal solution to the problem. The cycle of evolution is repeated until a desired termination criterion specified by the end-user is reached. This criterion can be the number of evolution cycles, the amount of variation of individuals between different generations, or a pre-defined value of fitness.

2.3.2 Applications of genetic algorithms in construction scheduling

Scheduling involves the allocation of resources over a period to perform a collection of tasks subject to known constraints, and is a difficult task for human planners especially when optimal solutions are required (Chan and Hu, 2002).

Chan *et al.* (1996) proposed a GA approach to schedule construction resources. The strength of this approach lies in the selection and recombination of the GA to learn the domain of the specific network instead of relying on any set of heuristic rules. Hegazy (1999) extends the research in this area by proposing the application of GA with improved resource allocation and levelling heuristics to search for near-optimal solutions.

Li and Love (1997) presented an "improved GA system" to reduce the computational time and increase the reliability of results obtained for the time-cost optimization

problems in construction projects. Similarly, Feng *et al*. (1997) proposed an algorithm based on the principles of GA to solve time-cost trade-off problems for large-scale CPM networks. Li *et al*. (1999) integrated a machine-learning method with GA to solve nonlinear time-cost trade-off scheduling problems. In addition, Hegazy and Wassef (2001) proposed a GA-based time-cost trade-off optimization model for projects with non-serial repetitive activities.

Leu and Yang (1999) proposed a multi-criteria computational optimal scheduling model that integrates the time-cost trade-off model, resource-limited model and resource leveling model using GA. Recently, Zheng *et al*. (2004) considered a GA-based multi-objective time-cost optimization model that is applicable to the alternative project delivery system, where the benefits and opportunities of seeking an earlier project completion are taken into account.

2.4 Concluding Remarks

Two issues were identified from the literature review.

Firstly, neither CPM nor the various graphical scheduling methods are adequate, on their own, in addressing the scheduling needs of repetitive projects. In scheduling repetitive projects, it is necessary to pay particular attention to the repetitive nature of the tasks and the need for an uninterrupted movement of crew from one unit to the next.

Secondly, repetitive scheduling presents the opportunity of optimization to determine schedules that are attractive from the perspective of overall project duration as well as

resource usage. The GA has proven to be a robust method of doing schedule optimization. These two observations form the motivation for the development of the repetitive scheduling model described in the next chapter.

CHAPTER 3

RECURRENCE-BASED SCHEDULING OF REPETITIVE PROJECTS

This chapter describes the development of a set of recurrence equations suitable for performing network-scheduling calculations for repetitive projects. Two specific scheduling requirements with relevance to industry are identified and the means of incorporating them into the recurrence equations are described. Finally, a case study illustrates the application of the proposed repetitive scheduling method when work continuity constraints and specific scheduling requirements are imposed. For ease of reference, the parameters appearing in the set of recurrence equations and discussed in this chapter are summarised in Table 3.1

3.1 Development of the Recurrence Equations

Consider a repetitive project comprising a set of tasks A_i = (1, 2, ..., M) to be executed over a number of repetitive units U_j = (1, 2, ..., Q) under a set of finish-to-start

Table 3.1 Parameters for mathematical representation

Parameters	Descriptions
A_i	Task i
U_j	Unit j
A_iU_j	Activity ij
T_i	Duration for task i
D_i	Effective duration for task i, defined by Eqn. 3.5
$\omega_{i,j}$	Number of days delayed on activity ij's earliest finish date
$\Pi_{i,j}$	Number of days delayed on activity ij's earliest and latest finish dates
Q	Total number of repetitive units
M	Total number of tasks in a project
EF_{ij}	Earliest finish date for activity ij
LF_{ij}	Latest finish date for activity ij
S	Contractual duration for project completion
C_T	Daily tardiness penalty
C_i	Daily basic wages for worker of task i
K_i	Crew size for task i

precedence relationships. The duration of a task, T_i, denotes the amount of time required to complete the task A_i over one of the units, or simply the time taken to complete an activity (A_iU_j) of the task. The predecessors and successors of task A_i are denoted by the sets A_p and A_s respectively. The earliest finish date (EF_{ij}) and latest finish date (LF_{ij}) for each activity A_iU_j can be calculated with the following set of equations which are also used in CPM:

Initial Conditions:

$$EF_{1,1} = T_1 \tag{3.1}$$

$$EF_{Q,M} = LF_{Q,M} \tag{3.2}$$

Recurrence relations:

Forward Pass:

$$EF_{i,j} = \underset{\forall p \prec i}{MAX}[EF_{p,j}, EF_{i,j-1}] + T_i \tag{3.3}$$

Backward Pass:

$$LF_{i,j} = \underset{\forall s \succ i}{MIN}[(LF_{i,j+1} - T_i), (LF_{s,j} - T_s)] \tag{3.4}$$

3.1.1 Effective duration

In considering work continuity in tasks, it is useful to use the concept of effective duration. The effective duration is the *minimum* task duration for which work continuity is maintained. Mathematically, the effective duration can be calculated by:

$$D_i = \{\underset{\forall p \prec i}{MAX}[EF_{p,Q}] - \underset{\forall p \prec i}{MAX}[EF_{p,1}]\}/(Q-1) \tag{3.5}$$

The concept of effective duration is introduced for two reasons. Firstly, the effective duration provides a reliable means of determining whether a task already has a continuous workflow without the need to impose any work continuity constraint.

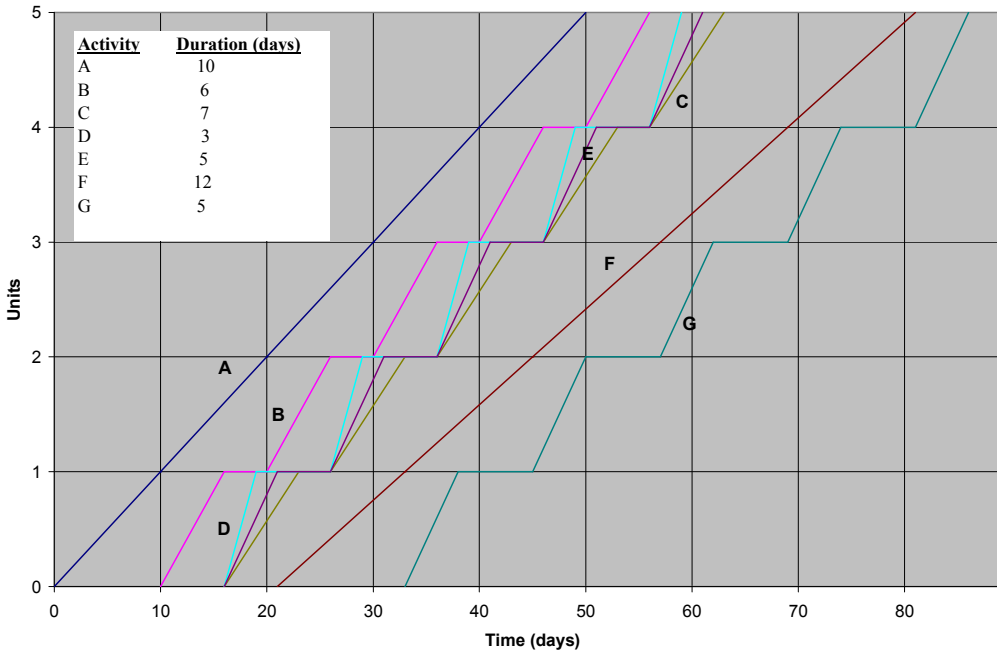


Figure 3.1 Linear schedule calculated using CPM equations

Once these tasks are identified, any available resources can be more efficiently utilized by channelling them to the discontinuous activities. Secondly, the difference between the effective duration (D_i) and duration (T_i) of a task denotes the number of idle days between two activities due to a break in the workflow for the task. The quantification of this idle period is useful in the incorporation of the work continuity term into the recurrence equations.

Fig. 3.1 depicts the linear schedule from the CPM calculations on the network shown in Fig 1.1. Fig 3.1 depicts the work progress for tasks B, C, D, E and G, each of which is a set of 5 discontinuous lines indicating a fragmented workflow for these tasks. For instance, there are four days of work discontinuity between each activity of task B. In contrast, despite having predecessors and successors that are discontinuous in their workflows, task F nevertheless enjoys full work continuity. The concept of effective

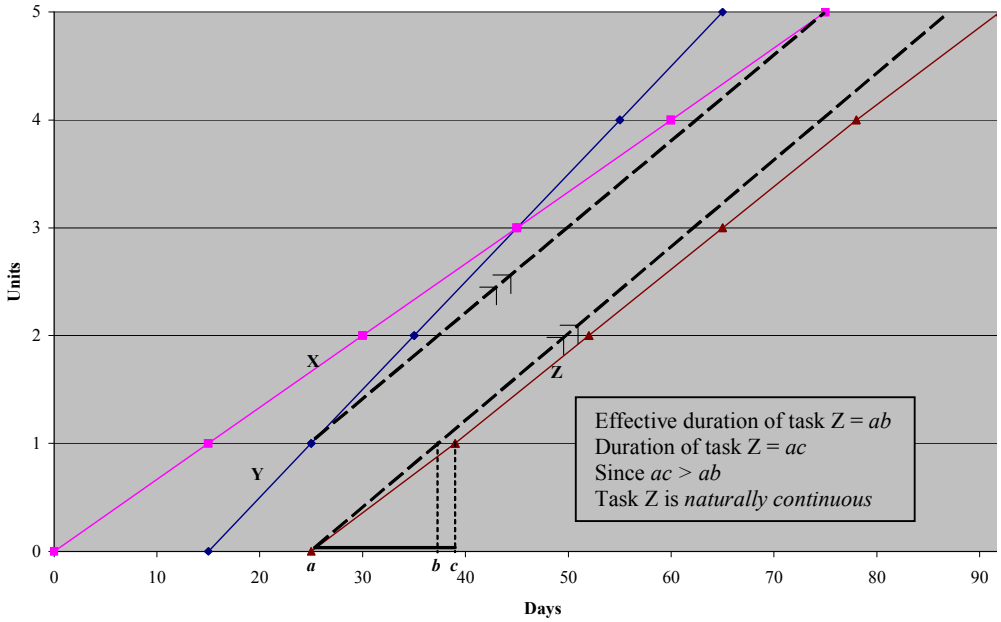


Figure 3.2 Illustration of effective duration

duration can be used to determine whether the workflow for a task is continuous without imposing work continuity on it.

Generally, a task has a fragmented workflow if it is progressing at a faster pace than its preceding tasks as precedence constraints prevent a task from starting until its preceding tasks are completed. For instance, task B is work discontinuous and its activity start dates are "held back" by 4 days because its duration of 6 days is less than the duration of its preceding task A that has a duration of 10 days. In terms of effective duration, the workflow of a task is fragmented when its effective duration (D_i) is larger than its duration (T_i). For example, task F is work continuous because its effective duration (D_F) of 10 days is less than its duration (T_F) of 12 days.

The effective duration concept for determining the work continuity status of a task is especially useful when it has numerous predecessors, and the lines of progress of

these predecessors intersect. Fig. 3.2 illustrates one such scenario, where tasks X and Y are predecessors of task Z, and their durations are 15, 10 and 14 days respectively. Task X commences immediately while task Y only starts on the 15th day. Task Z might be thought to be work discontinuous because it is progressing faster than task X. From Eqn. 3.5, the effective duration of task Z is calculated as:

$$D_Z = \{MAX[EF_{X,5}, EF_{Y,5}] - MAX[EF_{X,1}, EF_{Y,1}]\}/(5-1)$$

$$= \{MAX[75,65] - MAX[15,25]\}/4$$

$$= (75-25)/4$$

$$= 50/4$$

$$= 12.5 \quad days$$

This is less than its duration of 14 days, making task Z naturally work continuous.

3.1.2 Imposing work continuity

Generally, the graphical scheduling methods ensure work continuity for a task by delaying the earliest finish date of its first activity. To account for the effect of imposing work continuity on the start / finish dates of task activities, an additional term is introduced into Eqn. 3.3. By imposing the requirement that the earliest start date of the last activity remain unchanged, the necessary amount of delay for the first activity ($\omega_{i,1}$) so that work continuity can be effected for the entire task can be calculated as the sum of the periods of work discontinuity. As illustrated in Fig. 3.3, the period of work discontinuity between two units is the difference between the effective duration and the duration of a task. For Q repetitive units, there are Q-1 potential breaks in the work continuity of the task, each resulting in an idle period. Therefore, when the work continuity constraint is imposed on the task that has a

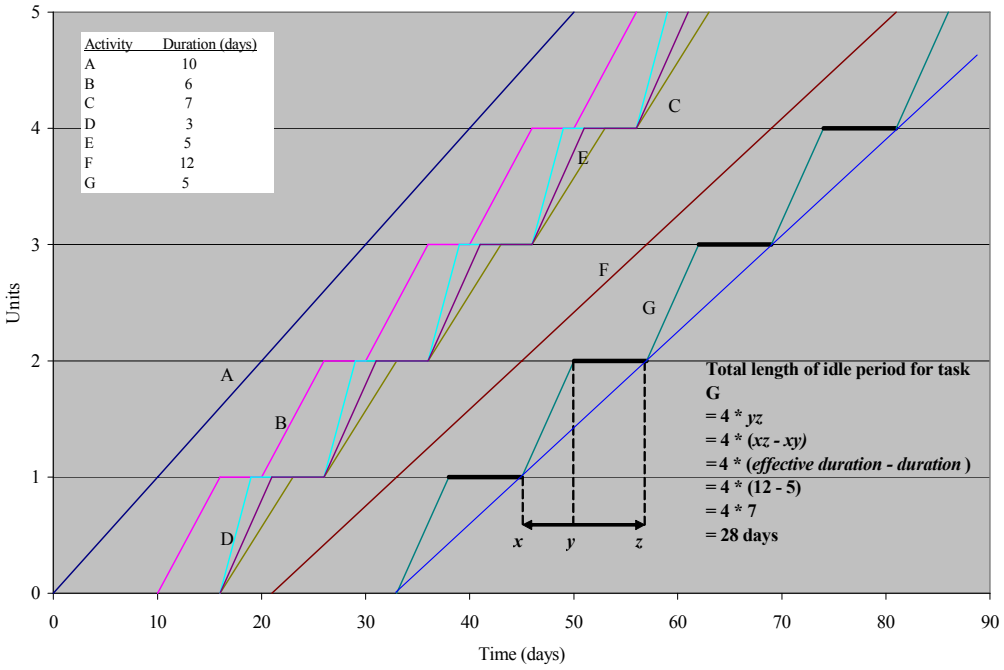


Figure 3.3 Calculation of the number of idle days between activities in a task

fragmented workflow, the necessary delay in the earliest completion of the first activity (to eliminate the breaks) is:

$$\omega_{i,1} = (Q-1)MAX[0,(D_i - T_i)] \tag{3.6}$$

If the work continuity constraint is imposed on task G in Fig. 3.3, the earliest finish date (EF) of its first activity will be delayed by the product of 4 ($Q - 1$) and 7 days ($D_i - T_i$), which is 28 days. These 28 days represent the shortest required length of delay for the EF of task G's first activity of task G in order to ensure work continuity in all subsequent activities. A lesser value will result in discontinuity between later units, while an excessive value will delay the overall project completion unnecessarily. Translating this method into mathematical terms, the *proposed recurrence equations* are:

Initial Conditions:

$$EF_{1,1} = T_1$$

$$EF_{Q,M} = LF_{Q,M}$$

Recurrence Functions:

Forward Pass:

$$D_i = \{\underset{\forall p \prec i}{MAX}[EF_{p,Q}] - \underset{\forall p \prec i}{MAX}[EF_{p,1}]\}/(Q-1)$$

$$EF_{i,j} = \{\underset{\forall p \prec i}{MAX}[EF_{p,j}, EF_{i,j-1}] + T_i\} + \omega_{i,j} \qquad (3.7)$$

Backward Pass:

$$LF_{i,j} = \underset{\forall s \succ i}{MIN}[(LF_{i,j+1} - T_i), (LF_{s,j} - T_s)]$$

where

$$\omega_{i,j} = \alpha\beta\{(Q-1)(MAX[0, (D_i - T_i)])\} \qquad (3.8)$$

$$i, p, s \in (1, 2, ..., M) \; ; \; j \in (1, 2, ..., Q)$$

$$\alpha = \begin{cases} 1 & for \quad j = 1 \\ 0 & otherwise \end{cases} \; ; \; \beta = \begin{cases} 1 & work \quad continuity \quad imposed \\ 0 & otherwise \end{cases}$$

Through the introduction of the work continuity term, the new set of recurrence equations is able to calculate activity start/finish times when the work continuity constraint is imposed on any selected group of activities.

The schedule information calculated using the recurrence equations can be represented in various ways graphically to effectively convey the information to the end-user. Examples include the matrix schedules and the “horse blanket”.

