FINANCE-BASED SCHEDULING OF ACTIVITY NETWORKS

BY

ANAS A. AL-GHAZI

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SYSTEMS ENGINEERING

June 2009

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **ANAS ALSAYED MOHAMMED ALGHAZI** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of his requirements for the degree of **MASTER OF SCIENCE IN SYSTEMS ENGINEERING**.

Thesis Committee

S. Selim
Dr. Shokri Z. Selim (Thesis Advisor)

Ashraf Elazouni
Dr. Ashraf M. El-Azouni (Member)

Dr. Umar M. Al-Turki (Member)

Dr. Fouad M. Al-Sunni
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

3/2/10
Date

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

I would like to dedicate this work to my parents, my wife

and my kids AlBatool & Omar

ACKNOWLEDGMENTS

In the name of Allah, the most beneficent, the most merciful

All praise goes to Allah (SWT), the Almighty, who granted me the strength and patience to complete this work. Peace and blessings of Allah be upon his last prophet Muhammad (PBUH), his family and his companions.

My deep appreciation and gratitude goes to my thesis advisor Prof. Shokri Selim for his guidance, consistent help and continuous support throughout my thesis work. His valuable suggestion and useful discussions made this work interesting to me. I am also very grateful for my thesis committee member Dr. Ashraf El-Azouni for his invaluable help, advice and constructive comments. I extend gratitude to my thesis committee member Dr. Umar Al-Turki for his interest, cooperation and insightful feedback. Acknowledgment is also due to Prof. Fouad Al-Sunni, chairman of the Systems Engineering Department.

My family has always been a pillar of support for me. I would like to thank my brothers and sisters for their love, support, and encouragement. Thanks to my parents who were - after Allah- the source of success in my life. I was carried through the most difficult moments in my life by their prayers, love, and support. Words fall short in conveying my gratitude towards them. A prayer is the simplest means I can repay them - May Allah

(S.W.T.) give them good health and give me ample opportunity serve them throughout my life.

My heartfelt appreciation and gratitude goes to my friend and wife, Afaf Effat, the one who has been sharing with me the difficult moments before the nice ones, the one whose support, keen understanding, patience and love enabled me to complete this work.

I am also thankful to the faculty and staff members of the Systems Engineering department for their kind support and cooperation. The support provided by King Fahd University of Petroleum and Minerals for completing this work is highly appreciated.

Last but not least, thanks are due to all of my friends and colleagues for their moral support, good wishes and the memorable days shared together. These include Khaled Al-Shareef, Syed Mujahid, Ahmad Bahjat, Muhammed Al-Durgam and Laith Al-Hadidi.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

Name: Anas AlSayed Mohammed AlGhazi

Title: Finance-Based Scheduling of Activity Networks

Major Field: Industrial and Systems Engineering

Date of Degree: June 2009

Contractors usually secure funds from banks by establishing credit-line accounts to finance all ongoing projects. Due to the nature of the common contracts with different clients, contractors often operate under cash-constrained conditions. Thus, contractors need to have operation planning that follows along with the project's financial planning. That is, to develop project schedules based on cash availability. Unfortunately, this integration between financing and scheduling tasks is rare in the literature and is missing in commercial scheduling software. Finance-based scheduling techniques integrate scheduling with financial planning by incorporating financing costs in scheduling activities under cash constraints.

In this thesis, a modified cash flow model is incorporated in the mathematical formulation of the finance based scheduling problem. This problem is then formulated as an integer program. Due to the NP-hardness nature of the problem, the exact solution fails to reach the optimal result in a reasonable time for large sized problems. Thus, we have implemented three meta-heuristics to solve this problem. A representation scheme was proposed along with a repair algorithm that guarantees the feasibility of all solutions

with respect to the precedence and financial constraints. In addition, the meta-heuristics were modified and applied to the multi projects finance-based scheduling problem. The application of this technique is illustrated by case studies solved using a program coded in Matlab. Finally, a study was made to compare the performance among the meta-heuristics based on a number of performance measures. We conclude this study by discussing the results obtained and propose some future research directions.

ملخص الرسالة

الاســـــــــم: أنس السيد محمد الغازي

عنوان الرسالة: الجدولة الزمنية المبنية على التمويل لشبكات الأنشطة

التخصـــــص: هندسة صناعية و نظم

تاريخ التخرج : جمادى الآخرة 1430

يلجأ المقاولون غالبا الى البنوك للحصول على التمويل اللازم لتنفيذ المشاريع الجارية و ذلك بفتح حسابات تمويلية ذات حد ائتماني معين. في العادة،يضطر معظم المقاولين للعمل تحت ظروف مالية مقيدة بسبب طبيعة العقود الموقعة مع مختلف العملاء. لهذا السبب، يحتاج المقاولون لاتباع خطة تشغيلية معينة تتناسب مع التخطيط المالي للمشاريع. ألا وهي وضع الجداول الزمنية للمشاريع بناء على الموارد المالية المتاحة. للأسف هذا التكامل بين الجدولة الزمنية و التمويل نادر في الأبحاث و مفقود في برامج الجدولة الزمنية التجارية. الجدولة الزمنية المبنية على التمويل تدمج بين التخطيط المالي و الجدولة الزمنية للمشاريع معا بأخذ تكاليف التمويل في الاعتبار عند جدولة الانشطة في ظل وجود قيود مالية.

في هذه الاطروحة يستخدم نموذج معدل للتدفق المالي في الصياغة الرياضية لمشكلة الجدولة الزمنية المبنية على التمويل. يتم بعد ذلك صياغة المشكلة كبرنامج صحيح. نظرا لدرجة تعقيد هذه المسألة حسابيا، فان البرنامج يجد صعوبة في الوصول للحل الامثل في فترة زمنية معقولة للمسائل الكبيرة. و بالتالي، قمنا بتصميم ثلاث طرق اجتهادية لحل هذه المشكلة حيث تم اقتراح نظام لتمثيل الحل في هذه الطرق بالاضافة الى خوارزمية اصلاح لضمان قابلية تنفيذ الجداول الزمنية الناتجة سواء من الناحية المالية او من ناحية ترتيب الانشطة. بالاضافة الى ذلك، تم تعديل الطرق الاجتهادية ليتم تطبيقها على مشكلة الجدولة الزمنية المبنية على التمويل لأكثر من مشروع. لتوضيح طريقة الجدولة المقترحة، تم تطبيق الطريقة على حالات دراسية و ذلك ببرنامج حاسوبي تم صياغته باستخدام Matlab. و أخيرا، تم مقارنة أداء الطرق الاجتهادية المختلفة على اساس عدد من المقاييس المقترحة. نختتم هذه الرسالة بمناقشة للنتائج التي تم الحصول عليها و اقتراح بعض الاتجاهات البحثية التي يمكن الخوض فيها مستقبلا.

NOMENCLATURE

A_i	set of activities being executed at day i
ASU	Additional shift units
B_i	Cash balance at day i
c_p	Direct cost disbursement of activity p
d_p	Duration of activity p
DI_i	Incurred daily interest for day i
E_i	Daily cost disbursement of day i
$ELST_p$	Extended late start of activity p
EST_p	Early start of activity p
ET_t	Total-cost disbursement of period t
$ExpVal(i,t)$	Expected value of individual i at time t in GA
Ext	Extra duration added for the integer program
F	Population size in SFLA
$f(i)$	Fitness of i in GA
$\bar{f}(t)$	Mean fitness of the population at time t in GA
G	Profit
G_{total}	Total profit of multi projects
I_t	Total financing cost for period t
$iter$	Number of iterations in SFLA
k	Boltzmann constant in SA
k_1	Overhead cost factor
k_2	Mark-up and retainage factor
L	Number of periods required to complete the project
LST_p	Late start of activity p
M	Sufficiently large number

m	Number of working days per period
N_t	Outstanding debt of period t
Ne	Number of memetic evolution steps in SFLA
NI_t	Interest on the outstanding debt of period t
Nm	Number of memeplexes in SFLA
Nn	Number of frogs per memeplexes in SFLA
Nq	Number of sub-memeplexes in SFLA
P_j	Triangular probability for sub-memeplex selection in SFLA
P_{shift}	Probability of shifting an activity
P_t	Payment of period t
Pre_p	Set of all predecessors of activity p
r_d	Daily interest rate
r_i	Interest rate charged on day i
r_w	Weekly interest rate
Rc	Retained cash
$S_{p,i}$	Start of activity p at day i
sm	Sufficiently small number
t	Week number
T	Project duration (days)
TB_{max}	Multi project's total maximum negative cash flow
$Temp$	Temperature in SA
TF_p	Total float of activity p
W	Credit limit
W_{total}	Multi projects credit limit
B_{max}	Maximum negative balance
WB_t	Weekly balance for period t
x_p	Start time of activity p
y_i	Cost disbursement of all activities performed at day i
α_t	Last payment indicator variable 1

β_t	Last payment indicator variable 2
δ_i	Financing cost indicator variable
Δ	Energy difference in SA
λ_p	Shift of activity p
Λ	Shift vector representation of a solution
Λ_B	Best frog within a sub-memeplex in SFLA
Λ_{GB}	Global best frog in SFLA
Λ_W	Worst frog within a sub-memeplex in SFLA
λ_p	Shift of activity p
τ_p	Adjusted total float of activity p
ρ_t	Interest on the outstanding debt indicator variable
$\sigma(t)$	Standard deviation of the population at time t in GA
Φ	Penalty factor
ψ	Cooling schedule factor in SA
$\lceil \ \rceil$	Ceiling function
$\lfloor \ \rfloor$	Floor function

.