3.2 Incorporating Specific Scheduling Requirements

It is sometimes necessary to accommodate interruptions in the work programme that are known ahead of time. For instance, the precast yard that is responsible for delivering the precast wall elements may have informed the contractor that they are not able to deliver the elements on particular days. The contractor then has to impose a mandatory work break on those days for the task involving the wall element.

Another kind of scheduling consideration arises when the amount of work that needs to be done from one unit differs from the others. For example, the amount of work for a task in units 3 and 4 is twice that for the other units. With a constant crew size, the duration for the task on units 3 and 4 would have to be doubled to account for the increased quantity of work.

A simple example involving these two scheduling considerations will be used to

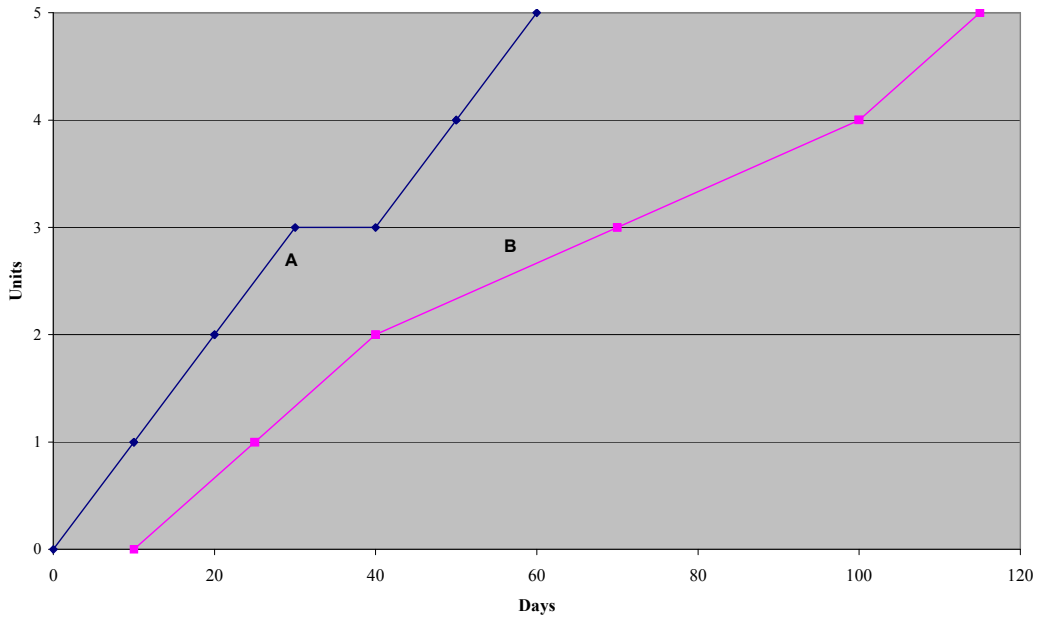


Figure 3.4 Illustrations of two user-specified scheduling considerations

illustrate how this type of breaks in the work sequence can be accommodated in the recurrence equations. Two tasks, A and B each with 5 units are shown in Fig. 3.4. A mandatory work break of 10 days is imposed between units 3 and 4 for activity A. In addition, and the duration of task B is doubled from 15 days to 30 days for units 3 and 4.

The mandatory break requirements can be incorporated into the recurrence equations simply by including a new variable, Π in Eqns. 3.4 and 3.7:

$$EF_{i,j} = \{\underset{\forall p \prec i}{MAX}[EF_{p,j}, EF_{i,j-1}] + T_i\} + \omega_{i,j} + \Pi_{i,j} \tag{3.9}$$

$$LF_{i,j} = \underset{\forall s \succ i}{MIN}[(LF_{i,j+1} - T_i), (LF_{s,j} - T_s)] + \Pi_{i,j} \tag{3.10}$$

The imposition of a mandatory work break between units 3 and 4 is analogous to imposing a delay on the start dates of the activity on unit 4. By subtracting the duration of the task from Eqns. 3.9 and 3.10, similar equations involving the start times of the activities are obtained. Therefore, a delay in the start dates is equivalent to a delay in the finish dates and the modified recurrence equations can incorporate work breaks of *w* days between any two activities A_iU_{j-1} and A_iU_j in the following manner:

$$\prod_{i,j} = w \tag{3.11}$$

Hence, using Eqn. 3.11, the modified recurrence equations can incorporate the mandatory work break between activities A3 and A4 by assigning the value of $\prod_{4,4}$ as 10.

In order to account for a change in work rate for an activity, the duration variable T_i is modified into a duration function $T_{i,j}$. Therefore, the duration function for activity B is represented by a piece-wise function:

$$T_{B,j} = \begin{cases} 30 & for \quad j = (3,4) \\ 15 & for \quad j = (1,2,5) \end{cases}$$

One advantage of this form of representation is that a suitably defined $T_{i,j}$ can be used to simulate the learning curve effect which results in a progressively shorter task duration as long as the work sequence is unbroken.

3.3 Illustrative Example

3.3.1 Scheduling with work continuity requirements

To illustrate the application of the recurrence-equations, consider the network given in Fig. 1.1 and the scheduling data defined in Table 3.2 over five repetitive units.

A computer spreadsheet provides a convenient way of implementing the proposed algorithm. The spreadsheet's intuitive cell-based structure and easy-to-use interface make it suitable for developing a scheduling program model from the set of recurrence equations for simple to moderately complex linear projects. Results are instantly updated when the data input values are changed. Various charts and graphs can also be constructed inside the spreadsheet to present the results. Furthermore, many engineers and project managers are already familiar with the use of a computer spreadsheet having used it as a convenient and productive tool for data processing during the course of their work. It was for these reasons that a simple prototype was

Table 3.2 Scheduling parameters for illustrative example

Tasks	A	B	C	D	E	F	G
Duration (days)	10	6	7	3	5	12	5

	A	B	C	D	E	F	J	K	L	M	N
2							Float				
3	Activity	Duration per repetitive unit d(i)	Repetitive Units	Resource Continuity	Predecessors		Unit 1	Unit 2	Unit 3	Unit 4	Unit 5
4	A	10	5				20	20	20	20	20
5	B	6	5	☑	A		4	8	12	16	20
6	C	5	5	☑	B		30	30	30	30	30
7	D	5	5	☑	B		0	5	10	15	20
8	E	5	5	☑	B		0	5	10	15	20
9	F	10	5	☑	D	E	0	0	0	0	0
10	G	4	5	☑	C	F	0	0	0	0	0

Figure 3.5 Spreadsheet interface for data input

developed using Microsoft Excel for the purpose of performing linear repetitive scheduling.

The spreadsheet has a simple interface as illustrated in Fig. 3.5. This provides the means of entering pertinent regarding the project including the number of tasks, the number of repetitive units for each task and their corresponding unit durations. A set of check boxes makes it simple to choose whether or not to impose the resource continuity constraint for each activity.

Fig. 3.6 illustrates the resulting line schedule plotted from the values calculated using the recurrence equations when work continuity constraints are imposed on all the tasks; the corresponding schedule calculated using the CPM equations is shown in Fig. 3.1. Comparing Fig. 3.6 with Fig. 3.1 indicates that the overall project duration has increased by 16 days (extending the project duration from 96 days to 112 days) or approximately 17 percent. This is expected, as the commencement dates of the first activities of six tasks (tasks B, C, D, E, F and G) are delayed, thus affecting the start dates of subsequent activities. This illustrates the point that imposing work continuity on discontinuous tasks will invariably lengthen the overall project duration.

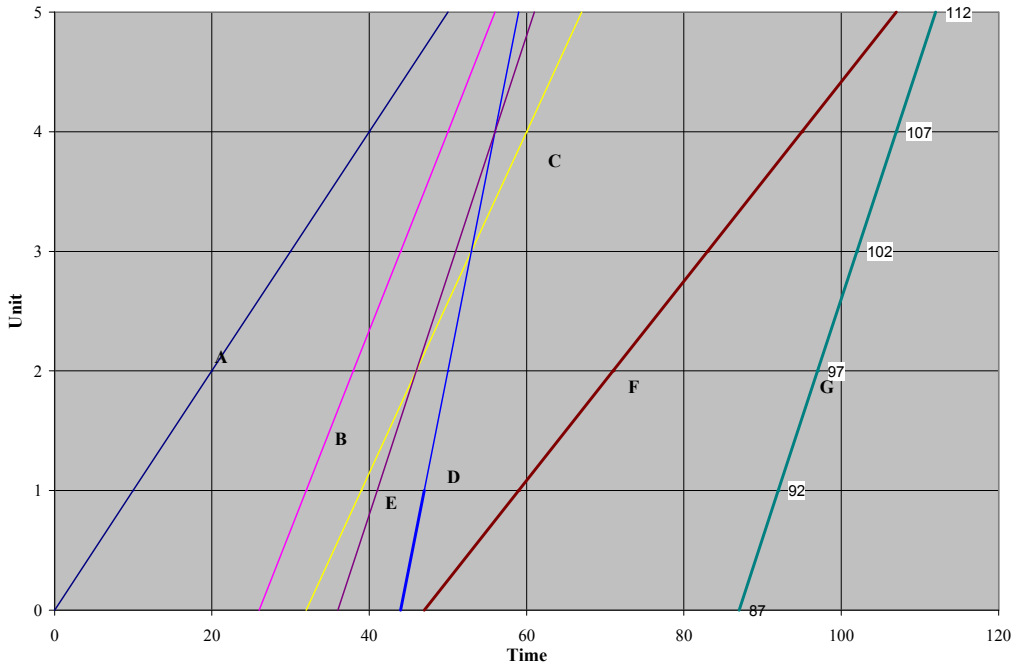


Figure 3.6 Linear schedule calculated using recurrence equations

Besides the line schedule, the built-in functionality of a computer spreadsheet can be used to easily generate other alternative schedule representations. For example, Fig 3.7 illustrates a matrix schedule showing the actual start and finish dates depicted previously in Fig. 3.6.

Another benefit of using the recurrence equations is the ability to determine important scheduling information like the total float for each activity. This allows the project managers to identify the critical units, as well as the maximum allowable delay for each activity unit without jeopardising the project completion. From Fig. 3.7, we can easily identify the set of critical activities as (D1; F1 - F5; G1 - G5). In addition, according to Barrie and Paulson, the critical path is a continuous chain of activities from the beginning to the end of a network with the minimum float value.

Units	A		B		C		D		E		F		G	
	3-Aug	6-Sep	9-Aug	12-Sep	20-Aug	29-Sep	12-Aug	17-Sep	14-Aug	17-Sep	29-Sep	29-Sep	4-Oct	4-Oct
5	24-Jul	27-Aug	3-Aug	6-Sep	13-Aug	22-Sep	9-Aug	14-Sep	9-Aug	12-Sep	17-Sep	17-Sep	29-Sep	29-Sep
	34		34		40		36		34		0		0	
	24-Jul	25-Aug	3-Aug	31-Aug	13-Aug	22-Sep	9-Aug	5-Sep	9-Aug	5-Sep	17-Sep	17-Sep	29-Sep	29-Sep
4	14-Jul	15-Aug	28-Jul	25-Aug	6-Aug	15-Sep	6-Aug	2-Sep	4-Aug	31-Aug	5-Sep	5-Sep	24-Sep	24-Sep
	32		28		40		27		27		0		0	
	14-Jul	13-Aug	28-Jul	19-Aug	6-Aug	15-Sep	**EFD**	**LFD**	4-Aug	24-Aug	5-Sep	5-Sep	24-Sep	24-Sep
3	4-Jul	3-Aug	22-Jul	13-Aug	30-Jul	8-Sep	**ESD**	**LSD**	30-Jul	19-Aug	24-Aug	24-Aug	19-Sep	19-Sep
	30		22		40		**Floats**		20		0		0	
	4-Jul	1-Aug	22-Jul	7-Aug	30-Jul	8-Sep	3-Aug	12-Aug	30-Jul	12-Aug	24-Aug	24-Aug	19-Sep	19-Sep
2	24-Jun	22-Jul	16-Jul	1-Aug	23-Jul	1-Sep	31-Jul	9-Aug	25-Jul	7-Aug	12-Aug	12-Aug	14-Sep	14-Sep
	28		16		40		9		13		0		0	
	24-Jun	20-Jul	16-Jul	26-Jul	23-Jul	1-Sep	31-Jul	31-Jul	25-Jul	31-Jul	12-Aug	12-Aug	14-Sep	14-Sep
1	14-Jun	10-Jul	10-Jul	20-Jul	16-Jul	25-Aug	28-Jul	28-Jul	20-Jul	26-Jul	31-Jul	31-Jul	9-Sep	9-Sep
	26		10		40		0		6		0		0	

Activity

Figure 3.7 Matrix schedule for illustrative example

Accordingly, the critical path is defined by the set of activities (A1; B1; D1; F1 – F5; G1 – G5) with the minimum float value of 36 days. However, this critical path consists of non-critical activities A1 and B1.

The controlling sequence (CS) (Harris and Ioannou, 1998) and the controlling activity path (CAP) (Harmelink and Rowings, 1998) provide useful comparisons to the critical units identified using the proposed recurrence equations. Following the respective procedures, the set of activities in the CS and CAP are (A1 - A5; B5; D2 - D4; F1 - F5; G5) and (A1 - A3; B1 - B3; D1; F1 - F5: G5) respectively. It is obvious that the activities in the critical path, CS and CAP are not all the same.

Both the CS and CAP include activity A1, which is a non-critical unit (with non-zero total float) according to the total floats calculated using the recurrence equations. Suppose there is a 10-day delay in the completion of activity unit A1. From Fig. 3.8, only the start times for the last activities of tasks B, C and D are affected by this

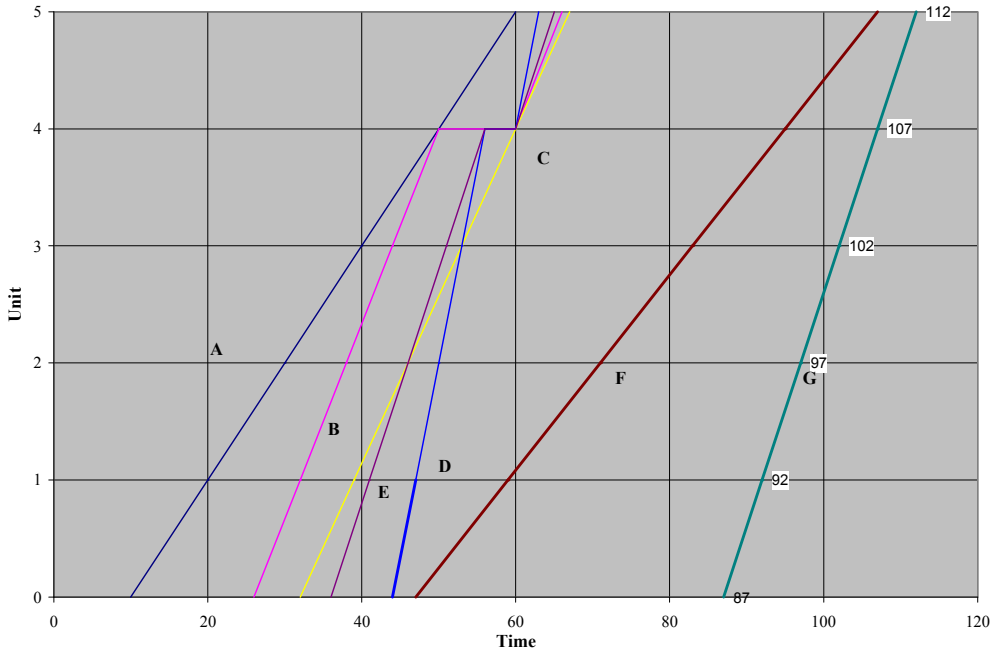


Figure 3.8 Impact of a 10-days delay on activity A1

delay. The overall project completion date remains unchanged at 112 days. Therefore, activity A1 is not critical, and the matrix schedule in Fig. 3.8 shows that activity A1 can be delayed by up to 26 days without jeopardising the completion date of 112 days. This is contrary to the impression given by the inclusion of A1 in CS and CAP.

The situation is reversed in the case of activities G1 to G4 – these activities are identified as critical by the recurrence equations but are not included in either CS or CAP. From Fig. 3.6 it can be observed that a delay in any one of these activities will cause a corresponding delay in the completion of activity D5 and thus jeopardize the timely completion of the project. Therefore, the proposed recurrence equations can correctly identify critical activities by adhering to the classical definition of a critical activity as an activity with zero total float.

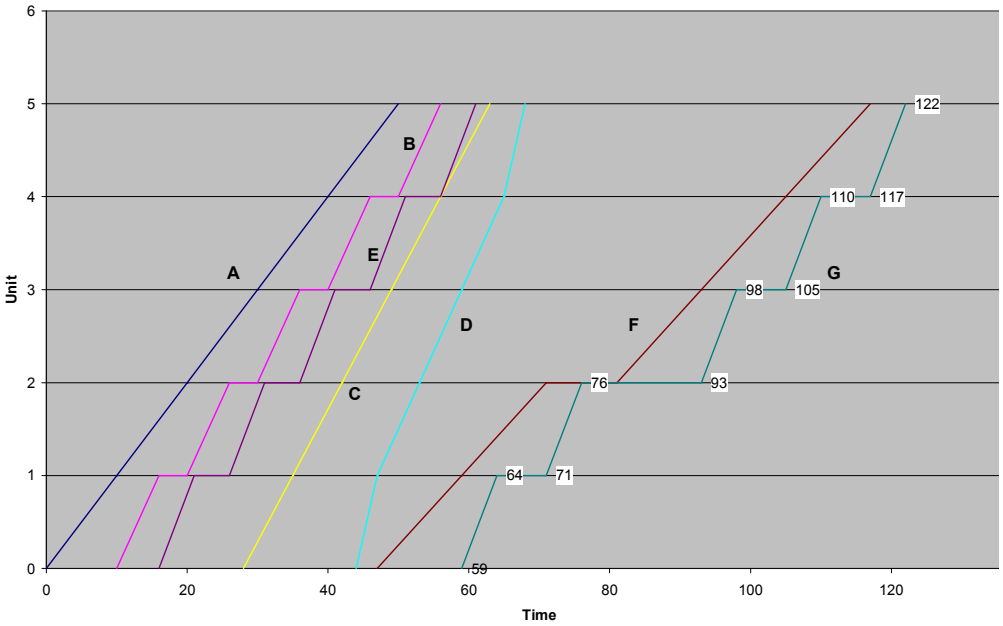


Figure 3.9 Linear schedule calculated using modified recurrence equations and duration function

3.3.2 Scheduling with additional user-specified constraints

The following example illustrates the application of the recurrence equations when there are specific scheduling considerations like mandatory work breaks and a change in the work rate for particular activities.

Consider that work continuity constraints are now imposed only on tasks C and D. In addition, the project manager knows that there will be a 10-day break (from days 71 –

Table 3.3 Float table for schedule with user-specified scheduling requirements

	Unit				
Task	**1**	**2**	**3**	**4**	**5**
A	36	37	39	41	44
B	36	37	39	41	44
C	54	54	54	54	54
D	10	13	19	25	37
E	36	38	40	42	44
F	10	10	**0**	**0**	**0**
G	38	31	14	7	**0**

81) in the work sequence of task F. Also, the workload for task D is doubled from units 2 to 4, so that it takes twice as long to complete an activity. Fig. 3.9 depicts the new schedule obtained with an overall completion date of 122 days using the set of modified recurrence equations (Eqns. 3.9 and 3.10). Finally, Table 3.3 shows the total floats for all the activities and identifies the critical activities to be (F3 - F5; G5).

CHAPTER 4

MODEL FOR SCHEDULE OPTIMIZATION

This chapter describes the model for schedule optimization using the proposed recurrence equations with Genetic Algorithms (GA). The basic mechanisms of GA are first introduced to facilitate subsequent discussion on the utilization of the recurrence equations for performing schedule optimization with GA. Specifically, the process of identifying the two relevant decision variables for scheduling repetitive projects is described, and the mapping of these decision variables onto the terms in the recurrence equations in alternative GA representations are discussed. The scheduling constraints applicable to forming schedules for repetitive projects are also considered. Two evaluation criteria, combined into a single objective function, are used to assess the merit of the alternative schedules identified by the GA-based schedule optimization procedure.

4.1 Basic Mechanisms of the Genetic Algorithms

Genetic algorithms (GA) are stochastic search methods introduced in the 1970s in the United States by John Holland (1975). Search methods are relevant in a wide variety of engineering and management problems which require the identification of solutions to a specified problem. Each individual solution is represented by a single string-like entity called a chromosome. Fig. 4.1 depicts a chromosome made up of genes characterized by their positions (physical ranks) and values (alleles). The values for key decision variables that comprise a solution to the problem are encoded as the values of the gene alleles.

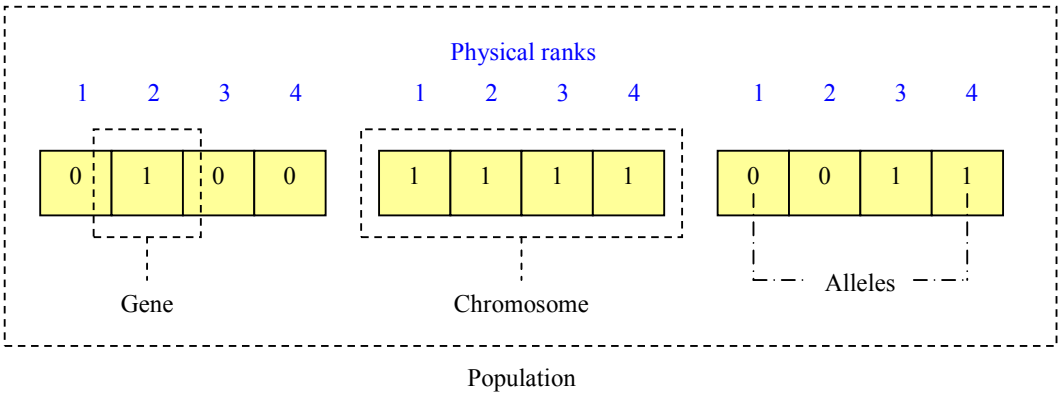


Figure 4.1 Building blocks of genetic algorithms

Generally, an initial population of potential solutions is randomly generated at the start of a GA optimization process. Each of the chromosome strings is assigned a fitness value determined by the quality of the solution to the problem encoded in the allele values of the chromosome string. The fitness value is determined using a problem specific *objective function*, and the fitness value determines the reproductive chances of the chromosome string. Fitter chromosome string individuals are given a higher chance of being selected to participate in the next step of the GA process when new individuals are created. These new "offspring" are created by applying the *crossover* operator on the chromosomes that are chosen to "reproduce" by a *selection* procedure. The mutation operator is then applied to randomly alter the composition of some of these offspring. The genetic operations of selection, crossover and mutation occur on the population of chromosomes over and over again, transforming the initial population of chromosomes to new populations in each succeeding "generation". In common with natural evolutionary selection, the GA selection process improves the average fitness of the population with each generation. This process continues until a user-defined criterion is reached.

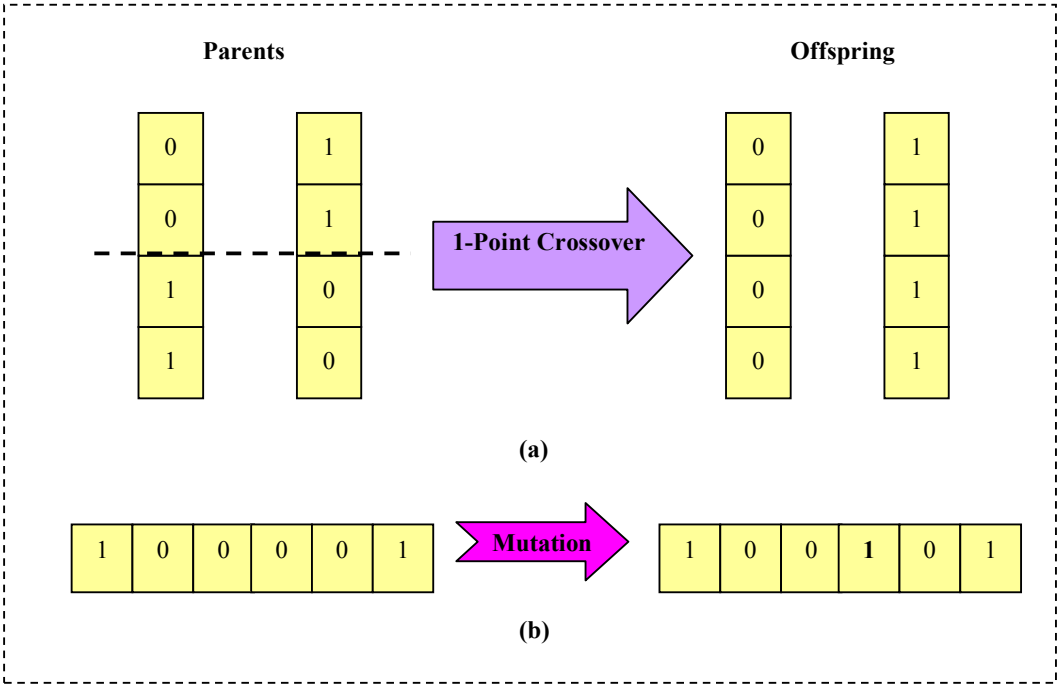


Figure 4.2 Illustrations of (a) one-point crossover and (b) mutation

The *objective function*, which quantifies the merit of a solution, is an important part of the design of a GA model. Better solutions benefit the chromosome strings that encode them by increasing the chance of being selected for the crossover operation. When GA search is applied to optimization problems, higher fitness values correspond to better solutions although it is not possible to guarantee global optimality.

The *selection* procedure represents the concept of *survival of the fittest* in GA. It usually involves a weighted function, where individuals with higher fitness are more likely to reproduce. There are several well-defined selection methods and the Roulette Wheel Selection is one of the most common techniques. The analogy to a roulette wheel can be envisaged by imagining a roulette wheel in which each candidate solution represents a pocket on the wheel; the size of the pockets are proportionate to the fitness of the solution. Selecting N chromosomes from the population is

equivalent to playing N games on the roulette wheel, as each candidate is drawn independently.

The *crossover* operation is then performed on the chromosomes selected for reproduction. The probability of crossover determines the likelihood that two selected chromosomes will actually "breed" through the crossover operation. Hence, mating between chromosomes is still governed by chance - the chromosomes are mated if a generated random number falls below the crossover threshold; otherwise, they are propagated into the next generation unchanged. The chromosomes of the parents are mixed in some way during crossover, typically by simply swapping a portion of the underlying data structure, and this results in two new offspring which are added to the second generation pool. This process is repeated with different parent chromosomes until there are an appropriate number of candidate solutions in the second generation pool. Fig. 4.2(a) illustrates a one-point crossover. The crossover probability can be adjusted to improve the performance of the GA.

The *mutation* operator re-introduces genetic diversity into the population. It operates upon single chromosomes by randomly changing the value of their bits. This change can be highly destructive to good chromosomes but is essential to prevent the risk of convergence upon a sub-optimal solution. Therefore, the probability of mutation is often very low in order to avoid the disruption of good solutions. Figure 4.2(b) illustrates a simple case of mutation.

4.2 Development of the GA-based Optimization Model

4.2.1 Organizational setup for repetitive construction projects

The organizational setup in a repetitive project is generally defined by the contractual relationships between various parties involved in the projects, and is relevant to the objectives identified for the repetitive scheduling problem. Typically, a general contractor undertakes the entire construction project from the owner. The majority of the work is then broken down into specific trades to be performed by individual specialist subcontractors under subcontracts to the general contractor. Although the subcontractor normally bids upon a portion of the owner's plans and specifications, their legal contractual relationships are directly with the general contractors; the latter is in turn responsible to the owner for all the work, including that which is subcontracted. Fig. 4.3 illustrates this form of arrangement.

The single fixed-price contract is the traditional contractual arrangement that is

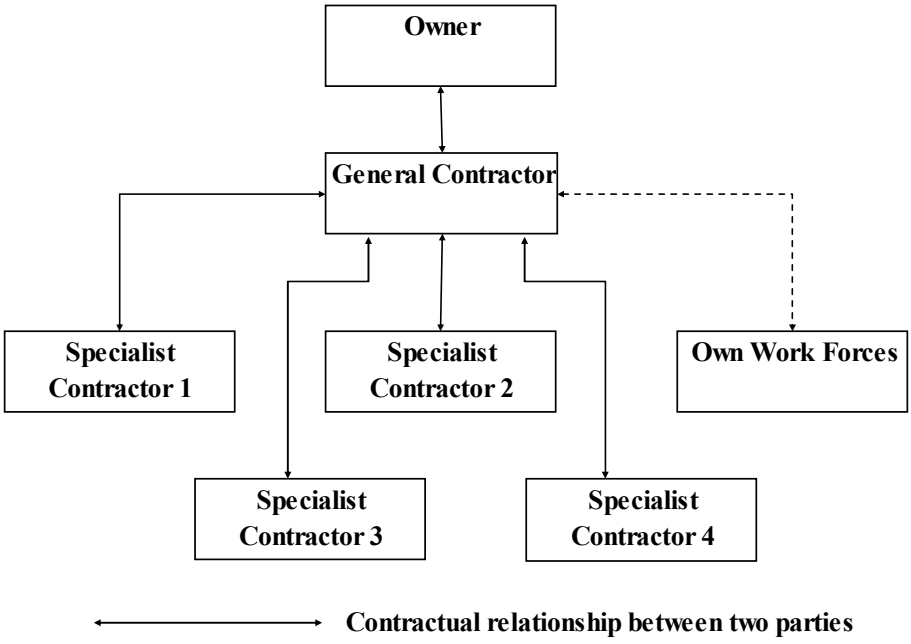


Figure 4.3 Typical organizational setup in a construction project

administered for construction projects that involve competitive bidding (Barrie and Paulson, 1992). In this fixed-price form, the contractor agrees to perform the work for a predetermined price that includes profit. The contractor also bears the economic risk for any project tardiness, which generally involves paying a specific amount of monetary compensation to the owner for each day of delay. Therefore, from the perspective of the general contractor, the scheduling objective is to ensure timely project completion.

The majority of specialist subcontractors are however, concerned with the productivity of their essential workers. For business reasons, they will want to keep this core group as lean as possible, as well as ensure that there are a sufficient number of on-going and future projects that these workers can be productively employed. The subcontractors in a survey conducted by O'Brien and Fischer (2000) noted that various undesirable site conditions have major effects on their incurred costs, and work discontinuity is one such condition. Specifically, additional costs are incurred in the form of lost wages paid to unproductive workers during the idle periods when they wait for preceding tasks to be completed. The subcontractors have also been reported to consciously shift their workers away from sites where poor scheduling arrangements make it difficult for their workers to be fully productive. Therefore, from the perspective of the subcontractors, the scheduling objective is to ensure an uninterrupted and continuous workflow so that they can complete their individual tasks without incurring unnecessary lost wages.

However, the schedule conditions necessary for maintaining work continuity for all the different trades / tasks and for ensuring early project completion are inherently

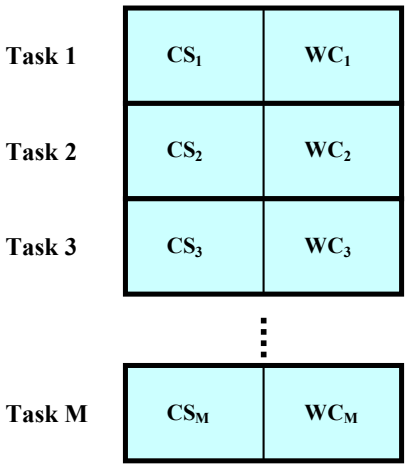
Decision Variables	GARA-I	GARA-II
(1) Crew size	$CS_i = \{x : x \in (1,2,...X)\}$ *where* $X = Total\ number\ of\ crew\ size\ options$	
Term in recurrence equations mapped to *crew size*: **T_i in Eqns. 3.4, 3.7 and 3.8**	T_i = *duration of the crew size selected*	
(2) Work continuity	$WC_i = \begin{cases} 1 \\ 0 \end{cases}$	$WC_i = \{y : y \in (0,[Q-1]MAX[0, D_i - T_i]); y \in Z\}$
Term in recurrence equations mapped to *work continuity*: **$\omega_{i,1}$ in Eqn. 3.7**	$\omega_{i,1} = \begin{cases} (Q-1)MAX[0,(D_i - T_i)] & for \quad WC_i = 1 \\ 0 & for \quad WC_i = 0 \end{cases}$	$\omega_{i,1} = WC_i$
(3) Graphical representation		

Figure 4.4 GARA-I and GARA-II chromosome representations

conflicting. In assuming that all the tasks involved in a project are carried out with the maximum number of workers and that each activity commences on the earliest possible date, timeliness in project completion is achieved at the expense of a fragmented workflow for some tasks. On the other hand, the ideal condition where subcontractors can "get on a task, work straight through it and leave" will involve imposing some delays on the first activities of these tasks. This will usually extend the project duration, thus jeopardizing timely project completion.

The selection of crew size and the imposition of a work continuity requirement ultimately influence the schedule generated. This study uses these two decision variables as the means of creating alternatives for compromise schedules that benefit both the general contractor and his specialist subcontractors.

4.2.2 GA model for scheduling repetitive projects

In general, maintaining work continuity creates a conflict of interest between the general contractor and the specialist subcontractors who are responsible for the individual work tasks. Whilst work continuity benefits the subcontractors by minimizing the idle time of their crews, it usually lengthens the overall project duration. This is detrimental to the main contractor, whose aim is to minimize the project makespan, especially in light of the tardiness penalty. This conflict of interest necessitates the search for scheduling arrangements that reap the maximum benefit from any work continuity requirement whilst mitigating tardiness costs for the main contractor.