Chapter 1

Introduction

A crucial challenge for construction contractors to run a sustained business represents the ability to timely procure adequate cash to execute construction operations. Alongside payments from their customers, contractors often procure additional funds from external sources including banks. Typically, such cash incurs financing charges. Given the facts that customers actually pay after the accomplishment of the work while retaining some money, and the cash that contractors can withdraw from banks is limited in amount, contractors often operate under cash-constrained conditions. The most proactive operating strategy contractors can follow for financial planning is to devise project schedules based on cash availability. Unfortunately, this integration between scheduling and financing functions is rare in the current research and is missing in the commercial scheduling software. The concept and technique of finance-based scheduling achieves the desired integration between scheduling and financing by incorporating financing costs into the project total cost as well as scheduling under cash constraints. The following section outlines the principles of the finance-based scheduling.

1.1 Finance-Based Scheduling

Contractors often procure funds from banks by establishing credit-line accounts. Typically, cash procurement through the banks' credit lines incurs financing costs. Contractors normally deposit the progress payments into the credit-line accounts to continually reduce the outstanding debit and consequently the financing costs. As the cash flow in Figure 1.1 indicates, contractors charge the expenses caused by labor, equipment, materials, subcontractors, and overheads (cash outflows ET_t) against, and deposit progress payments (cash inflows P_t) into the credit-line accounts. In practice, it can be reasonably assumed that these transactions occur as of the cut-off times between periods. Accordingly, the cash out flow, ET_t, and the financing cost, I_t, as of the cut-off times are determined. The summations of the values of the cash out flow, ET_t, and the respective financing costs, I_t, and the outstanding debt constitute the negative cumulative balance B_{mt}. The cumulative net balance value, N_t, constitute the cumulative balance after depositing the progress payments. The cumulative net balance of all the cash inflows and outflows constitutes the profit, G, as of the end of the project. The complete formulation of the previous financial parameters will be explained later in the next chapter.

Another concern of financing, though more important than the incorporation of financing costs, constitutes the credit-limit constraints imposed on the credit lines. The credit limit specifies the maximum value the negative cumulative balance as of the cut-off times are allowed to reach. Thus, finance-based scheduling achieves the desired integration between scheduling and financing by incorporating financing costs into the

project total cost as well as scheduling activities' such that the values of the negative cumulative balance as of the cut-off times never exceeds the specified credit limit. The techniques employed to devise finance-based schedules normally fulfill this financial constraint with the objective of minimizing the project's duration or maximizing the project's profit.

Being an aspect of the whole corporate rather than the individual projects, contractors manage the financing aspect at the corporate level. In other words, contractors' concern is generally to timely procure cash for all ongoing projects. Finance-based scheduling in this context ensures that the resulting values of the negative cumulative balances of all projects do not add up to exceed the credit limit, whereas the positive cumulative balances that occur in some projects are utilized to schedule activities of some other projects. This ensures that scheduling concurrent projects can be related to the overall liquidity situation of contractors. The sole objective of maximizing the profit of a single project is changed in this context to the objective of maximizing the profit value of all ongoing projects. Finance-based scheduling techniques schedule projects' activities such that the total profit of the projects is maximized while the financial constraint is fulfilled.

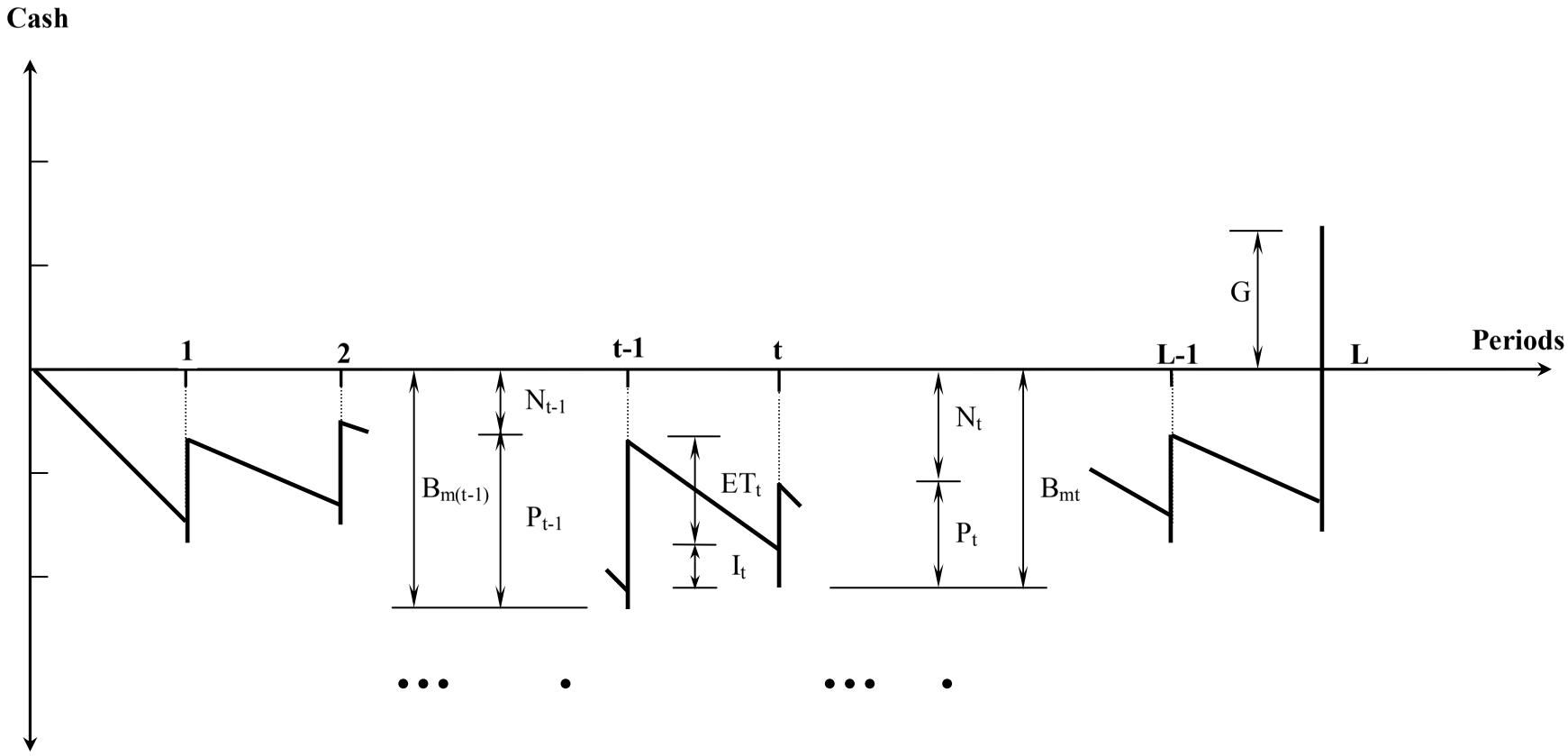


Figure 1.1: Cash flow diagram of a typical construction project.

1.2 Research Objectives

The objective of this research is to develop various solution methods to solve large scale real life finance based scheduling problems within a reasonable time. Exact solution to the problem using integer programming is given. Later, a number of meta-heuristics will be applied to solve to the problem.

1.3 Research Methodology

1. Modify the current cash flow model to reflect accurate financing cost calculations by incorporating the different cases of interest calculations.
2. Design an exact solution method to solve the problem by incorporating the modified cash flow model in the integer program.
3. Employ meta-heuristics such as: genetic algorithm, shuffled-frog leaping, simulated annealing to solve the single and multi projects finance based scheduling problems.

1.4 Thesis Organization

The rest of this thesis is organized as follows. In chapter 2, a literature review on related work is presented. In chapter 3, the modeling of the finance based scheduling problem is presented and the parameters of the cash flow model are explained. In chapter 4, an overview on the meta-heuristics used is presented followed by the implementation of the

meta-heuristics to the single project finance based scheduling problem. In chapter 5, the implementation of the meta-heuristics to the multi projects finance based scheduling problem is given. In chapter 6, the results of the study are presented along with a discussion. And finally in chapter 7, the thesis concludes with a conclusion and possible future work.