A GA Recurrence-equations Approach (GARA) is proposed to seek this 'optimal' arrangement. Firstly, the decision variables of crew size and work continuity are represented by the terms T_i *and* $\omega_{i,1}$ respectively in the proposed recurrence equations. In this way, different values for the variables will lead to different schedules with various states of work continuity and project lengths. GA is then used to determine the optimal values for these two variables for each of the different tasks so that schedules with a tradeoff between continuous workflow and project duration can be found. Two ways of expressing work continuity in the GA chromosome are explored. GARA-I only considers full work continuity for each task but GARA-II makes it

possible to impose partial work continuity. It is not clear if one representation is inherently better than the other and experiments are conducted to compare their performance. Fig. 4.4 illustrates the two chromosome representations.

The chromosomes in GARA-I and II consist of *M* gene-pairs, where each gene-pair represents the two decision variables of work continuity and crew size for a particular task. The crew size decision variable is identical in both GARA-I and GARA-II, and the integer value of the crew size variable (in the range of one to *X)* represents a particular choice of the crew size for the task. This crew size implies a corresponding number of workers and duration for the task, T_i in the recurrence equations for the scheduling calculations.

In GARA-I, work continuity is modelled as a binary variable that indicates whether work continuity is imposed on a particular task. A value of one for this binary decision variable indicates that work continuity is imposed on the task. If work continuity for a task is imposed, the necessary delay in the first activity of the task to achieve work continuity, $\omega_{i,1}$ is calculated using:

$$\omega_{i,1} = \begin{cases} (Q-1)MAX[0,(D_i - T_i)] & for \quad WC_i = 1 \\ 0 & for \quad WC_i = 0 \end{cases} \tag{4.1}$$

Consequently, the schedules found using GARA-I will consist of tasks with either fragmented or fully continuous workflows.

In contrast, GARA-II allows GA to explore the possibility of finding better schedules by imposing partial work continuity. In the case of partial work continuity, the delay in the start of the first activity of the task can range from zero to the value computed

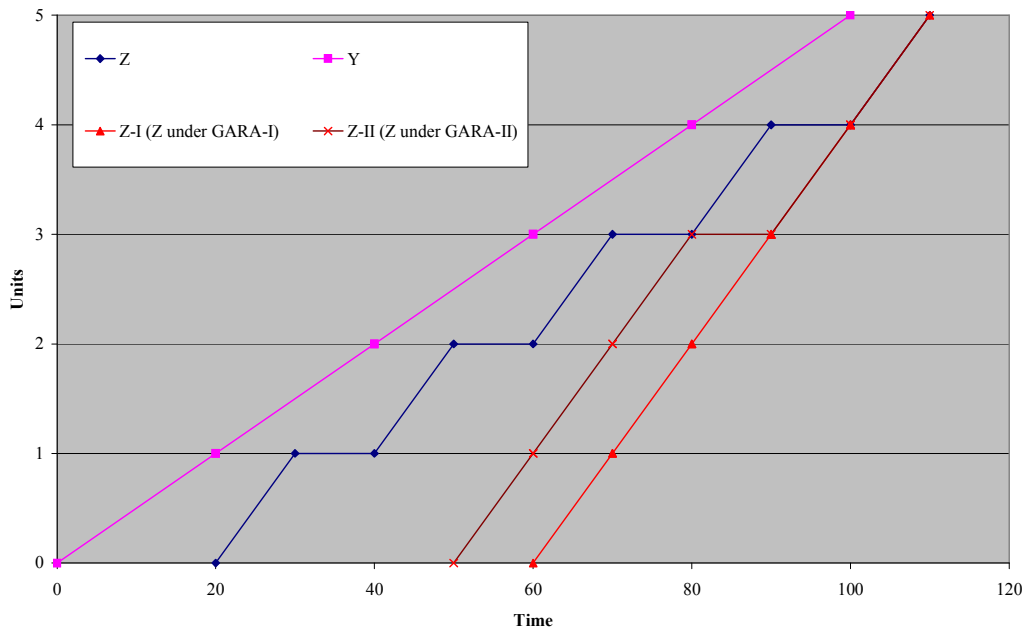


Figure 4.5 Comparison of the different state of work continuity represented in GARA-I and GARA-II

using Eqn. 4.1. This gives the GA search procedure more options to explore when looking for better schedules.

Fig. 4.5 shows different schedules obtained using GARA-I and GARA-II on a simple two-task configuration where task Y precedes task Z. The durations of these tasks are 20 days and 10 days respectively. Without the work continuity requirement imposed on Z, the schedule for Z would be the broken blue line labelled 'Z'. As indicted by that particular line, Z1 can only start after Y1 finishes (at t = 20); similarly, Z2 can only start after Y2 finishes (at t = 40) resulting in the first break in work continuity for task Z. A delay of 40 days has to be imposed for Z1 under GARA-I to effect work continuity for task Z, resulting in the schedule for Z denoted by the line labelled 'Z-I'.

Under GARA-II, the start of Z1 can be delayed anywhere from zero to 40 days. A delay of 30 days has been used in the schedule for Z labelled 'Z-II'. In this case, the

work continuity for task Z-II only extends from activity Z1 to activity Z4 and there is a break of 10 days between activities Z4 and Z5.

4.2.3 Scheduling constraints

Repetitive project scheduling requires allocating resources over time to a set of tasks while satisfying a variety of constraints and objectives. Hard constraints must always be satisfied for a schedule to be valid. Soft constraints on the other hand, can be relaxed when necessary. For this optimization model, the precedence and resource availability constraints are treated as binding while the due date and work continuity constraints can be relaxed.

Precedence constraints define the logical interrelationships among the project tasks in a construction project. This constraint requires that a particular activity is started only after all its preceding activities are completed. For instance, activity Y1 in Fig. 4.5 commences at the start of the project (since Y1 has no predecessors) whilst activity Z1 can only begin after the completion of activity Y1. This logical sequence of workflow from one activity to another must be strictly adhered, and any schedule that violates the precedence constraints is invalid. In the GA schedule optimization model, all predefined precedence constraints are coded into the recurrence equations and enforced during the course of the scheduling calculations to ensure that these constraints are always observed in the process of optimization.

The resource availability constraints arise when there is only one available group of work crew for each task, and as a result, an activity cannot commence until the work crew is available again, usually through the completion of the preceding activity of

the same task. Therefore, activity Z2 in Fig. 4.5 cannot begin until activity Z1 is completed. Like the precedence constraints, the resource availability constraints are coded in the recurrence equation and enforced during the course of the scheduling calculations.

On the other hand, the due date constraint and the work continuity requirement are treated as soft constraints. Soft constraints are handled by including them in the definition of the objective function used for the optimization procedure.

4.2.4 Objective function

1. Tardiness penalty

The due date constraint specifies the requirement for a timely completion of the project. This constraint can be expressed as an inequality between the latest finish dates for the last activity of the project and the contractual deadline for the completion of the project, S:

$$LF_{M,Q} \leq S \tag{4.1}$$

Violation of this constraint will not invalidate a schedule but does incur penalty costs for late completion beyond the target schedule data.

This inequality is incorporated into the objective function as a penalty term, Z_T:

$$\text{Min } Z_T = C_T \times Max(0, LF_{M,Q} - S) \tag{4.2}$$

2. Work discontinuity penalty

The total number of days lost to work discontinuity can be expressed as:

$$(Q-1)\sum_{i=1}^{M} MAX[0,(D_i - T_i)] \quad (4.3)$$

This can be translated into man-days of wages lost by multiplying by the daily wage-rate of the workers, C_i and the number of workers on the task, K_i. The total lost wages due to work discontinuity can be incorporated into the objective function as another penalty term:

$$\text{Min } Z_D = (Q-1)\sum_{i=1}^{M} \left(MAX[0,(D_i - T_i)] \times K_i \times C_i\right) \quad (4.4)$$

In order to obtain compromise schedules that balance the objectives of both the general contractor and his subcontractors, the objective function is defined as the minimization of the sum of the two penalty terms:

$$\text{Min } Z = Z_T + Z_D \quad (4.5)$$

4.3 GA Parameters

The optimal values for several GA parameters are difficult to determine. These parameters include the population size, the number of iterations performed, the crossover rate, the mutation rate and the termination criterion. The process of determining the default values of some of these parameters are discussed.

Table 4.1 summarizes the average time to convergence for a set of GA experiments conducted for a 40-unit repetitive project using different GA parameters. Convergence is reached when the GA attains a benchmark value. This benchmark is obtained by selecting the best solution obtained from twenty different GA runs. Each GA run iterates over 500 generations and begins with a randomly generated initial population. Ten separate runs are conducted for each set of GA parameters, the

Table 4.1 Average time to convergence with different GA parameters

Population size	20						50						70					
Crossover prob. (%)	85		90		95		85		90		95		85		90		95	
Mutation prob. (%)	5	1	5	1	5	1	5	1	5	1	5	1	5	1	5	1	5	1
Freq. of convergence (times)	7	6	8	6	8	7	9	7	10	9	10	10	10	9	10	10	10	10
Ave. time to convergence (sec)	38	43	39	45	38	46	72	85	78	80	81	89	143	159	154	166	159	164

number of times out of the ten runs where the solutions converge to the benchmark value and the time taken for each convergence are noted (a maximum computation time of ten minutes is set). The average time to convergence is calculated by taking the runs when convergence was achieved and averaging the times taken. For example, under the population size, crossover probability and mutation probability of 20, 85% and 5% respectively, seven out of the ten runs produce schedules with fitness values that converge to the benchmark value. The sum of the time taken for these seven convergences is 265 seconds and this returns an average time to convergence of 38 seconds (265 / 7). Runs that return solutions inferior to the benchmark after ten minutes are not taken into account when calculating the average time to convergence.

From the results in Table 4.1, values for the population size (50), one-point crossover probability {90%} and mutation rate (0.05) are selected as they return the highest frequency of convergence and the lowest average time to convergence; the termination criterion is set to 500 iterations. All the subsequent experiments are conducted with these GA parameters.

The GA runs are made using a commercially available GA toolbox running on a Pentium IV 1.2GHz desktop is used in this study. The next chapter discusses the results obtained when the performance of GARA is investigated with respect to cases of increasing number of repetitive units and different due date constraints.

CHAPTER 5

RESULTS FROM SCHEDULE OPTIMIZATION

This chapter describes several experiments conducted to assess the performance of the GA-based optimization model (GARA) and presents an analysis of the results obtained. The performance issues covered include: (i) relative performance of GARA vis-à-vis the Critical Path Method Approach (CPMA) and Graphical Scheduling Approach (GSA) on projects with an increasing the number of repetitive units under different due date constraints; (ii) a comparative merits of two alternative chromosome representations; and (iii) the time to convergence of GARA on networks of different complexity.

5.1 Experimental Setup

Table 5.1 summarizes the scheduling parameters of a project used in the experiments. The network schedule, first depicted in Fig. 1.1, consists of seven individual tasks with three possible crew sizes each. The crew size selection will affect the task duration so that employing more workers will reduce the amount of time required to

Table 5.1 Scheduling parameters of crew size options and the corresponding task durations

	Predecessors	**Crew Size Option**						**Basic daily wages**
		1		2		3		
		Crew size	Duration	Crew size	Duration	Crew size	Duration	
A	-	9	9	7	10	5	13	55
B	A	6	5	5	6	4	7	60
C	B	4	5	3	7	2	10	70
D	B	13	2	8	3	6	5	58
E	B	3	3	2	5	1	7	62
F	D, E	8	10	5	12	3	15	64
G	C, F	4	4	3	5	2	7	43

Parameters:
Tardiness penalty = $2500 / day

Table 5.2 Representations of the decision variables in CPMA and GSA chromosomes

DECISION VARIABLES	CPMA	GSA
(1) Crew size option	$CS_i = \{x : x \in (option\ \ 1, option\ \ 2, ..., option\ \ n)\}$	
Term in recurrence equations mapped to *crew size option*	T_i= *duration corresponding to the crew size option*	
(2) Work continuity	$WC_i = 0$	$WC_i = 1$
Term in recurrence equations mapped to *work continuity*	$\omega_i = 0$	$\omega_{i,1} = (Q-1)MAX[0,(D_i - T_i)]$

complete an activity. The daily wages for each worker employed in any of the seven trades range from $43 to $70, and each day of delay in the project completion incurs a "penalty" cost of $2,500.

As mentioned previously, GARA works out which task should have work continuity and there are two possible options to be investigated - full work continuity (GARA-I) or partial work continuity (GARA-II). One set of experiments is performed to investigate whether full or partial work continuity is preferable. The performance of GARA for schedule optimization is compared against that of CPMA and GSA.

To compute the schedule under CPMA and GSA, the same set of recurrence equations can be used except that there is no requirement for work continuity on any task under CPMA whereas *all* the tasks involved have continuous and uninterrupted workflows under GSA. Setting the decision variables for work continuity in the optimization model to the value of zero (for CPMA) and one (for GSA) takes care of this requirement. The optimization procedure still needs to determine the best crew size

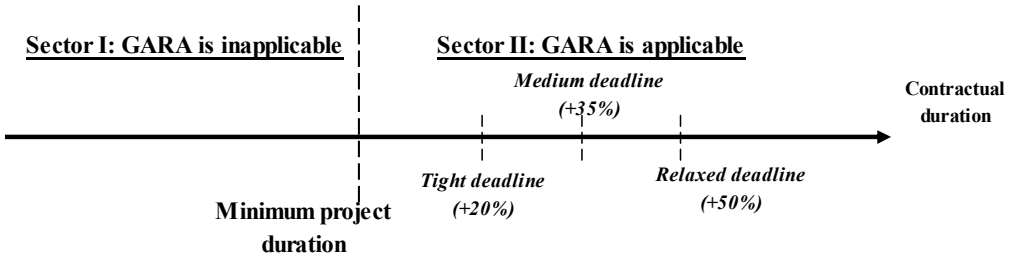


Figure 5.1 Applicability of GARA with respect to contractual duration

under CPMA and GSA. Table 5.2 summarizes the representations of the decision variables in CPMA and GSA chromosomes.

In performing the optimization, the experiments are conducted using three different due date constraints for five projects with a different number of repetitive units in the project. The setting of the due dates for the experiments requires some explanation.

Fig. 5.1 illustrates the point that GARA is applicable to repetitive schedule optimization only when the stipulated project duration is longer than the minimum project duration. The minimum duration for a project is defined as the project duration calculated using the CPM equations under the assumption that (i) every task uses the largest crew size and therefore completes in the minimum duration and (ii) work continuity is not imposed on any task. For example, the minimum duration calculated using Eqn. 3.3 for a 5-unit project with the scheduling parameters in Table 5.1 is 71 days.

GARA is not applicable to projects where the specified project duration is shorter than the minimum project duration as changing the crew size selection for tasks will not enable GARA to find a schedule that is shorter than the minimum project

duration. Imposing work continuity on project tasks will usually lengthen the project duration.

GARA can be used to optimize the schedule for a repetitive project when the specified project duration exceeds the minimum project duration. In this case, the slack time to project completion (given by the difference between the stipulated duration and the minimum duration) provides the opportunity to effect work continuity for project tasks. GARA can be used to determine the best crew size for each task as well as the appropriate time to deploy them the crew for the first activity of a task in order to minimize project tardiness and the periods of discontinuous workflows. It could be said that GARA works out the best allocation of the slack time to project completion.

Three types of due date constraints were conceived to investigate GARA's performance under deadlines. The three due date constraints were defined by increasing the project duration by a certain percentage of the minimum project duration. The three deadlines are called tight (+20%), medium (+35%) and relaxed (+50%). In this way, the experiments could study the effect of the different kinds of deadlines independently of the number of repetitive units in the project.

For each case, the best solution from ten separate GA runs with randomly generated initial populations is noted. Table 5.3 summarizes the results obtained from the experiments when the number of repetitive units and deadlines were varied. These results are discussed in the next section.

Table 5.3 Results for various optimization methods over various repetitive units and due date constraints

Repetitive Units	5			10			20			30			40		
Due date constraint	Tight	Medium	Relaxed	Tight	Medium	Relaxed	Tight	Medium	Relaxed	Tight	Medium	Relaxed	Tight	Medium	Relaxed
Stipulated completion days	85	96	106	145	163	181	265	298	331	385	433	481	505	568	631
(I) GARA-I															
(i) Best fitness value ($)	2784	0	0	6264	0	0	13224	2500	0	20184	10092	0	27144	13572	0
(ii) Total project duration (days)	85	91	102	145	160	160	265	299	300	385	413	442	505	543	579
(iii) Tardiness penalty ($)	0	0	0	0	0	0	0	2500	0	0	0	0	0	0	0
(iv) Number of days with work discontinuity	8	0	0	18	0	0	38	0	0	58	29	0	78	39	0
(v) Work discontinuity penalty ($)	2784	0	0	6264	0	0	13224	0	0	20184	10092	0	27144	13572	0
(II) GARA-II															
(i) Best fitness value ($)	2390	0	0	7008	0	0	14022	2500	0	27338	12363	0	31130	18632	0
(ii) Total project duration (days)	85	91	102	145	160	160	265	299	300	385	413	422	505	543	579
(iii) Tardiness penalty ($)	0	0	0	0	0	0	0	2500	0	0	0	0	0	0	0
(iv) Number of days with work discontinuity	8	0	0	22	0	0	44	0	0	89	35	0	97	55	0
(v) Work discontinuity penalty ($)	2390	0	0	7008	0	0	14022	0	0	27338	12363	0	31130	13572	0
(III) GSA															
(i) Best fitness value ($)	10000	0	0	35000	0	0	85000	2500	0	135000	15000	0	185000	27500	0
(ii) Total project duration (days)	89	90	101	159	162	175	299	299	301	439	439	443	579	579	621
(iii) Tardiness penalty ($)	10000	0	0	35000	0	0	85000	2500	0	135000	15000	0	187500	27500	0
(iv) Number of days with work discontinuity	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(v) Work discontinuity penalty ($)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(IV) CPMA															
(i) Best fitness value ($)	9016	9016	9016	20286	20286	20286	42826	42826	42826	65366	65366	65366	87906	87906	87906
(ii) Total project duration (days)	80	80	80	130	130	130	230	230	230	330	330	330	430	430	430
(iii) Tardiness penalty ($)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(iv) Number of days with work discontinuity	44	44	44	99	99	99	209	209	209	319	319	319	429	429	429
(v) Work discontinuity penalty ($)	9016	9016	9016	20286	20286	20286	42826	42826	42826	65366	65366	65366	87906	87906	87906

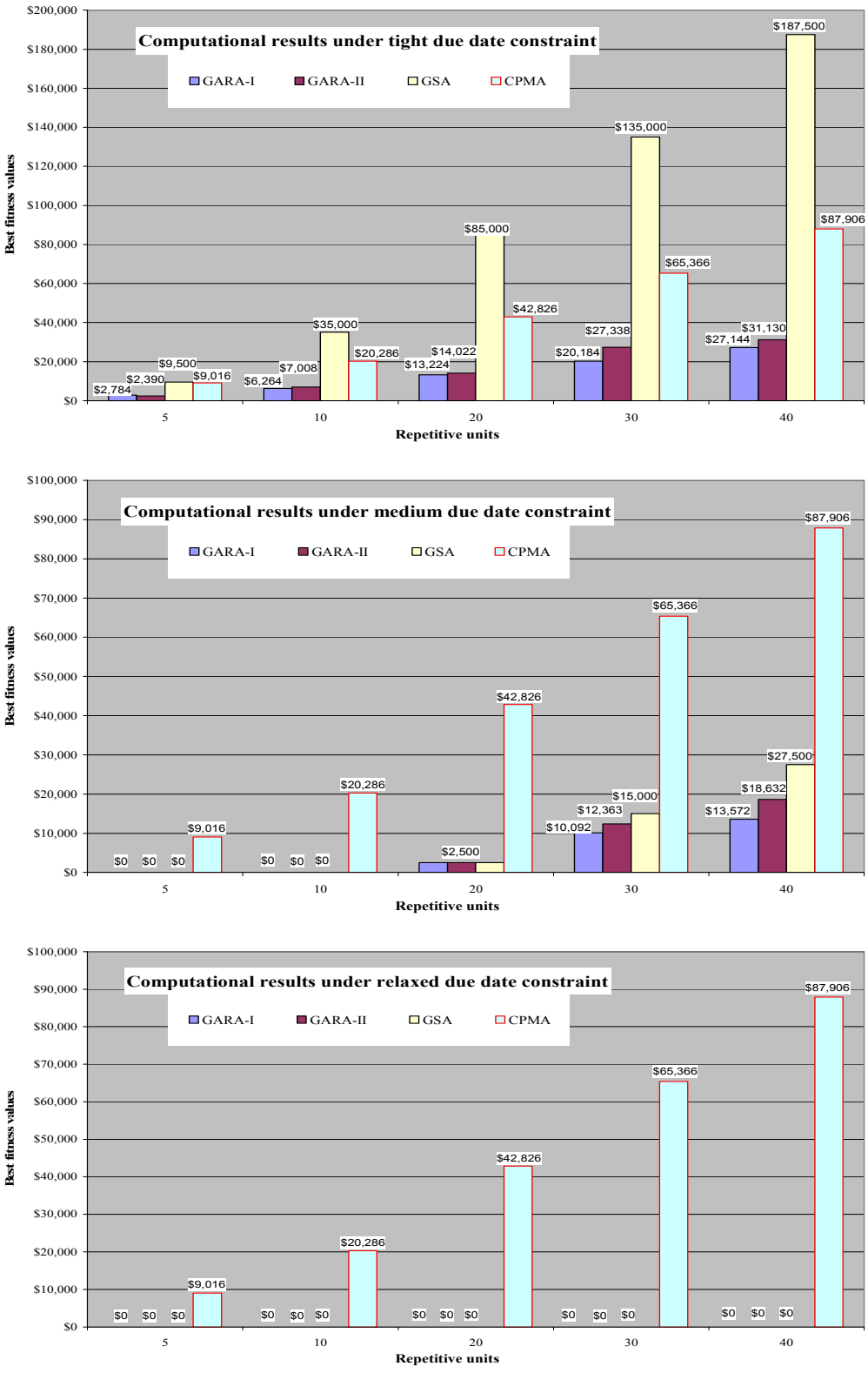


Figure 5.2 Results for various optimization methods under (a) tight, (b) medium, (c) relaxed deadline constraints

5.2 Analysis of Results

5.2.1 General comments

Fig. 5.2 shows the best solutions obtained by GARA, CPMA and GSA under the three different deadline requirements. Good schedules are those that allow punctual project completions while maintaining continuous workflow for each individual tasks and one schedule is better than another if it incurs lower penalties due to project tardiness and work discontinuity. The difference in performance between GARA-I and GARA-II is relatively small, indicating that the schedules produced by both representations are comparable in terms of the ability to complete projects on time and maintaining work continuity. For convenience, GARA is henceforth used to refer to GARA-I. In general, GARA consistently produces schedules with a higher figure of merit than either CPMA or and GSA; this is especially so under tight deadlines. Under medium deadlines, GARA performs only as well as GSA when the number of repetitive units is low but outperforms both CPMA and GSA when the number of repetitive units increases. Finally, both GARA and GSA are able to find schedules that both meet the deadline and do not incur any work discontinuity penalty. On the other hand, CPMA performs poorly under relaxed deadlines.

Table 5.3 shows values for the (i) total project duration and (ii) tardiness penalty incurred for schedules computed using the different methods. GARA schedules have the best performance index value because GARA is able to minimize the periods of work discontinuity whilst still meeting the specified due date for project completions (except for one case where a project was late by one day). GARA consistently incurs lower work discontinuity penalties compared to CPMA. At the same time, GARA

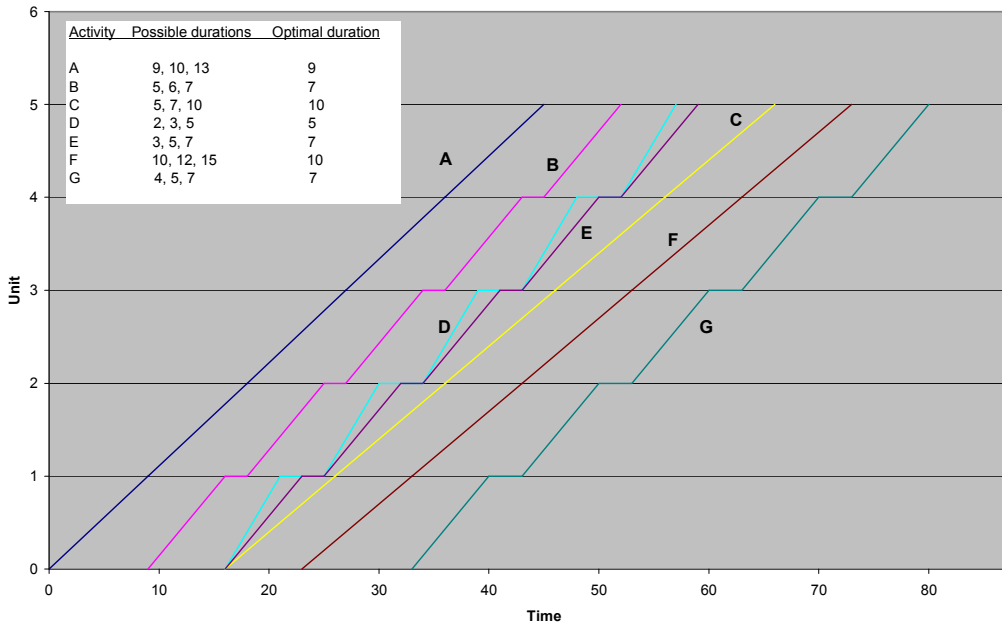


Figure 5.3 Linear schedule for a 5-unit project using CPMA

also maintains an excellent record of completing the projects on time except for one instance where a tardiness penalty for one day is incurred.

It is not possible to conclude that GARA always outperforms CPMA and GSA as the present results could be influenced by the relative magnitude of the unit penalty costs assumed in the objective function. The work discontinuity penalty reflects the wage losses due to unproductive workers during idle periods, and its magnitude is dependent on the wage rate assumed. The tardiness penalty represents the amount of money payable by the general contractor when the project is not completed on time, and its value is depends on the daily tardiness penalty assumed. Due to the difference in magnitudes of the basic wages - ranging from $43 to $70 per person-day – and the daily tardiness penalty ($2,500) assumed, the objective function is weighted to favour the minimization of the tardiness penalty. This is reflected in the results obtained by GARA which gives schedules with no tardiness penalty.

Table 5.4 Optimization results for a 5-unit project under various optimization methods

Task		A	B	C	D	E	F	G
GARA-I: completed in 85 days	Work continuity imposed	yes	yes	yes	no	yes	yes	yes
	Crew size / duration (days)	9 / 9	4 / 7	4 / 5	6 / 5	1 / 7	8 / 10	4 / 4
Tardiness penalty: $0	Number of days with work discontinuity	0	0	0	8	0	0	0
	Work discontinuity penalty ($)	0	0	0	2784	0	0	0
CPMA: completed in 80 days	Work continuity imposed	no	no	no	no	no	no	no
	Crew size / duration (days)	9 / 9	4 / 7	2 / 10	6 / 5	1 / 7	8 / 10	2 / 7
Tardiness penalty: $0	Number of days with work discontinuity	0	8	0	16	8	0	12
	Work discontinuity penalty ($)	0	1920	0	5568	496	0	1032
CPMA: completed in 89 days	Work continuity imposed	yes	yes	yes	yes	yes	yes	yes
	Crew size / duration (days)	9 / 9	6 / 5	3 / 7	6 / 5	2 / 5	8 / 10	4 / 4
Tardiness penalty: $10000	Number of days with work discontinuity	0	0	0	0	0	0	0
	Work discontinuity penalty ($)	0	0	0	0	0	0	0

GSA and CPMA return poorer schedules because these two methods make particular assumptions concerning work continuity. GSA imposes work continuity on all tasks and this usually produces schedules with longer project durations. This proves to be detrimental when the project has to be completed under tight due dates. As shown in Table 5.3, the hefty penalties incurred by the GSA schedules are entirely due to not being able to meet the tight due date constraints. This also explains why the performance of GSA improves significantly when the due dates are relaxed.

CPMA assumes no work continuity; it gives schedules with the shortest project makespan as the method schedules the commencement of activities as soon as the precedence and the resource availability constraints are satisfied. However, indiscriminate early starts can lead to discontinuous workflow for the tasks. For

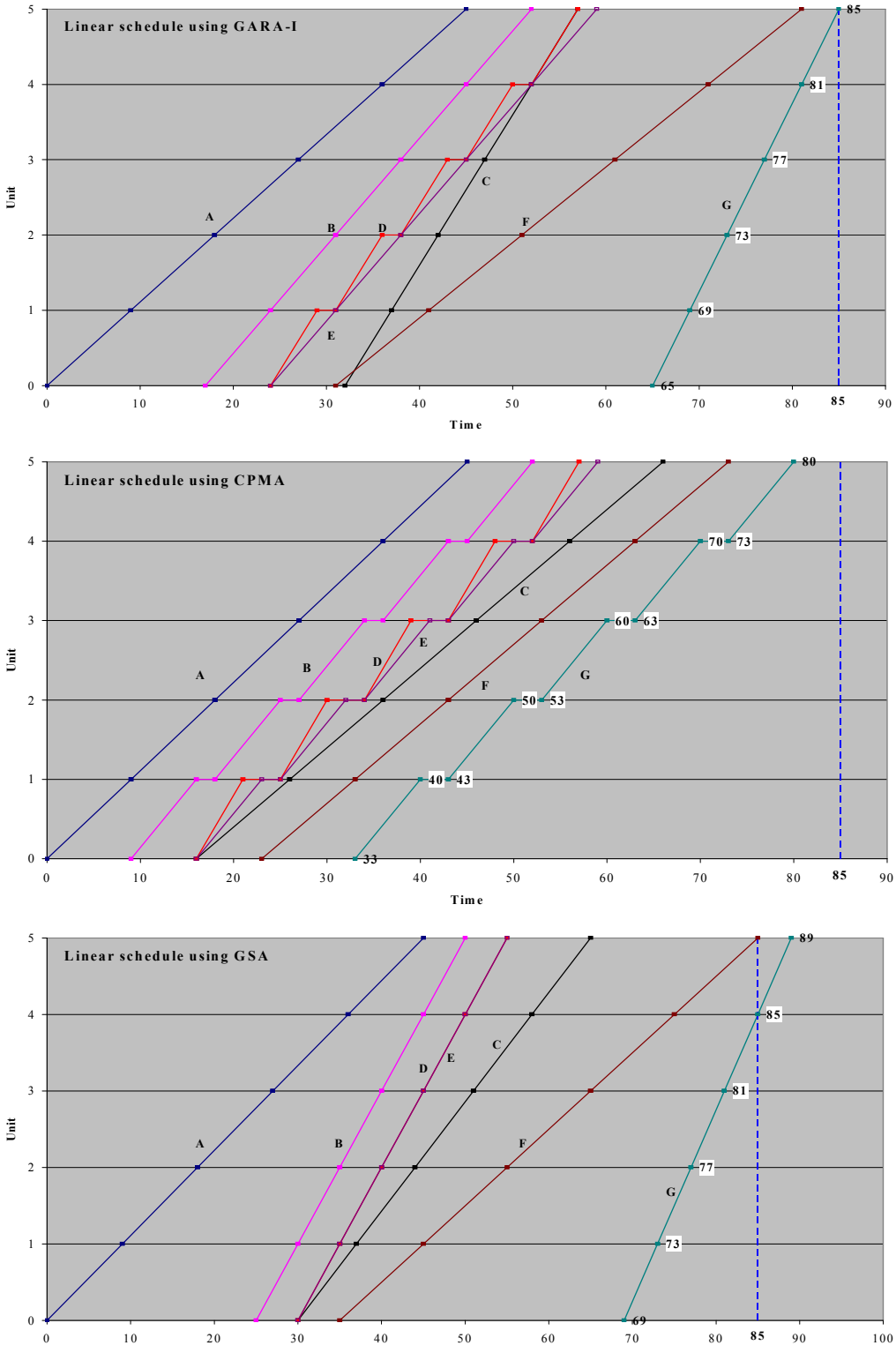


Figure 5.4 Linear schedules of results obtained under various optimization methods

example, the CPMA schedule for the 5-unit project illustrated in Fig. 5.3 shows five out of the seven tasks (tasks B, C, D, E and G) have discontinuous workflows. The large number of idle periods incurs a considerable amount of work discontinuity penalty (see Table 5.3). This effect becomes more pronounced with an increase in the number of repetitive units.

Table 5.4 presents more details of the schedules obtained for the 5-unit example under a tight due date constraint by the different approaches. Fig. 5.4 depicts the resulting linear schedules graphically. Both GARA and CPMA schedules meet the due date, and the GARA schedule has a significantly lower work discontinuity penalty (only one of the seven tasks is not work-continuous). By not imposing work continuity on task D, all the other tasks can be work-continuous and the project completes in 85 days. Even though the duration of the CPMA schedule is shorter, it is of no consequence in this example as the due date is met. On the other hand, four of the seven tasks in the CPMA schedule are not work-continuous. For example, task G is discontinuous because every activity in CPMA is scheduled to start *as soon as* all the precedence relationships are satisfied. The GSA schedule incurs a due date penalty of $10,000 as the price for imposing work continuity on all the tasks. This example also shows that it is necessary to determine both the appropriate crew size and the work continuity requirements in order to get the best schedules. GARA is able to seek the most favourable trade-off between work continuity and early project completion.