Chapter 2

Literature Review

Until the 70s of the previous century, the cost considerations in project scheduling were only in terms of the total cost of the project and the time value of money was entirely omitted (Kazaz and Sepil 1996). The research which considers the time value of money addresses the financial implications of the project activities. When the financial aspects of project management emerged, the Net Present Value (NPV) was the most frequent criterion used in project scheduling. The NPV is being determined using cash outflows and cash inflows of the project. For the contractor, cash outflows represent the expenses caused by labor, equipment, materials, and subcontractors while cash inflows represent the owner's payments. The NPV was first introduced by Russell in 1970. Russell's model was based on Activity On Arrow (AOA) representation and assumed that cash flows occur at the event times of AOA network nodes. Russell's model to determine the event times which maximize the NPV with absolutely no consideration of any kind of resource. In an extended effort, Kazaz and Sepil (1996) invalidated the assumption made by Russell that cash inflows occur at the realization times of some events during the course of the project. Kazaz and Sepil considered the problem of project scheduling developed a mixed integer programming formulation to maximize the NPV where the cash inflows occur as progress payments for the work completed during each month and cash outflows occur at the completion of activities. Kazaz and Sepil developed a mixed

integer programming formulation that determines how much the finish time of each activity can be delayed beyond their earliest finish times so that the NPV of the cash flows associated with all the activities of a project is maximized. However, Kazaz and Sepil didn't present any resource-constrained scheduling technique.

Subsequently, the problem of NPV maximization was expanded by other studies (Russell 1986, Padman and Smith-Daniels 1997) to include resource constraints. This problem represents a resource-constrained project scheduling problem with discounted cash flows. These two studies evaluated the performance of heuristic rules in scheduling resource-constrained projects to maximize the NPV of cash flows. In a more recent research effort, Chiu and Tsai (2002), considered the significant effect of the high cost of capital, and proposed an efficient priority-based heuristic rule for the resource-constrained multi projects scheduling problem to maximize the project net present value. However, the research trend in these three papers didn't present a cash-constrained scheduling technique. The exclusive justification for using the cash flows in these three studies was to achieve the objective of maximizing the monetary objective of NPV rather than the traditional objective of minimizing the total project duration which was used throughout the other resource-constrained scheduling research in the literature.

A sub problem of the previous resource-constrained scheduling with NPV maximization represents the problem of NPV-optimal with capital-constrained scheduling. Doersch and Patterson (1977) published the pioneer work in which the NPV is maximized while a limit on the amount of capital is available, followed by two remarkable papers of Smith-Daniels D.E. and Smith-Daniels V.L. (1987) and Smith-Daniels et al. (1996). Doersch and Patterson (1977) defined the capital-constrained

project scheduling problem as scheduling a project with both the positive and negative cash flows that take place over the course of the project, where investment in project activities is constrained by a capital constraint. This formulation discounts all cash flows occurring within an activity to the end of the activity. Smith-Daniels D.E. and Smith-Daniels V.L. (1987) introduced an approach to the project scheduling problem where the NPV of a project is maximized subject to capital and material constraints. This work considered the integration between material acquisition decisions with the process of scheduling the project activities. Schedules and the associated acquisition plans become infeasible when capital constraint is relatively tight with respect to activity capital requirements at a particular point in the project. Smith-Daniels et al. (1996) presented heuristic methods to solve the intractability problem of optimal solution for capital-constrained NPV-optimal problem.

The previous models of capital-constrained scheduling suffered from the major drawback of using the time unit of month to specify the durations of activities and determine activities' shifts which makes these models entirely invalid for the scheduling projects such as construction projects where activities durations and shifts need to be specified using the time unit of the working day. In other words, activities' shifts expressed in terms of months are not acceptable at all in construction projects. In addition, these models assumed that cash inflows occur at the realization times of some events during the course of the project. However, this assumption contradicts the typical practice in construction industry where cash inflows occur regularly as of the ends of fixed periods set forth by the owner in the contract, usually as of the end of the month for the work completed during the same month, to compensate contractors for the finished

and partially finished activities during the month. The capital-constrained scheduling models can accommodate financial planning of a big enterprise implementing a set of small projects each represented by a node on a large network that combines all the small projects, the financing of each small project may yield a requirement and a payback (Doersch and Patterson 1977). In other words, these formulations can be used to treat any set of investments in which the desired final state is known, but the timing sequence is optional (Doersch and Patterson 1977).

Following the same trend of research in the general area of project scheduling till the 70s of the previous century as outlined above, the research in project scheduling, specifically construction project scheduling, was directed towards minimizing the total project cost with no consideration given to the time value of money. For instance, the resource allocation techniques schedule limited resources to minimize the increase in project time and consequently project overhead cost. The resource leveling techniques schedule activities to minimize fluctuation in resource usage thereby minimizing the cost of recurrent hiring and laying-off and non-efficient operation. Time/cost tradeoff techniques minimize the total project cost considering the overhead costs and cost of crashed-duration activities. Thus, the vast majority of cost-optimization scheduling techniques in construction projects entirely discarded the financing cost which represents a direct cost component for the object of the project. However, few notable research efforts in construction have identified financing costs as a project cost component. Karshenas and Haber (1990) divided the cost of a resource in a construction project into resource mobilization cost and resource use cost. They identified cash as a separate resource and argued that it is required sometimes to keep the net monthly cash flow

(revenues less expenses) within a certain limit. However, Karshenas and Haber didn't identify the financing cost properly but considered the cash use cost, which is supposedly the financing costs according to the definition given in the same paper that resource use cost is a function of the use of resource, as the sum of activity costs for a given period. In addition, the formulation of the constraint fulfilling that the required resources should be less than the available resources for all project periods is absolutely not applicable to cash as the major financing component of the owner's payments was entirely neglected. Hegazy and Ersahin (2001) developed a spreadsheet-based model that combines a CPM network scheduling with time/cost tradeoff analysis, resource allocation and leveling, and cash flow management. The Genetic Algorithm (GA) technique was then used to optimize the overall schedule, considering all aspects simultaneously. In this model, cash flow computations were formulated regarding daily expenditures, cumulative expenditures, owner payments, and cash flow balance. The financing cost which was obtained from the cash flow calculations is then added to the total project cost. Li (1996) presented a mathematical model to schedule multiple subprojects with the objective of minimizing the construction costs. This model considered the interest cost associated with the investments on the individual subprojects. However, Karshenas and Haber, Hegazy and Ersahin, and Li didn't introduce the concept of finance-based scheduling which is to devise cash-constrained schedules. Warszawski (2003) presented a parametric model for the evaluation of financing cost in a construction project without incorporating his model into scheduling techniques.

Apart from scheduling techniques, other related research efforts in project management were directed to cash flow forecasting and management models. Sears

(1981) presented a method of accomplishing integration between project schedule and cost. This method produces an expense flow projection by assigning estimated costs to the time-scaled CPM network. Au and Hendrickson (1986) model allows contractors to enter the cash inflows and outflows as of the ends of periods which could be weeks or months, use the entered values to calculate the values of other financial parameters, and utilize the entered and calculated values of parameters to delineate the project cash flow. Barbosa and Pimentel (2001) developed a linear programming model for optimal cash flow management of a single construction project. The model considered investments with distinct asset returns and level of liquidity, and also available credit lines from banks. The optimization algorithm finds an efficient way to manipulate the cash transactions over the project duration, aiming at achieving a greater profitability at the end of the project. Navon (1995) presented a resource-based computerized cash-flow forecasting model. The model automatically integrates the bill of quantities, the estimate and the schedule databases using a non-project-specific database. Automating the cash flow forecast, as proposed by Navon, ensures highest accuracy, avoids manual labour, and becomes generic enough. Kaka and Lewis (2003) presented a dynamic cash flow forecasting model that assist contractors to effectively plan and manage the cash flow of individual projects and at a company level. These models presented different techniques to help contractors perform financial planning and management. However, these models failed to address the aspect of financial planning and management through the tool of scheduling, though this is the most effective approach since contractors have full control on scheduling their own activities.

A recent study (Elazouni and Gab-Allah 2004) developed an integer-programming model to devise finance-based schedules. This model revises CPM activities start times to produce minimum-duration schedules that correspond to desired credit limits. This method renders schedules executable under overdrafts of specified credit limits. The model considers the direct expenses of activities and add indirect expenses of job overhead, taxes, markup, and bond on a pro rata basis. However, the integer program they developed was a static model that can't adequately model all expenditures and income cash flows and simultaneously perform the necessary adjustments as the original schedule is being extended.

Genetic Algorithms (GA) technique was used (Elazouni and Metwally 2005) to search for a solution for the problem of devising CPM schedules that correspond to desired credit limits. For a particular project, schedules are generated using random start times of activities while maintaining dependency between activities. The corresponding profiles of cash requirements of these randomly generated schedules are produced. Then, the GA procedure searches for the schedule that produces debit values below the specified credit limit, minimizes financing costs, minimizes project indirect costs through minimizing project duration, and ultimately maximizes project profit. The GA method provided full flexibility to model project disbursements and income cash flows. However, the previous two studies were concerned mainly with developing CPM schedules financed by constricted credit limits which tend to prolong project duration.