5.2.2 Effect of increasing the number of repetitive units

Fig. 5.2 also illustrates the performance of the three scheduling methods for projects with an increasing number of repetitive units under different due date constraints. In

general, GARA produces schedules that incur lower penalties than those obtained from GSA and CPMA. The performance of GARA is particularly good under tight due date constraints especially with an increasing number of repetitive units.

The previous observation can be attributed to the ability to decide whether work continuity should be imposed for any particular task under GARA. When this flexibility is used in an optimization search, balanced schedules that consider the trade-offs between the work discontinuity penalty and tardiness penalty are obtained. By contrast, CPMA emphasizes early project completions at the expense of work continuity, and since the total period of such discontinuity increases with the number of repetitive units, the schedules returned by CPMA are increasingly penalized by the work continuity penalty. On the other hand, GSA imposes work continuity on all the tasks. To achieve this, artificial delays (calculated using Eqn. 3.6) are introduced to the start of the first activity of tasks that are not work-continuous. These delays are proportionate to the number of repetitive units, thus resulting in increasingly poorer performance when the number of units increases and a higher tardiness penalty is incurred.

Therefore, it is not surprising to note that the performance of GSA improves significantly when the due date constraints are relaxed. The schedules produced using GSA are as good as those generated by GARA for projects with a lower number of repetitive units under medium and relaxed due date constraints. In the latter instance, GSA performs just as well as GARA regardless of the number of repetitive units.

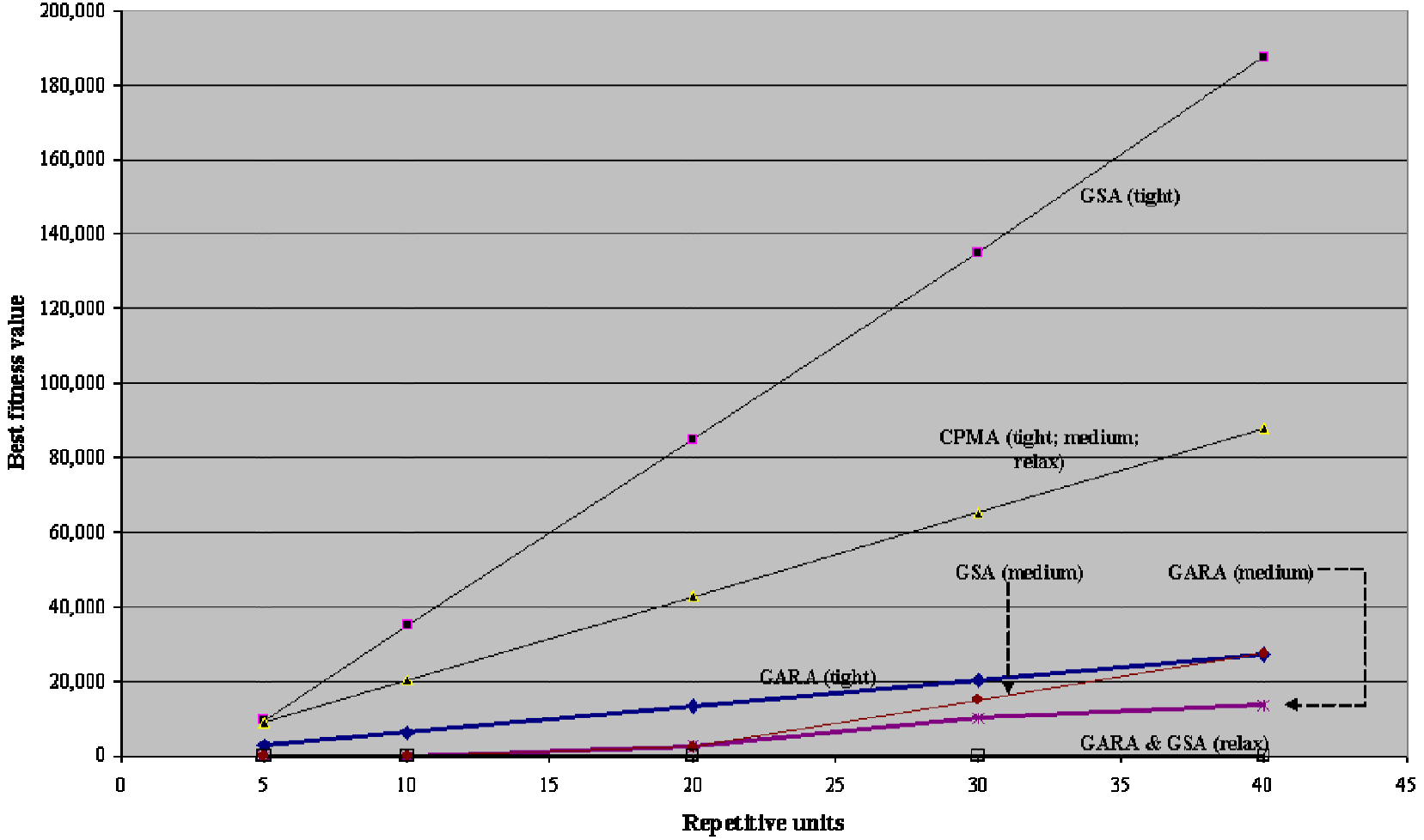


Figure 5.5 Performance of various optimization methods under different due date constraints

5.2.3 Effect of imposing different due date constraints

Fig 5.5 illustrates the performance of the various optimization methods under different due date constraints. GSA is the worst performer under tight due date constraint while CPMA fares the worst under medium and relaxed due dates. This is because CPMA produces the same schedule for a given number of repetitive units regardless of the due date constraints imposed while the results of GARA and GSA generally improve as the due dates are extended. For example, the solutions from GARA and GSA improve from $20,184 and $135,000 to $10,092 and $15,000 respectively for a 40-unit project when the due date is extended from 385 days to 433 days. This is a significant improvement in terms of percentage change as a 12.5% ([433 – 385] / 385) in project duration reduces the penalties by 50% ([10092 – 20184] / 20184) and 88.9% ([15000 – 135000] / 135000) for GARA and GSA respectively.

Fig. 5.5 also indicates that the performance of GSA improves most significantly when the due dates are extended from 120 percent (tight due date) to 135 percent (medium due date) of the shortest project durations. In fact, from Table 5.2, it is observed that the GSA performs as well as GARA under medium due dates for 5 to 20 repetitive units, and the schedules that it generated for over 20 repetitive units are only slightly worse off than the results of GARA. Under relaxed due dates (set at 150% of the shortest project duration), both GARA and GSA produce “perfect” schedules that are able to meet the due date without incurring any work discontinuity penalty for all the number of repetitive units tested.

These observations draw two practical conclusions. Firstly, the better performance of GARA and GSA over CPMA under medium and relaxed due dates indicates that

work continuity is an important factor to be considered when scheduling projects with deadlines that are not too tight. This is especially so when the number of repetitive units involved is significant, as seen by the trend in Fig 5.5 where the difference in performance between GARA or GSA and CPMA widens as the number of repetitive units increases. However, Fig. 5.5 also shows that GSA should not be used for scheduling repetitive projects that have to be completed within tight schedules, and this is especially so when the number of units is large.

Secondly, GSA and GARA perform equally well under relaxed due dates; this suggests that GSA can be used as an alternative scheduling method for projects with generous due dates. The advantage of GSA is its simplicity since it is only necessary to decide the best crew size for each task.

5.2.4 Comparing two different means of imposing work continuity

This section compares the effect of imposing different degrees of work continuity. Work continuity for a task can be achieved by delaying the earliest finish date of its first activity. Eqn 3.8 gives the minimum number of days of delay necessary to ensure that the task is fully continuous. However, it is also possible to introduce a shorter delay in which case the task will not be fully continuous. A set of experiments, involving two different chromosome representations for the work continuity decision variable, was done to investigate whether full or partial work continuity would be more advantageous in performing the optimization search. In GARA-I, the chromosome encodes for whether a full delay is to be introduced or not whilst in GARA-II the chromosome encodes the option to introduce partial delays, which is defined as any integer values in the range from zero, indicating no delay, to the value

Table 5.5 Optimization results for the 5-unit project using GARA-I and GARA-II

Task		A	B	C	D	E	F	G
GARA-I: completed in 85 days	Work continuity imposed	full	full	full	full	full	full	full
Tardiness penalty: $0	Crew size / duration (days)	9 / 9	4 / 7	4 / 5	6 / 5	1 / 7	8 / 10	4 / 4
Ave. time to convergence: 36 secs	Number of days with work discontinuity	0	0	0	8	0	0	0
	Work discontinuity penalty ($)	0	0	0	2784	0	0	0
GARA-II: completed in 85 days	Work continuity imposed	full	partial	full	partial	partial	full	full
Tardiness penalty: $0	Delay on the first activity (days)	0	7	6	3	0	0	24
Ave. time to convergence: 194 secs	Crew size / duration (days)	9 / 9	4 / 7	3 / 7	6 / 5	1 / 7	8 / 10	2 / 7
	Number of days with work discontinuity	0	1	0	6	1	0	12
	Work discontinuity penalty ($)	0	240	0	2088	62	0	0

indicating full delay. The search space for GARA-II is larger but offers the possibility of finding better solutions involving tasks with partial work continuity. By contrast, GARA-I operates in a smaller search space since only schedules that contain tasks that are either fully work continuous or not are considered.

The advantage of considering partial work continuity lies in the possibility of distributing work discontinuity among a group of tasks, so that all or most of the tasks enjoy some form of continuous workflow in the earlier activities, with work discontinuity setting in only at a later stage for some tasks. This is illustrated in the 5-units example under a tight due date constraint, where GARA-II returns a better solution compared to the schedule of GARA-I. Table 5.5 and Fig 5.6 illustrate the optimization results and linear schedules for these two approaches. GARA-II returned

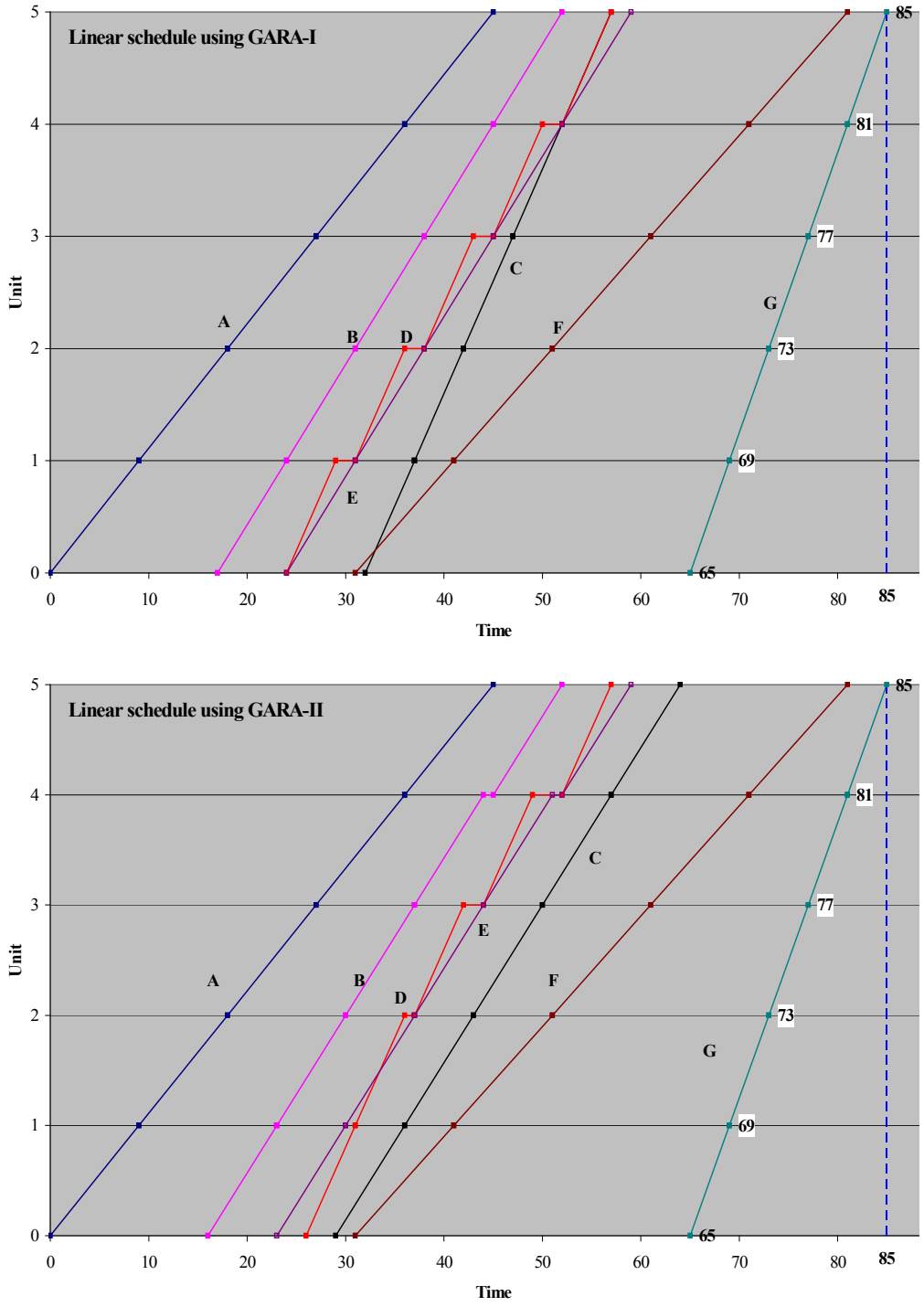


Figure 5.6 Linear schedules for results obtained using GARA-I and GARA-II

the better schedule by distributing 8 days of idle period between tasks B, D and E such that the work discontinuity penalty is reduced from $2,784 to $2,390, or a decrease of approximately 14%.

The drawback is that GARA-II takes longer to execute because of the larger search space. This can be an issue when the size of the problem increases. For larger problems involving ten repetitive units or more, GARA-II generally returns solutions that are inferior to those of GARA-I even when the termination criterion is extended to 1000 iterations or approximately 60 minutes of GA run-time. GARA-II is able to match the performance of GARA-I only in those cases under relaxed due date constraints. The poor performance of GARA-II is attributed to the rapidly increasing search space as the repetitive units increase. An estimate of the search space in GARA-II is presented in Appendix A1.

In GARA-I, assuming full work continuity for tasks significantly reduces the search space. More importantly, the size of the search space no longer depends on the number of repetitive units but only on the number of tasks in the schedule. GARA-I is the preferred scheduling approach in this study because it is able to determine quickly very good schedules under a wide variety of circumstances.

5.2.5 Time of convergence

Due to the concerns on computation time raised by the experience with GARA-II, another set of experiments was conducted to investigate the time of convergence characteristic of GARA-I. It was not known if the complexity of the interrelationships between the project tasks would contribute to increasing the difficulty of the optimization search. The task interrelationships form a precedence network and more precedence constraints between tasks could conceivable lead to greater difficulty in scheduling. The experiments recorded the time of convergence using GARA-I on projects of different network complexity and different number of repetitive units.

Among the different measures of network complexity are Coefficient of Network Complexity (Kaimann, 1974) and Restrictiveness Estimator (Thesen, 1977). This study has chosen the *Restrictiveness Estimator* (RT) because RT relates the complexity of a network to the number of feasible sequences that exist in the network. The restrictiveness estimator can be calculated by:

$$RT = \frac{2\sum r_{ij} - 6(M-1)}{(M-2)(M-3)} \tag{5.1}$$

where r_{ij} are the elements of the reachability matrix discussed in Appendix B2, and M is the total number of tasks in the network. The value of *RT* lies between zero and one, where zero represents parallel directed graphs (digraphs) and one denotes series digraphs.

From Eqn. 5.1, it is observed that *RT* can be varied by either changing the number of tasks in the network, or by changing the precedence relationships between the tasks. Fig. 5.7 illustrates three 7-task networks with RE ranging from 0.4 to 0.8 obtained by changing some of the relationships between the tasks. For every given number of repetitive units in the network, the benchmark solution is assumed to be the best solution obtained from ten separate experimental runs of GARA-I with randomly generated sets of the initial population over 500 iterations. The benchmark solutions are obtained for fifteen cases of three 7-task networks over five different numbers of repetitive units each. Subsequently, for each case, ten separate runs are conducted and the time taken for the solution to converge to the benchmark solution in each run is noted. The total convergence time taken is the sum of the ten individual times of convergence, and the average time of convergence for each case is given by

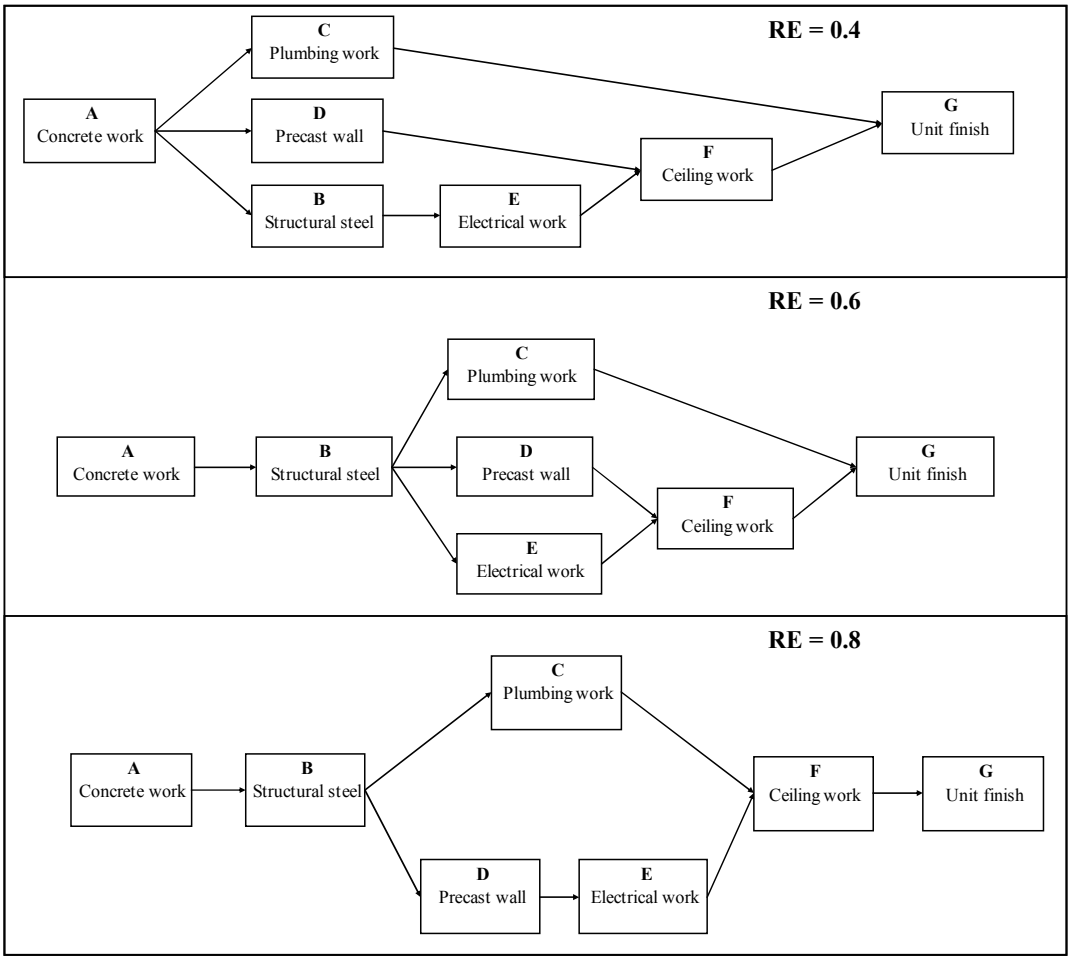


Figure 5.7 Illustrations of three 7-task networks with different restrictiveness estimators

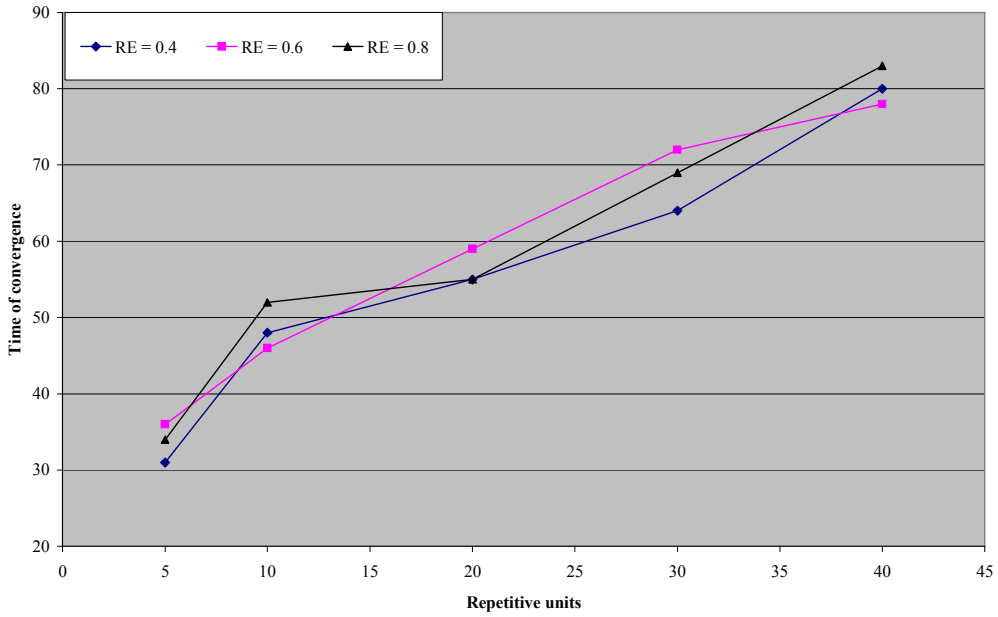


Figure 5.8 Average time of convergence for 7-task networks with different restrictiveness estimators

normalizing the total convergence time with the number of runs. Fig. 5.8 depicts the average convergence time calculated.

From Fig. 5.8, it is observed that the time of convergence increases as the number of repetitive units increase. A likely explanation for the longer time of convergence observed is the longer computation time required to evaluate the recurrence equations since the number of variables increases with increasing repetitive units. In addition, it is observed that the network complexity does not seem to have any significant effect on the time of convergence. One possible reason for this observation is that the precedence constraints between the tasks are incorporated into the recurrence equations and do not appear in the search space defined by the chromosome.

The experiment was repeated on different networks comprising of eight tasks as illustrated in Fig. 5.9. The results of the experiment as shown in Fig. 5.10 provides further confirmation of that network complexity does not influence time to convergence because of the way the optimization search is conducted. However, the convergence times for the 8-task problems are approximately double those of the 7-tasks projects. The most likely explanation for this is that the addition of new tasks increases the number of possible solutions to the problem and affects the time taken for GA to converge to the benchmark solutions.

5.2.6 Minimization of idle periods for CPM schedules

Analysis of the results obtained from CPMA schedules reveals some interesting observations with regard to the objective of minimizing the idle periods in the schedule. Firstly, it is observed that the schedules obtained by CPMA under different

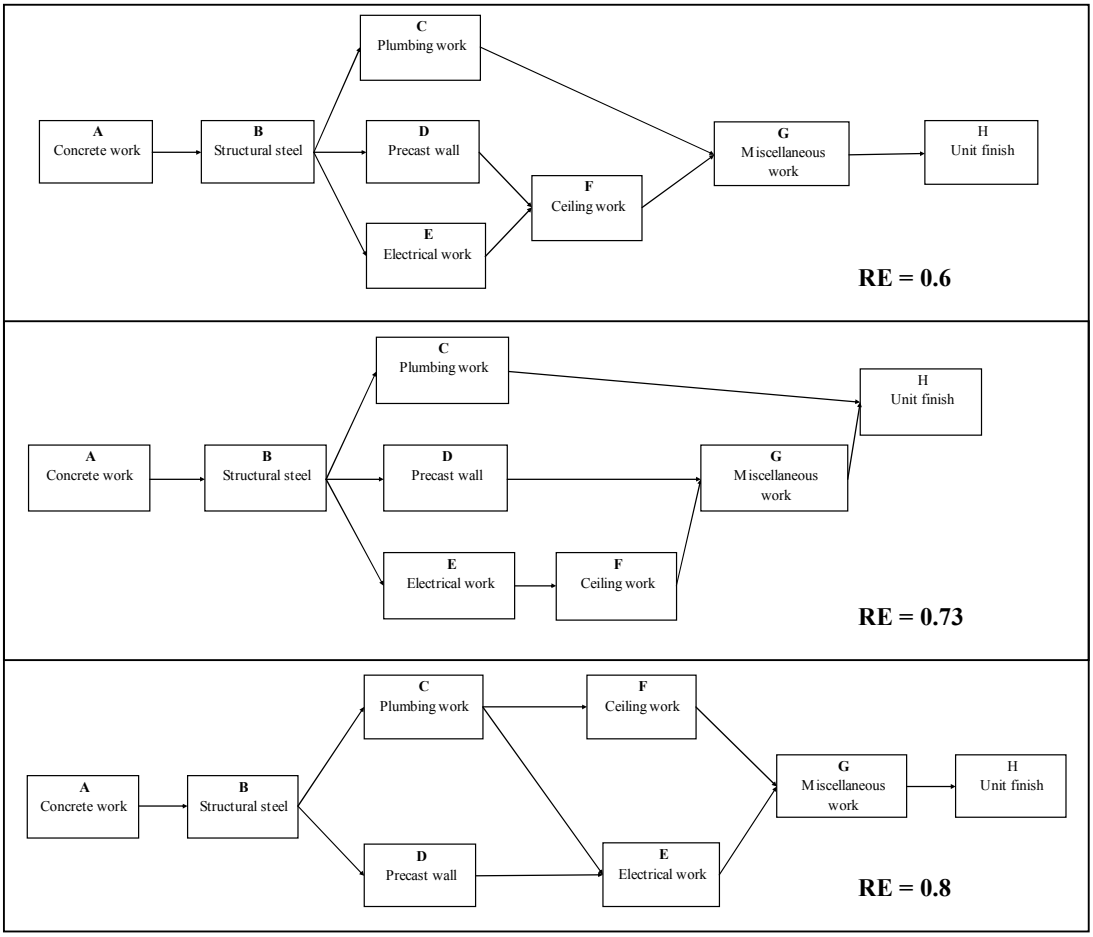


Figure 5.9 Illustrations of three 8-task networks with different restrictiveness estimators

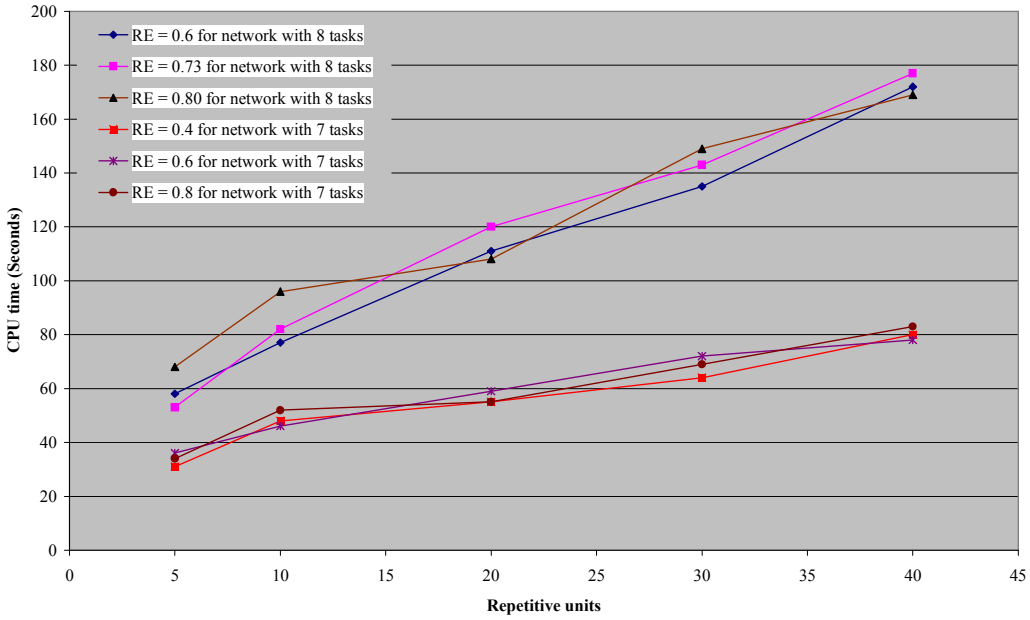


Figure 5.10 Average time of convergence for networks with different restrictiveness estimators

due date constraints are identical for each given number of repetitive units. Furthermore, the crew size compositions obtained from CPMA are identical across all the cases of varying due date constraints and repetitive units for the given network. In other words, the crew size composition is identical for a 5-unit or 40-unit project for example. These two observations, taken together, suggest that for a given network there exists an optimal composition of crew sizes that minimizes the total idle period regardless of the due date constraints and repetitive units. The deterministic early start heuristic used within CPMA does not seem to be responsive to different due date constraints and the number of repetitive units.

The existence of this optimal crew size composition can be explained in relation to Fig. 3.3, which illustrates that the calculation of idle days is dependent on the effective durations and durations of the tasks. In addition, since there are Q-1 finish-to-start intervals for every task that is repeatedly executed over Q units, the total period of work discontinuity for a task *i* is equals to *(Q-1)*(Max[0, D_i -T_i])*. This mathematical expression establishes that the period of work discontinuity in a schedule depends only on the task durations and the precedence relationships between different tasks from the calculations of effective durations using Eqn. 3.5. Therefore, the minimum length of idle periods for a network with any number of repetitive units is given by the same set of task durations.

Consequently, an optimization function for minimizing the idle periods in schedules derived from CPM calculations can be developed:

$$MIN \quad Z_{CPM} = \sum_{i=1}^{M} MAX[0,(D_i - T_i)] \tag{5.2}$$

The development of this optimization function has two purposes. Firstly, it defines the variables that affect the length of idle periods that exists in schedules derived using the CPM calculation. In doing so, it illustrates how the work rates of preceding tasks affect the state of work continuity of subsequent tasks, and where work discontinuity exists, it quantifies the length of the idle periods. Therefore, project planners who are using CPM equations to schedule repetitive projects can still minimize the total idle periods in the schedules by making an informed selection of the appropriate set of task durations.

Secondly, the optimization function requires only minimal information of the precedence constraints between tasks and the set of possible durations for each task. If the number of possibilities is not great, a simple iterative process can be set up using an electronic spreadsheet to determine the optimal composition of task durations. An example of the iterative approach for minimizing idle periods in CPM schedules is illustrated in Appendix C.

However, the optimization function gives equal weightage to each idle day regardless of the number of workers involved and it cannot distinguish between idle periods that involve one worker or ten workers. Notwithstanding this shortcoming, the iterative process is a simple and quick way to optimize the schedules of repetitive projects, especially when the early-start CPM heuristic is adopted for scheduling.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

This chapter first summarizes the significant findings and observations in this thesis. This is followed by a discussion of the limitations of the study, before recommendations are made for future research.

6.1 Conclusions

The repetitive nature of multiple-units construction projects necessitates the creation of schedules with continuous and uninterrupted workflows for the tasks involved. However, maintaining work continuity for some tasks can lengthen the project duration, leading to a possible conflict of interest between the sub-contractor and the main contractor. This thesis proposed an efficient way to schedule repetitive projects by introducing a set of recurrence equations that considers work continuity. The proposed equations retain the analytical capability to calculate important scheduling parameters like total floats and are flexible enough to incorporate specific user-specified scheduling constraints. A case study illustrated the working of this set of equations in scheduling repetitive projects, and the schedule information calculated was represented in various graphical forms to convey the information to the end-user effectively.

The recurrence equations form the basis of Genetic Algorithms Recurrence-equations Approach (GARA) for optimizing schedules of repetitive projects. The schedules are optimized with respect to tardiness (delay in meeting project deadline) as well as the degree of work discontinuity. Work continuity and crew size both influence the

scheduling of repetitive projects, and are incorporated into the recurrence equations. The GA was used to search for the best settings for these two kinds of decision variables. The results obtained using GARA were compared against those obtained using two other scheduling methods. The following sections present summarized analyses of the experimental results obtained.

6.1.1 Increasing the number of repetitive units

Repetitive projects are characterized by the existence of numerous units, and the performance of GARA is evaluated across increasing number of repetitive units. In terms of computational results, the schedules produced by GARA were at least as good as the schedules obtained from GSA and CPMA for all the cases considered. More significantly, GARA was able to produce schedules that incur lower penalties than those obtained from GSA and CPMA under tight and medium due date constraints and the performance of GARA improved with an increasing number of repetitive units. Finally, under relaxed due date constraints, GARA was able to return schedules that ensure punctual project completions while at the same time maintain full work continuity for every construction task involved.

The inferior results from CPMA can be attributed to the huge work discontinuity penalty that it incurred. As the number of repetitive units multiples, the total length of idle periods for schedules under CPM invariably increases and CPMA is penalized for its inability to maintain work continuity. On the other hand, GSA did badly because it sacrifices timely project completions for work continuity. This is especially detrimental when the projects have tight schedules for completion. Therefore, GSA is

heavily penalized in terms of the tardiness penalty, causing it to perform poorly in comparison to GARA.

The increasing disparity between the results obtained by GARA and the control cases illustrates that it is insufficient to focus on either work continuity or punctual projection completion alone, especially when multiple repetitive units are involved. In essence, it emphasizes the importance of work continuity as a scheduling consideration in the management of construction projects involving increasing number of repetitive units.