Elazouni and Metwally (2007) utilized compressed activities to broaden the scope of the finance-based scheduling introduced in Elazouni and Metwally (2005). The broadened finance-based scheduling enables schedulers to schedule under relaxed as well as constricted credit limits and investigates the effect of the variation in credit limits on the total project costs. In addition, this paper employed resource allocation and leveling techniques to schedule under resource limitations and ensure the efficient use of resources. Consequently, the objective of the optimization is to minimize the total project costs. The constraints to the cost minimization represent the credit limit, resource availability. The demands of time minimization and efficient utilization of resources were fulfilled by expressing them as overhead costs and resource un-leveling penalty. Thus, finance-based scheduling constitutes an effective technique to manage cash, cost, time, and resources simultaneously. The GA technique was utilized as an environment to devise the overall-optimized project schedules.

Liu and Wang (2008) establishes a resource constrained project scheduling model based on constraint programming, whose solution can be found by using combinatorial optimization algorithms. The proposed model integrates the issue involving resource constrained problems and cash flow, and maximizes net project cash flow to optimize project profit from the perspective of contractors. They also performed model validation and two scenarios, including multi resource, resource combination selection and various constraints such as resource limit.

Elazouni (2009) proposed a heuristic method for scheduling multiple projects subject to cash constraints. The heuristic determines cash availability during a given period, identifies all possible activities' schedules, determines the cash requirements for each schedule, ranks schedules based on the contribution on minimizing the increase in the project duration, schedules all activities of the selected schedule, and determines the impact of the scheduled activities on the project cash flow. However, a major drawback in this heuristic is the extensive computational time needed when the set of eligible activities for one project is big or the time span of the period is long.

Chapter 3

Modeling of the Finance Based Scheduling Problem

In this chapter, we will first present the cash flow model for activity networks .After that, a mathematical model of the problem is presented followed by the modeling tricks used to put it as an integer program.

3.1 Cash Flow in Activity Networks

The cash flow model used in this thesis is based on the model proposed by Au and Hendrikson (1986) with some modification. The modification applied on the model made a more accurate calculation of the total financing cost which is the sum of the daily interest on expenditure, DI, and the periodic outstanding debt interest, NI. Moreover, this has helped in modeling an accurate way of calculating the financing cost in the integer program, i.e., the financing cost is only calculated if the contractor is in debt.

We consider the cash flow from the contractor's point of view. It should be noted that all cash-out transactions are entered as negative values and all cash-in transactions are entered as positive values. Typical cash-out includes costs such as disbursement, overhead and interest. In contrast, a usual cash-in is the payments received from the client. An example of a daily cash flow is shown in Figure 3.1.

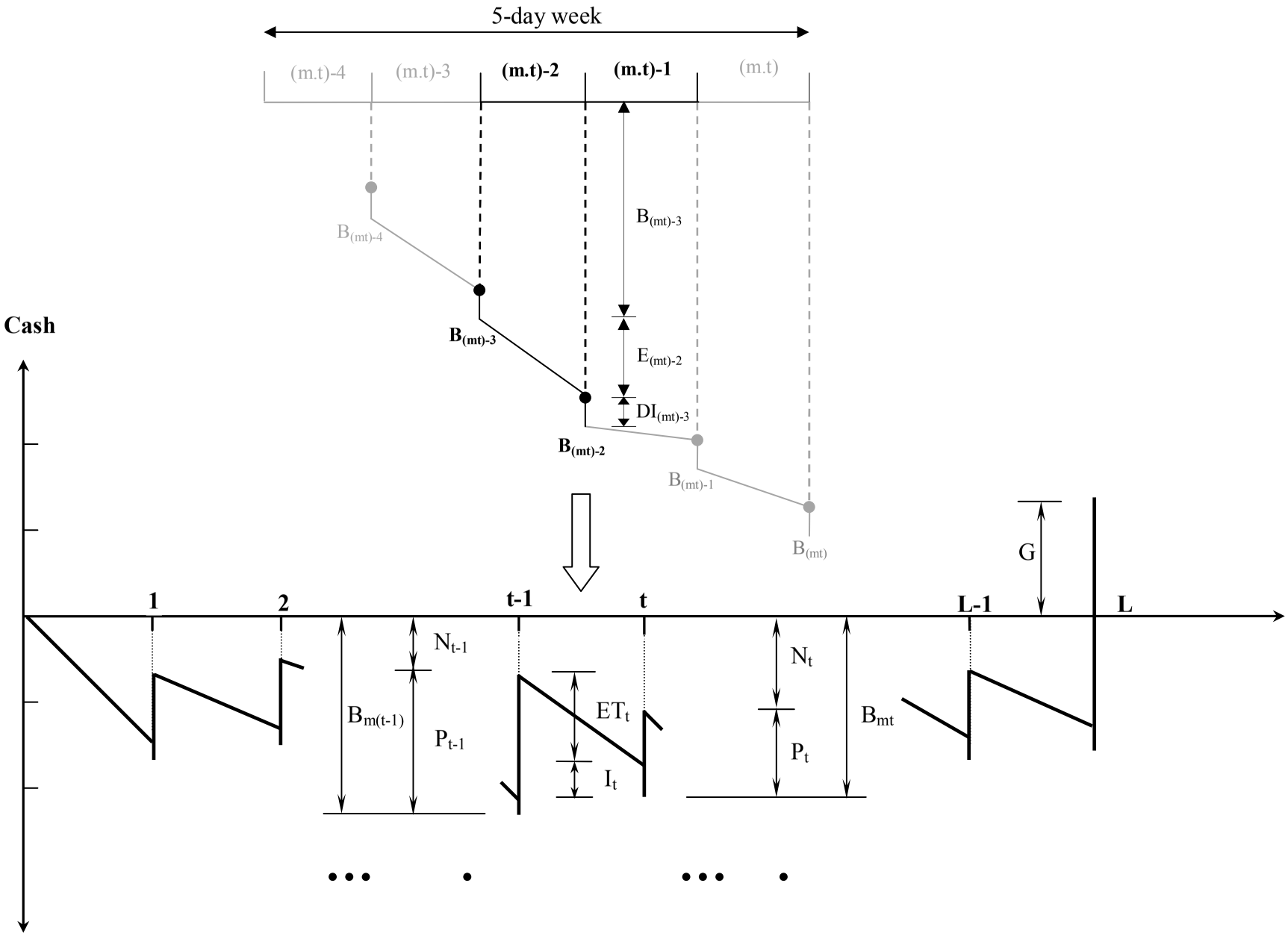


Figure 3.1: Daily cash flow example for a typical construction project.

The cash flow's daily transactions:

We assume that the contractor executing the activities borrows money from the bank daily as needed. This assumption is valid if the contractor pays for his expenses using checks drawn on the lending bank or a credit card provided by the lending bank. In addition, we assume that client payments are received periodically. Thus, daily and periodic cash flows are considered in this model where a period could be a week or a month. In Au and Hendrikson (1986), the interest on borrowed money is approximated by averaging the weekly debt to the bank. Usually this approximation is far from the exact value as shown in the example in Table 3.1. The example has two cases of weekly debt.

Table 3.1: Example cases.

	Case 1						Case 2				
Day	1	2	3	4	5		1	2	3	4	5
Expenditure	1000	1000	1000	2000	8000		8000	2000	1000	1000	1000

According to Au and Hendrikson's (1986) model, the financing cost is approximated by $I = r_w . ET/2$, where r_w is the weekly interest rate and ET is the total expenditure per week. Using this approximation, the financing cost is the same in both cases and it is equal to 19.5 for r_w =3%. However, this approximation is far from the exact value because in this case borrowing more money at the beginning of the period will accrue the same interest value as borrowing it at the end of the period.

In our cash flow model, we avoid this pitfall by considering the daily cash flow. This way, the financing cost can be calculated on a daily basis using a daily interest rate. The daily interest rate can be calculated using

$$(1 + r_d)^m = 1 + r_w$$

Where r_d and r_w are the daily and weekly interest rates respectively and m is the number of working days per week. , i.e. five working days in a week.

For five working days per week, if cash is borrowed on day i of the week and the interest is paid after the end of that week then the interest rate at day i is given by

$$r_i = (1 + r_d)^{m-i+1} - 1 \qquad i = 1,2,3,4,5$$

Using our proposed daily cash flow calculation with a weekly interest rate r =3%, the financing cost for cases 1 and 2 respectively are equal to 14.4 and 32.4.

Let A_i be the set of activities being executed at day i, and c_p , $p \in A_i$ be the direct cost disbursement of activity p. Then the direct cost disbursement of all activities performed at day i , y_i , is given by

$$y_i = \sum_{p \in A_i} c_p \qquad i = 1,2, \dots, T \qquad (3.1)$$

Where T is the number of days required to finish the project.

We assume that the contractor will borrow money from the lending bank at day i if there isn't enough money on hand to execute the activities scheduled for that day. The amount of money needed for day i is given by

$$E_i = k_1 . y_i \qquad i = 1,2, \dots, T \qquad (3.2)$$

Where E_i is the daily cost disbursement of day i and $k_1 > 0$ accounts for the overhead cost.

The cash balance in this model is updated daily, the balance at the first day is equal to the cost disbursement at the first day, $B_1 = E_1$. The cash balance is updated daily using

$$B_i = B_{i-1} + DI_{i-1} + E_i \qquad \begin{matrix} i = mt-3, \dots, mt-1 \\ t = 1,2, \dots, L \end{matrix} \qquad (3.3)$$

Where t denotes the week number and DI_i is the daily interest charged at day i which is accrued only if the cash balance at day i is negative $(B_i < 0)$. That is, the interest is only accrued if the contractor is in debt. The daily interest at day i is given by

$$DI_i = r_i . E_i \qquad i = 1,2, \dots, T \qquad (3.4)$$

The cash flow's weekly transactions:

A project total-cost disbursement during a typical period t is given by

$$ET_t = \sum_{i=m(t-1)+1}^{mt} k_1 . y_i \qquad t = 1,2, \dots, L \qquad (3.5)$$

Where L is the number of periods required to complete the project. The timing of receiving payments from the client depends on the contract between the client and the contractor. We assume that payments for a certain period are received at the end of that period. This payment is given by

$$P_t = k_2 . ET_t \qquad t = 1,2, \dots, L \qquad (3.6)$$

Where k_2 is a multiplier that accounts for the mark-up of the contractor and the retainage by the client, which is a held back amount of money to assure the quality of the work done.

Moreover, If the contractor owes money to the bank at the end of period $t-1$, named the net balance N_{t-1}, then interest NI_t is charged if the net balance is negative ($N_{t-1} < 0$) as follows

$$NI_t = r_w . N_{t-1} \qquad t = 1,2,\dots,L \qquad (3.7)$$

The total financing cost for period t is the sum of daily interest accrued over the week in addition to the outstanding debt interest. The total financing cost given by

$$I_t = NI_t + \sum_{i=m(t-1)+1}^{mt} DI_{i-1} \qquad t = 1,2,\dots,L \qquad (3.8)$$

The cash balance at the end of period t , $B_{(m.t)}$, represents the maximum cash flow at that period and is given by

$$B_{(m.t)} = N_{t-1} + ET_t + I_t \qquad t = 1,2,\dots,L \qquad (3.9)$$

The net balance of period t is the balance at the end of the week after receiving the payment and it's given by

$$N_t = B_{(m.t)} + P_t \qquad t = 1,2,\dots,L \qquad (3.10)$$

The balance at the start of the following week is given by

$$B_{(m.t)+1} = N_t + E_{(m.t)+1} + DI_{(m.t)} \qquad t = 1,2,\dots,L \qquad (3.11)$$

The objective of the finance based scheduling is to come up with a schedule that minimizes the project duration, such that the negative cash balance of each period t never exceeds a specified credit limit W.

$$B_{(m.t)} > -W \qquad t = 1,2,\dots,L \qquad (3.12)$$

In the next section, we introduce the integer program formulation of the cash flow model discussed above.

3.2 Modeling Cash Flow using Integer Programming

We consider activity networks with activity on node (AoN) representation (Demeulemeester & Herroelen, 2002). Figure 3.2 shows such a network where each node corresponds to an activity and arcs represent the precedence constraints. For activity p , the early start EST_p can be found using a forward pass calculation and the latest start LST_p can be calculated using a backward pass calculation (Uher, 2003). The total float of activity p, TF_p, is defined as the maximum shift in an activity's starting time that will not affect the project duration and is given by

$$TF_p = LST_p - EST_p$$

. The project duration T for the critical path is defined as the minimum time needed to finish all of the activities in the network. The search for a schedule using the exact method is bounded, that is each activity can only start between its early and late start values. However, developing schedules that are constrained by a specific credit limit might involve extending the project duration. That is, if the credit limit is small it may result in delaying some activities. For the sake of modeling this problem as an integer program, an extra duration, Ext, is added for each activity and the new extended late start $ELST_p$ is calculated using

$$ELST_p = EST_p + TF_p + Ext$$

Where $TF_p = 0$ for critical activities.

In the following subsections, the objective and the constraints of the mathematical model are introduced followed by the tricks used to build the integer program.

3.2.1 The Mathematical Model

The decision variables of the finance based scheduling problem are the starting times of the project activities. For a project with n activities, let x_p define the starting time of activity p, and $d_p$denotes the duration of activity p. The starting time of activity p is represented by a binary variable $S_{p,i}$ where $S_{p,i} = 1$ if activity p starts in day i and $S_{p,i} = 0$ otherwise. Hence, the variable x_p can be defined as

$$x_p = \sum_{i=Est_p}^{ELst_p} i.S_{p,i} \qquad (3.13)$$

The completion time of the project can be defined as the completion time of the last activity/activities in the project's network. For example, in the activity network shown in Figure 3.2, the maximum finishing time among activities K,L,M will define the completion time of the project.

The objective of the finance-based scheduling is to minimize the project's duration Z.

$$\text{Minimize} \quad Z$$
$$\text{Subject to} \quad Z \geq x_p + d_p \qquad p = 1,2,\dots,n \qquad (3.14)$$

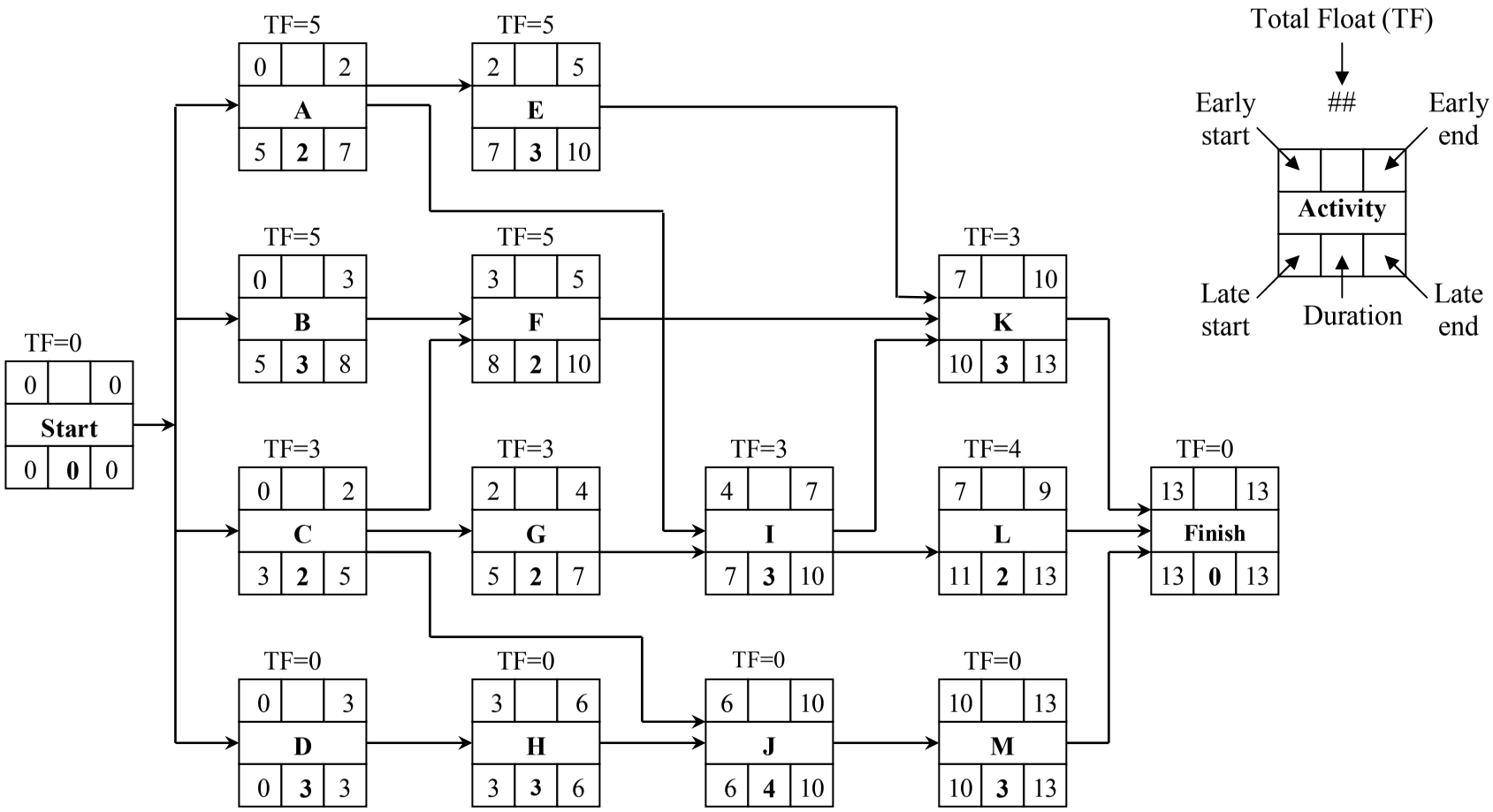


Figure 3.2: Activity network of a 13-activity project.

There are two types of constraints in the finance-based scheduling problem. These constraints are the network related constraints (precedence constraints) and the financial constraint. The precedence constraint between any two consecutive activities p and q where p precedes q is formulated as

$$x_p + d_p \leq x_q \qquad (3.15)$$

The financial constraint presented in (3.11) is the credit limit constraint which guarantees that the negative cash flow at any period t of the project's cash profile will never exceed the specified credit limit W. In order to incorporate this constraint in the mathematical model, the cash flow model equations are used to identify the weekly balance variable.