6.1.2 Imposing different due date constraints

GARA was able to return schedules with the lowest penalties under all three different types of due date constraints. In fact, the performance of GARA was most outstanding under tight schedules with increasing number of repetitive units when compared to the solutions from the control cases. Furthermore, GARA was able to generate improved solution as the due date constraints were relaxed.

GARA's superior performance, especially under tight due date constraints, can be attributed to its ability to make meaningful trade-offs between work continuity and project durations. By appropriately delaying the start date of the first activities, GARA was able to seek out the best composition of tasks with which work continuity can be imposed for a given completion deadline to minimize the work discontinuity penalty without jeopardizing project tardiness.

6.1.3 Different means of imposing work continuity

Although GARA-II is able to consider the possibility of partial work continuity, it returned the best solution in only one instance when compared to GARA-I. Nonetheless, analysis of this single case wherein GARA-II performed better illustrated two potential advantages for its use. Firstly, by considering partial work continuity, the idle periods were re-distributed among several tasks, resulting in a lower work discontinuity penalty. Secondly, the re-distribution of idle periods among several tasks so that all or most trade specialists can enjoy some degree of work continuity might be more favorable than to have full work discontinuity on a few tasks.

However, GARA-II was consistently outperformed by GARA-I despite its potential advantages. This can be explained by the increase in number of possible solutions as the number of repetitive units or tasks increase, and this rapidly expanding search space restricted the efficiency for GA to seek the best solutions. In contrast, the search space for GARA-I is dependent only on the number of tasks, and this translates to a smaller search space in which convergence can be more rapid.

6.1.4 Time of convergence

The performance of GARA-I is further investigated in terms of its time of convergence for networks with varying complexity quantified by the restrictiveness estimator (RT). The results indicted that there was no significant different in the time of convergence for networks with the same number of tasks but different RT values over various number of repetitive units. Instead, the time to convergence increases only when an additional task was added to the network. This increase can be attributed to an increased in the number of possible solutions, which resulted in longer research times

for the best solutions. In addition, it was observed that there was a minimal increase in the convergence time as the number of units increased. This can be explained by the longer computation time required to evaluate the recurrence equations used to calculate the start / finish times for each activity.

6.2 Limitations of the Study

There are several limitations in the proposed model. Firstly, the recurrence equations assumed that the set-up time for an activity is negligible. However, some tasks in real-life involve extensive machinery setup and the time incurred due to this operation can be significant thereby affecting the computation of the appropriate delay on the first activities in order to ensure work continuity. It is likely that the computed project duration will be underestimated when these set-up times are not taken into account.

Secondly, the penalty defined in the objective function is assumed to be the same for each day of work discontinuity regardless of where the idle day occurs. For example, the objective function assumes that a given period of idle days that occurs between the first and second activities incurs the same penalty as where the period of idle days occurs between the 30^{th} and 31^{st} activities. However, this assumption may not accurately reflect the costs incurred by the sub-contractor due to work discontinuity in real-life, particularly in construction projects for multi-story buildings. For instance, the work discontinuity costs for tasks like slab casting in multi-story building construction projects may increase for higher stories due to the additional costs incurred for reestablishing the raw material supplies to the higher levels. Therefore, it would be beneficial to investigate the nature of the major tasks in a repetitive project,

and apply an appropriate distribution of work discontinuity costs for each activity of every task.

Finally, the model has not been extensively tested on real-life projects with specific scheduling constraints and planning idiosyncrasies. Therefore, the usefulness of GARA for scheduling repetitive projects in real-life industry context cannot be readily assessed.

6.3 Recommendations for Future Research

The following issues are identified as directions for possible future research to improve the proposed recurrence equations and optimization approach for schedules of repetitive projects:

1. Improvements on the recurrence equations

The set of recurrence equations can be improved by incorporating the relevant set-up time for every activity. This can be done by introducing a new term in the equations to account for this additional amount of time required to recommence an activity. In addition, the recurrence equations can be enriched to deal with schedules that require various unique requirements. In doing so, the effectiveness of this set of proposed equations will be increased, and end-users need not modify the equations in order to enjoy its functionalities.

2. Resource availability

GARA assumes that the limiting resource is always the crew. However, it is possible that resources like tools, machineries, and even workspace are the limiting factor in

real life. Therefore, a resource model can be developed to account for the availability of various resources required for any particular task. This resource model should ideally be able to "communicate" with GARA in the optimization process in order to produce schedules that are more comprehensive.

3. Preferred workflow

GARA assumes work continuity is equally preferred whether at earlier or later activities of a task. However, specific trades could associate different costs for work discontinuity at various stages of their work. Therefore, in order to generate schedules that are more realistic to life-real demands, the preferred workflow for every major trade should be investigated. The objective function can then be modified accordingly to reflect the suitable penalties for discontinuous workflow at various activities for each task.

4. Genetic Algorithms Parameters Setting

The need to define the optimal GA parameters for the best solution sets cannot be compromised. The crossover and selection operators used, the population size, the terminating criterion and the crossover and mutation rates will require further investigation to be optimized.

5. User Interface

Lastly, a friendly user interface will be beneficial for the application of the proposed model in the industry. Naturally, the interface should incorporate the user-input interface for specific schedule requirements. The output interface can present the schedules in generic Gantt charts and linear schedules, or other predefined

representations like matrix schedules. One might even indulge the managers by presenting a number of different alternative schedules, each optimized for different evaluation parameters.

REFERENCES

Ashley, D. B. (1980). Simulation of repetitive-unit construction. *Journal of Construction Engineering and Management*, ASCE, 112(3), pp. 411-424.

Barrie, D. S., and Paulson, B. C. J. (1992). *Professional Construction Management*, 3rd Ed., McGraw-Hill, New York.

Chan, W. T., Chua, D. K. H., and Kannan, G. (1996). Construction resource scheduling with genetic algorithms. *Journal of Construction Engineering and Management*, ASCE, 122(2), pp. 125-132.

Chan, W. T., and Hu, H. (2002) Production scheduling for precast plants using a flow shop sequencing model. *Journal of Computing in Civil Engineering*, ASCE, 16(3), pp. 165-174.

Chrzanowski, E. N., and Johnston, D. W. (1986). Application of linear scheduling. *Journal of the Construction Division*, ASCE, 112(4), pp. 476-491.

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

Gen, M., and Cheng, R. W. (1997). *Genetic Algorithms and Engineering Design*, Wiley, New York.

Harmelink, D. J., and Rowings, J. E. (1998). Linear scheduling model: Development of controlling activity path. *Journal of the Construction Division*, ASCE, 124(4), pp. 263-268.

Harris, R. B., and Ioannou, P. G. (1998). Scheduling projects with repeating activities. *Journal of Construction Engineering and Management*, ASCE, 124(4), pp. 269-278.

Hegazy, T. (1999). Optimization of resource allocation and levelling using genetic algorithms. *Journal of Construction Engineering and Management*, ASCE, 125(3), pp. 167-175.

Hegazy, T., and Wassef, N. (2001). Cost optimization in projects with repetitive nonserial activities. *Journal of Construction Engineering and Management*, ASCE, 127(3), pp. 183-191.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Anbor, Michigan.

Johnston, D. W. (1981). Linear scheduling method for highway construction. *Journal of the Construction Division*, ASCE, 107(2), pp. 247-261.

Kaimann, R. A. (1974). Coefficient of network complexity. *Management Science*, 21, pp. 172-177.

Kallantzis, A., and Lambropoulos, S. (2004) Discussion of "Comparison of linear scheduling model and repetitive scheduling method". *Journal of the Construction Division*, ASCE, 130(3), pp. 463-467.

Kang, L. S., Park, I. C., and Lee, B. H. (2001). Optimal schedule planning for multiple, repetitive construction process. *Journal of Construction Engineering and Management*, ASCE, 127(5), pp. 382-390.

Leu, S.-S., and Yang, C.-H. (1999) GA-based multicriteria optimal model for construction scheduling. *Journal of Construction Engineering and Management*, ASCE, 125(6), pp. 420-427.

Li, H., Cao, J.-N., Love, P. E. D. (1999) Using machine learning and GA to solve time-cost trade-off problems. *Journal of Construction Engineering and Management*, ASCE, 125(5), pp. 347-353.

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

Lumsden, P. (1968). *The Line of Balance Method*, Pergamon Press Ltd., Industrial Training Division, London.

Mattila, K. G., and Abraham, D. M. (1998) Resource levelling of linear schedules using integer linear programming. *Journal of Construction Engineering and Management*, ASCE, 124(3), pp. 232-244.

Mattila, K. G., and Park, A. (2003). Comparison of linear scheduling model and repetitive scheduling method. *Journal of the Construction Division*, ASCE, 129(1), pp. 56-64.

Moselhi, O., and El-Rayes, K. (1993). Scheduling of repetitive projects with cost optimization. *Journal of Construction Engineering and Management*, ASCE, 119(4), pp. 681-697 .

Moselhi, O., and Hassanein, A. (2003). Optimized scheduling of linear projects. *Journal of Construction Engineering and Management*, ASCE, 129(1), pp. 664-673.

Neale, R. H., and Neale, D. E. (1989). *Construction Planning*, 1st Ed., Thomas Telford Ltd., London, England.

O'Brien, J. J. (1975). VPM scheduling for high-rise buildings. *Journal of the Construction Division*, ASCE, 101(4), pp. 895-905.

O'Brien, W. J., and Fischer, M. A. (2000). Importance of capacity constraints to construction cost and schedule. *Journal of Construction Engineering and Management*, ASCE, 126(5), pp. 366-373.

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

Selinger, S. (1980). Construction planning for linear projects. *Journal of the Construction Division*, ASCE, 106(2), pp. 195-205.

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*, ASCE, 120(3), pp. 667-684.

Thesen, A. (1977). Measures of the restrictiveness of project networks. *Networks*, 7, pp. 193-208.

Yamin, R. A., and Harmelink, D. J. (2001). Comparison of linear scheduling model and critical path method. *Journal of Construction Engineering and Management*, ASCE, 127(5), pp. 374-381.

Zheng, D. X. M., Ng, S. T., and Kumaraswamy, M. M. (2004) Applying a genetic algorithm-based multiobjective approach for time-cost optimization. *Journal of Construction Engineering and Management*, ASCE, 130(2), pp. 168-176.

APPENDIX A

Discussion on the search space of GARA-II

Table A1 Sample calculation for the number of possible schedules considered under GARA-II

Task	**A**	**B**	**C**
(Q - 1)Max[0, (D_i – T_i)]	0	4*2 = 8	4*3 =12
Number of terms in the range (0, 1, 2, …, *(Q - 1)Max[0, (D_i – T_i)]*)	1	9	13
Number of possible solutions under GARA-II	1*9*13 = 117		

In GARA-II, the decision variable of work continuity is represented by an integer value with which the earliest completion of the first activity of a task is delayed and this integer value is bounded by the range of (0, 1, 2, …, *(Q - 1)Max[0, (D_i – T_i)]*). In order to illustrate the computation of the number of possible solutions, assume that a project consists of three tasks A, B and C with durations of 10, 8 and 5 days respectively to be executed over five units, where task A precedes task B, and task B in turn precedes task C. The calculation for the total number of possible schedules under GARA-II is shown in Table A1.

Analogous to the above illustration, the number of possible solutions using the scheduling parameters in the experiment for GARA-II is:

Total number of possible solutions

$$= 3^7 \times \prod_{i=2}^{7} (Q-1) \times \{MAX[0,(D_i - T_i)] + 1\} \qquad \text{(A.1)}$$

$$= 3^7 \times (Q-1)^6 \times \prod_{i=2}^{7} \{MAX[0,(D_i - T_i)] + 1\}$$

The term of 3^7 accounts for the three choices of crew size for each task. Eqn. A.1 illustrates that the search space for GARA-II is related to the number of repetitive units by the term $(Q-1)^6$. Fig. A1 illustrates the rapid increase in the value of the $(Q-1)^6$ term with increasing number of repetitive units, which in turn suggests show that the search

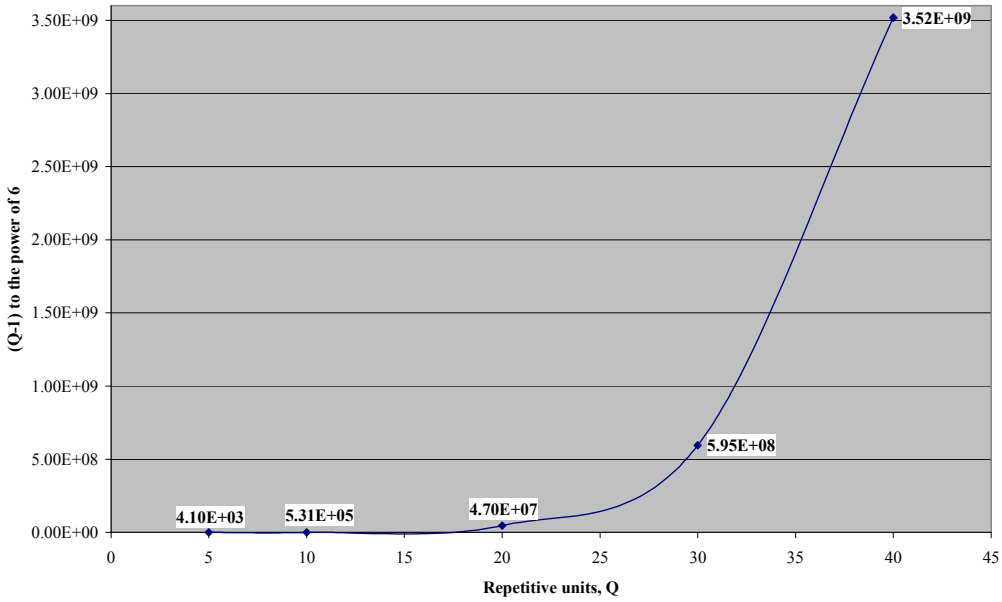


Figure A1 Variations of the (Q-1) term with respect to the number of repetitive units

space for GARA-II also increases rapidly. Therefore, the poor performance of GARA-II and the significantly higher amount of time that it needs to search for the best solutions can be explained by its comparatively large and rapidly increasing search space.

APPENDIX B

Discussion on the reachability matrix

The computation of RT requires the construction of a reachability matrix R = [r_{ij}], such that r_{ij} is equals to one if there is a path from node *i* to node *j*; otherwise, r_{ij} is equals to zero. For example, the reachability matrix for one unit of the network used in the experiment (Fig. 1.1) is:

$$R = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \end{matrix}$$

To illustrate the workings of R, consider row 3, which represents the available paths for task C. Since there cannot exist a path from task C to task A and B, r_{31} and r_{32} are both equal to zero. Similarly, no path exists between task C and task D / task E, and accordingly, r_{34} and r_{35} are zeros. Finally, r_{37} reflects the fact that a path exists from task C to task G.

APPENDIX C

Illustration of the iterative approach for minimizing idle periods in CPM schedules

Activities, i	Preceding activities	Possible durations			Effective duration	MAX[0, D(i) - T(i)]	Duration, T(i)
A	nil	9	10	13	0	0	13
B	A	5	6	7	13	6	7
C	B	5	7	10	13	3	10
D	B	2	3	5	13	8	5
E	B	3	5	7	13	6	7
F	D, E	10	12	15	13	0	15
G	C, F	4	5	7	15	8	7
						31	
		Iteration 1			**Effective duration**	**MAX[0, D(i) - T(i)]**	**Duration, T(i)**
					0	0	**9**
					9	2	7
					9	0	10
					9	4	5
					9	2	7
					9	0	15
					15	8	7
						16	
		Iteration 2			**Effective duration**	**MAX[0, D(i) - T(i)]**	**Duration, T(i)**
					0	0	9
					9	2	7
					9	0	10
					9	4	5
					9	2	7
					9	0	**10**
					10	3	7
						11	
		Total period of discontinuity =				**11 (Q-1)**	

Figure C1 Iterative method to determine the optimal values of T_i

Fig. C1 illustrates the workings of this iterative process under spreadsheet modeling. Using the scheduling data in Table 5.1, the optimal set of task durations for which the total idle period in the CPM schedule is minimized by first creating two new columns of calculations: (1) the effective duration for each task calculated using Eqn. 3.3, and (2) the value of the function $MAX[0,(D_i - T_i)]$. Assuming that the task durations are their maximum values initially, the starting value of the optimization function is 36 days. From the calculations of the effective duration, it is noted that the effective duration of task B, D_B, is equals to the duration of A, T_A. Therefore, in the first iteration, T_A is reduced to 9 days so that the value of $D_B - T_B$ is minimized. Corresponding, the difference between the effective duration and the duration of tasks, B, C, and D are also minimized. In the second iteration, the duration for task F is reduced in order to reduce the effective duration of task G in the second iteration. The two iterations reduce the optimization function to 11 days, which corresponds to the

value found in CPMA. Subsequently, the idle period in the schedule for any given number of repetitive units (Q) can be computed by $(Q\text{-}1) \times Z_{CPM}$.