The weekly balance at any period is formulated in terms of cash-outs (daily disbursement, financing costs) and cash-ins (periodic payments). The disbursement, y_i, at day i is given by

$$y_i = \sum_{p \in A_i} c_p$$

Where $A_i = \{p \mid x_p \leq i \leq x_p + d_p\}$ and c_p is the cost of activity p.

The weekly expenditure, ET_t, and the payment received at the end of each period, P_t, are given by (3.5) and (3.6) respectively. The total incurred financing cost per week, I_t, is given by (3.8). Finally, the balance at the end of the week, the net balance after receiving the payment, and balance at the start of the following week, $B_{(m.t)}$, N_t and $B_{(m.t)+1}$, are given by (3.9), (3.10) and (3.11).

In order to implement the finance-based scheduling model, given by the system of equations (3.1-3.15), as an integer program, some modeling tricks are needed to employ the IF statements that appeared in the model (Williams, 1993). These modeling tricks are presented in the following sections.

It should be noted that this model overcomes a limitation in Elazouni et al. (2004) integer programming model in which the cash flow of the project is assumed to be always negative. In this thesis we allow the cash flow represented by the balance to be either negative or positive. A negative value means that the contractor is in debt while a positive value means that the contractor has cash on hand. It assumed in this model that contractor will use any cash on hand to execute the project and no financing cost is accrued in this case. On the other hand, if there is cash on hand but not enough to execute the scheduled activities, the contractor borrows the shortage from the bank. The financing cost accrued in this case is charged on the total daily expenditure even if the borrowed cash is less than the total daily expenditure.

3.2.2 Modeling the Financing Cost Calculation

There are two sources of financing cost, the daily interest and the weekly outstanding debt interest cost. These costs are formulated in the mathematical model, however, they should be modeled such that they are charged only when the cash balance is negative, i.e. the contractor is in debt. To model such condition, indicator variables are introduced to the model to indicate the state of the balance.

Let $\delta_i = \begin{cases} 1 & if\ B_i < 0 \\ 0 & if\ B_i > 0 \end{cases}$

This indicator variable can be modeled as follows

$$B_i \leq (1 - \delta_i).M$$

$$B_i \geq -\delta_i.M$$

Where M is a sufficiently large number.

The daily interest, DI_i, should be charged only if $\delta_i = 1$, this can be modeled as follows

$$DI_i \leq r_i.E_i$$

$$DI_i + (1 - \delta_i).M \geq r_i.E_i$$

$$DI_i - \delta_i.M \leq 0$$

$$DI_i \geq 0$$

Similarly, the periodic outstanding debt interest cost, NI_t, should be charged if $N_{t-1} < 0$

Let $\rho_t = \begin{cases} 1 & if\ N_t < 0 \\ 0 & if\ N_t > 0 \end{cases}$

This indicator variable can be modeled as follows

$$N_t \leq (1 - \rho_t).M$$

$$N_t \geq -\rho_t.M$$

$$NI_t + (1 - \rho_t).M \geq r_w.N_{t-1}$$

$$NI_t - (1 - \rho_t).M \leq r_w.N_{t-1}$$

$$NI_t - \rho_t.M \leq 0$$

$$NI_t \geq 0$$

3.2.3 Modeling the Last Payment

To assure the quality of the contractor's work, usually the client retains some of the money he owes the contractor. This retained money is paid to the contractor after completing the project. This amount of retained money is a percentage of the weekly payments and can be defined as a constant regardless of the schedule. This retained money should be added to the last payment the contractor receives. This can be modeled in the integer program using two indicator variables to indicate the end of the project. The project is completed when all the activities are finished, which is equivalent to the last period with disbursements.

Let $\alpha_t = \begin{cases} 1 & if\ ET_t > 0 \\ 0 & if\ ET_t = 0 \end{cases}$

This variable can be modeled as

$$ET_t - \alpha_t . M \leq 0$$

$$ET_t - \alpha_t . sm \geq 0$$

Where sm is a sufficiently small number and it should be less than any non-zero expenditure.

Another indicator variable is needed to indicate payment in which the retained amount of cash will be added. This variable will indicate a switch in α_t between any consecutive periods.

$$\alpha_{t-1} - \alpha_t = \beta_t \qquad t = 2, \dots, L$$

Finally, the retained money is added to the last payment using the formula

$$P_t = k_2 . ET_t + (Rc . \beta_t) \qquad t = 1, \dots, L$$

Where Rc is the amount of retained cash. It can be seen that the amount Rc will be added only when $\beta_t = 1$. In other words, the retained cash will be added to the last payment. The complete integer programming model is presented next.

Minimize Z

Subject to

$$x_p = \sum_{i=Est_p}^{ELst_p} i.S_{p,i} \qquad p = 1,2,\dots,n$$

$$Z \geq x_p + d_p \qquad p = 1,2,\dots,n$$

$$x_p + d_p \leq x_q \qquad 1 \leq p, q \leq n, \text{ and } p \text{ precedes } q$$

$$y_i = \sum_{p \in A_i} c_p \qquad i = 1,2,\dots,T$$

$$E_i = k_1.y_i \qquad i = 1,2,\dots,T$$

$$ET_t = \sum_{i=m(t-1)+1}^{m.t} k_1.y_i \qquad t = 1,2,\dots,L$$

$$ET_t - \alpha_t.M \leq 0 \qquad t = 1,2,\dots,L$$

$$ET_t - \alpha_t.sm \geq 0 \qquad t = 1,2,\dots,L$$

$$\alpha_{t-1} - \alpha_t = \beta_t \qquad t = 1,2,\dots,L$$

$$P_t = k_2.ET_t + (Rc.\beta_t) \qquad t = 1,2,\dots,L$$

$$DI_i \leq r_i.E_i \qquad i = 1,2,\dots,T$$

$$DI_i + (1-\delta_i).M \geq r_i.E_i \qquad i = 1,2,\dots,T$$

$$DI_i - \delta_i.M \leq 0 \qquad i = 1,2,\dots,T$$

$$DI_i \geq 0 \qquad i = 1,2,\dots,T$$

$$N_t \leq (1-\rho_t).M \qquad t = 1,2,\dots,L$$

$$N_t \geq -\rho_t.M \qquad t = 1,2,\dots,L$$

$$NI_t + (1-\rho_t).M \geq r_w.N_{t-1} \qquad t = 1,2,\dots,L$$

$$NI_t - (1-\rho_t).M \leq r_w.N_{t-1} \qquad t = 1,2,\dots,L$$

$$NI_t - \rho_t.M \leq 0 \qquad t = 1,2,\dots,L$$

$$NI_t \geq 0 \qquad t = 1,2,\dots,L$$

$$B_{(m.t)} = B_{(m.t)-1} + DI_{(m.t)-1} + E_{(m.t)} + NI_t \qquad t = 1,2,\dots,L$$

$$B_i = B_{i-1} + DI_{i-1} + E_i \qquad i = mt-3,\dots,mt-1$$

$$B_i \leq (1-\delta_i).M \qquad i = 1,2,\dots,T$$

$$B_i \geq -\delta_i.M \qquad i = 1,2,\dots,T$$

$$N_t = B_{mt} + P_t \qquad t = 1,2,\dots,L$$

$$B_{(m.t)+1} = N_t + E_{(m.t)+1} + DI_{(m.t)} \qquad t = 1,2,\dots,L$$

$$B_{(m.t)} > -W \qquad t = 1,2,\dots,L$$

$$S_{p,i}, \alpha_t, \beta_t, \rho_t, \delta_i = 0 \text{ or } 1$$

Chapter 4

Meta-heuristic Solutions to the Finance Based Scheduling Problem

Practice shows that real life engineering problems are usually of the large scale type. For some of these problems, efficient analytically based algorithms exist, such as linear programming, for obtaining globally optimal solutions. However, for discrete and/or combinatorial optimization problems, no such efficient algorithms are available for a general problem. This means that exact methods such as integer programming usually fail to reach the optimal solution in a reasonable time. This is due to the huge and complex solution space of these large scale problems which makes finding the optimal in reasonable time is almost impossible. To overcome this problem, researchers came up with "heuristics" that produce fast near optimal results for specific problems (Artigues, Demassey & Neron, 2008).

A heuristic is a procedure applied to a problem in order to get a good (near optimal) solution at a reasonable computation time and cost. In other words, it's a rule of thumb that will hopefully find a good answer but doesn't guarantee optimality. Heuristic are problem specific, that is, it's designed to work on a certain problem. However, some of these heuristics are alerted in a way that makes it general and not problem specific. These heuristics are named meta-heuristics (Michalewicz & Fogel, 2002).

Thus, a meta-heuristic can be defined as a heuristic method for solving a very general class of computational problems by combining a set of procedures in the hope of obtaining a more efficient or more robust procedure to find a good solution for a problem. Figure 4.1 shows the timeline of some of the main meta-heuristics used in the literature.

In this chapter, a description on the mechanism of each meta-heuristic is provided along with the pseudo code for each one.

The meta-heuristics that will be used are:

a) Genetic algorithms.

b) Shuffled frog-leaping algorithm.

c) Simulated annealing.

A general overview of each meta-heuristic and the implementation of each to the finance based scheduling problem will be discussed in the following subsections

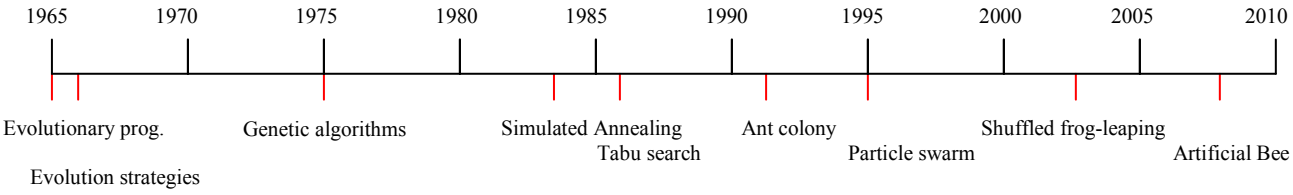


Figure 4.1: The time line of different meta-heuristics. (Source: Wikipedia.com)

4.1 Overview of the Meta-heuristics Used

4.1.1 Genetic Algorithms

Genetic Algorithms (GA) is a meta-heuristic search algorithm that is based on the evolutionary concepts of natural selection. The basic idea behind the GA is designed to simulate processes in natural system of evolution, specifically the principle of "survival of the fittest". As such it's considered an intelligent exploitation of a random search within a defined search space of a given problem. In other words, Genetic Algorithm is a search technique used to find an approximate, or if lucky an exact, solutions to optimization problems. Genetic algorithms are classified as a population based global search heuristic and more specifically as an evolutionary algorithm that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

The evolution starts from an initial population of randomly generated individuals and occurs over generations. Individuals in this population represent possible solutions to the problem. In each generation, the fitness of each individual in the population is evaluated according to the fitness function, multiple individuals are then selected from the current population (based on their fitness), and modified (recombined or randomly mutated) to form a new population. The newly generated population is then used in the next iteration of the algorithm. Usually, the algorithm terminates when either a maximum number of iterations has been carried out, or an acceptable fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of iterations, a satisfactory solution is not guaranteed (Eiben & James, 2003) (Michalewicz, 1994).

The major components of GA are: Representation, Fitness function (objective), Initialization, Selection, Crossover, Mutation and Repair.

Representation:

The most important step in Genetic Algorithms is representation of the solution domain in which the decision variables of the problem are gathered as an individual. Decision variables (genes) are structured as a vector (chromosome) which is the solution structure of the GA. Each chromosome is a representation of a complete solution to the problem yet it is in some cases not a feasible solution and will require the repair operator to return it to feasibility. The aim of the GA is to find the best feasible solution (individual) along many generations that evolve using the genetic operations of crossover and mutation. Thus it can be seen that the chromosome representation is a crucial step in any GA.

Initialization:

Initially a number of individual solutions are randomly generated to form an initial population. The population size is a parameter that depends somehow on the complexity of the problem, and usually it ranges from several hundreds to thousands of possible solutions. The random generation of the population should cover the entire range of possible solutions (the search space). Typically, the initial population is generated using an upper and lower limit for each gene. The gene is created between these limits using a random number generator.

Fitness function:

For any given problem, the fitness function is defined using the objective function of that problem. The fitness function is used to measure the quality (fitness) of a given individual (solution). Thus, the fitness function always depends on the problem. For instance, in the knapsack problem in which the objective is to maximize the total value of objects that can be put in a knapsack of a fixed capacity. A representation of the solution may be an array of bits, where each bit corresponds to an object, and the value of the bit (0 or 1) denotes whether the object is in the knapsack or not. Not every such representation is reasonable, as the size of objects may exceed the capacity of the knapsack. The fitness of the solution in this example is the sum of values of all objects in the knapsack if the representation is valid or 0 otherwise.

Selection:

In order to simulate the process of evolution, a proportion of each generation is selected to breed the offspring which will be the new generation. Individual solutions can be selected through different selection schemes. Most commonly, a fitness-based process of selection is used, where fitter solutions (as measured by a fitness function) are more likely to be selected. Certain selection methods rank the fitness of each solution and stochastically select the best solutions. Other methods rank only a random sample of the population, as this process may be very time-consuming if the fitness calculation complex.

Most selection functions are stochastic and designed so that less fit solutions are selected but in small proportions. This helps in avoiding premature convergence to poor

solutions by keeping a diverse population. Popular selection methods include roulette wheel selection and tournament selection.

Crossover:

Crossover is the process of generating a new population (offspring) by mating the selected parents of the current generation. Each new solution (child) is produced by a selected pair of "parent" solutions from the pool selected previously. Offspring solutions share many of the characteristics of its "parents". New parents are selected to produce a new generation, and the process continues until the termination criterion is satisfied.
These processes yield a population of chromosomes that is different from the initial generation. Commonly the average fitness of the population keeps improving, since only the best individuals from the first generation are selected for breeding, along with a small proportion of less fit individuals.

Mutation:

The process of mutation is done by randomly altering some selected individuals in the population. This mutation is done by changing the value of one or more genes in an individual. Mutation operation is usually used to avoid premature convergence to a local optimal solution by introducing some variation in each population. Usually the percentage of mutation is set to low values.

A pseudo code for the algorithm is given in Table 4.1.

Table 4.1: GA pseudo code.

```
Begin
Generate a new population of solutions;
While (terminating condition not met)
        {
        Evaluate solutions through fitness assignment;
        Select best individuals to reproduce based on their fitness value;
        Breed new solutions through crossover operator;
        Mutate;
        Repair;
        }
End
```

4.1.2 Shuffled Frog-Leaping Algorithm

The shuffled frog-leaping algorithm (SFLA) is a memetic meta-heuristic that is based on the evolution of memes carried by interactive individuals, and on a global exchange of information among themselves. A meme can be defined as a transmittable information pattern that replicates by infecting host minds and altering their behavior, which causes them to propagate the pattern. In other words, a meme is any kind of information that survives long enough to be recognized as such and that can pass from mind to mind.

Eusuff, Lansey and Pasha (2006) designed this meta-heuristic by combining the ideas of the shuffled complex evolution algorithm and the particle swarm optimization. Traditional evolutionary algorithms like GA are based on the concept of population which is a set of individuals. Each individual is associated with a fitness value that measures how good it is. On the other hand, SFLA the individuals are not so important yet they are seen as hosts of memes. Each host carries a meme that is analogous to the chromosome in GA. While genes can only be transmitted from parents to offspring, memes can be transmitted between any two individuals. Thus, a better individual (solution) that takes generations to propagate takes a relatively shorter time to spread in the SFLA population.

The name of this method came from applying this memetic approach to a group of frogs leaping in a swamp and searching for food. The swamp has a number of stones at discrete locations on to which the frogs can leap to find the stone that has the maximum amount of available food. The frogs are allowed to communicate with each other, so that

they can improve their memes using others' information. Improvement of a meme is done when a frog that is far from the stone with the maximum amount of food leaps toward a frog closer to the food. This leap results in altering the faraway frog's position to be closer to the stone with maximum amount of food. Here, the change of memes is only allowed to be a discrete value by analogy with the frogs and their discrete positions.

Steps of SFLA:

The search begins with a randomly selected population of frogs covering the entire swamp. The population is partitioned into several parallel communities, called memeplexes, which evolve independently to search the space in different directions. Within each memeplex, the frogs are infected by other frogs' ideas which results in an improvement in the individual frog's performance towards a goal. To ensure that the infection process is fruitful, it is required that frogs with better memes (ideas) contribute more to the development of new ideas than frogs with poor ideas. Thus, the selection process for frogs using a triangular probability distribution provides a competitive advantage to better ideas.

During the evolution, the frogs may change their memes using the information from the memeplex's best frog or from the best frog among the entire population. Incremental changes in memes correspond to a leaping step size and the new meme corresponds to the frog's new position. After an individual frog has improved its position, the community will be able to make use of this improvement to find a better one closer to the maximum amount of food. After a certain number of memetic evolutions between the frogs of each memeplex, the memeplexes are forced to mix and new memeplexes are

formed through a shuffling process. This shuffling enhances the quality of the memes after being infected by frogs from different regions of the swamp. This in turn accelerates the searching procedure for frogs by sharing their experience in the form of infection and it ensures that the cultural evolution towards any particular interest is free from regional bias (Eusuff et al., 2006) (Chung & Lansey, 2009).

A pseudo code for the algorithm is given in Table 4.2.

Table 4.2: SFLA pseudo code.

```
Begin
Generate a virtual population of frogs
While (convergence criteria not met)
        {
        Sort the population of frogs in order of decreasing fitness value
        Partition frogs into memeplexes
        While (evolution for all memeplexes not done)
                {
                Memetic evolution within each memeplex (Local Exploration)
                }
        Shuffle memeplexes
        }
end
```

4.1.3 Simulated Annealing

Simulated Annealing (SA) was first introduced by Kirkpatrick, Gelatt and Vecchi (1983). In the 1980s, SA had a huge impact on the field of heuristic search for its simplicity and efficiency in solving combinatorial optimization problems. The idea of simulated annealing was inspired by the annealing process in metallurgy, a technique involving the heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. For a minimization objective, the heat frees atoms to be able to move from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slow cooling gives those more chances of finding configurations with lower internal energy than the initial one.

Using the same idea, this process is simulated and applied to search for a feasible solution to an optimization problem. Starting with an initial solution, SA algorithm searches for a nearby solution in by alerting the current solution using a generation function. This is similar to the local neighborhood search method which usually tends to be trapped into a local optima rather than the global one. The SA method was built to avoid such trap by assigning a probability to accept a new solution even if it's worse than the current one. This acceptance probability is controlled by the Temperature parameters which starts high and tends to cool down over the iterations. So during the initial iterations, the temperature is high and the probability of accepting worse solution is high. The temperature is decreased over the iteration using a certain cooling schedule which will result is lower probabilities of accepting bad solutions. The search continues until a

stopping criterion is met. This simulation helps in exploring the feasible solution space at higher temperatures and avoids premature convergence to local optima.

The pseudo code for the algorithm is given in Table 4.3.

Table 4.3: SA pseudo code.

```
Begin
Initialize temperature
Randomly generate an initial solution IS
Calculate the initial solution's energy f(IS)
While (Stopping criteria is not met)
        {
        Update the temperature using the cooling schedule
        While (number of iterations at each temperature are not met)
                        {
                        Generate a neighbor solution IS'
                        Calculate the new energy for this solution f(IS')
                        Delta = f(IS') – f(IS)
                        If Delta ≤ 0
                                Accept the new solution IS=IS'
                        Else
                                Accept the new solution with a probability e^(-Delta/Temp)
                        End if
                        }
        }
End
```

4.2 Meta-heuristics' Implementation

In the following subsections, the common concepts that are shared among the heuristics are introduced first. Later, the implementation of each heuristic is illustrated using the 13-activity project network shown in Figure 4.2.

4.2.1 Common Concepts among Meta-heuristics

In order to design a heuristic, a representation of the solutions handled by the algorithm should be carefully chosen. In addition, the definition of the objective function that will guide the search is also important. Choosing a good representation and defining a suitable objective function greatly depends on the problem's constraints. For any optimization problem, constraints can be handled, when a heuristic is designed, using different strategies such as rejection, penalization, decoding and repairing strategies. In this section, the representation (encoding) of the solution is demonstrated. Then, the objective function is defined for the penalization and the repairing strategies respectively. Finally, the repair algorithm used for the repairing strategy is presented.

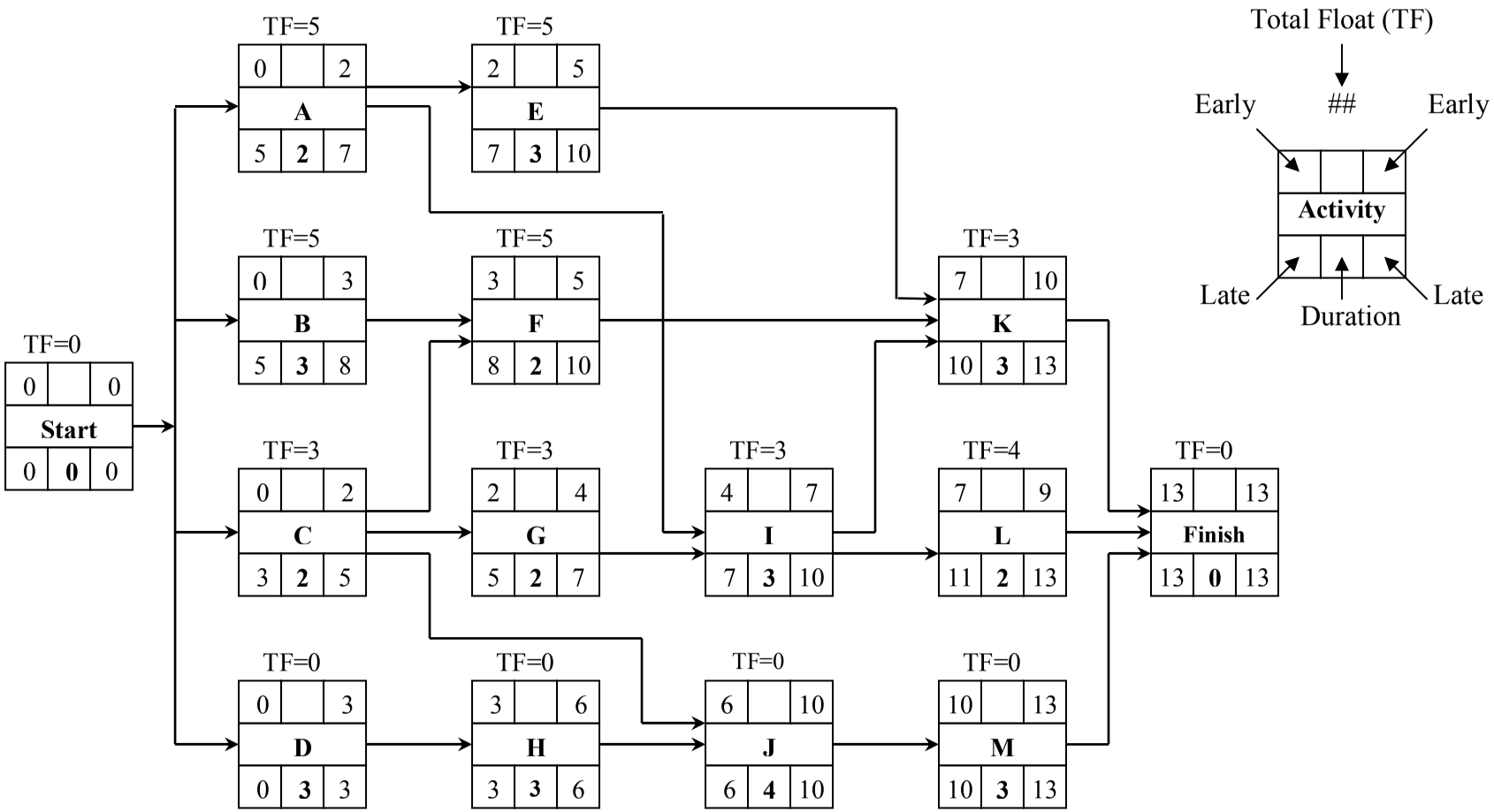


Figure 4.2: Activity network of a 13-activity project.

Representation:

One of the most important steps in the design of a heuristic is the encoding (representation) of a solution as it plays a major part in the efficiency of the heuristic. Encoding helps in handling some of the problem's constraints and taking advantage of this will help in improving the effectiveness of the designed heuristic. As been presented earlier in the finance-based scheduling model (Chapter 4), there are two main constraints; the network precedence relations constraint and the financial constraint. Choosing the right representation will take care of one of these constraints.

In our problem, decision variables are the starting times of each activity. This can be represented by a vector that shows the starting time of each activity. While this might sound like the best way to represent the solution, representing the solution this way will allow infeasibility with respect to the precedence constraint. That is, the starting time of each activity is not forced to follow the precedence relation's constraint. Moreover, starting with a feasible solution or a population of feasible solutions will not solve the problem of violating the precedence relation constraint because each heuristic has its own method of searching and generation of new potential solutions. However, the shift vector representation will guarantee that all solutions are feasible with respect to the precedence relation constraint.

In the shift vector representation, each solution is represented by a vector where each non-negative integer in a position indicates how many days the corresponding activity is scheduled beyond its early start time. The shift vector representation is illustrated in Figure 4.3.

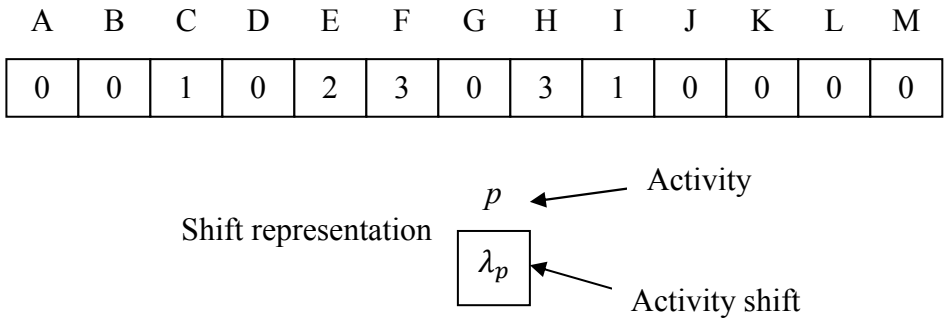


Figure 4.3: Shift vector representation.

Let λ_p be the shift of activity p and $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_p\}$ be the shift vector representation of a solution. In order to get the starting times vector (solution) from the shift vector representation a decoder must be used. This is illustrated in Figure 4.4 using the example network given in Figure 4.2. The decoder's pseudo code is given in Table 4.4.

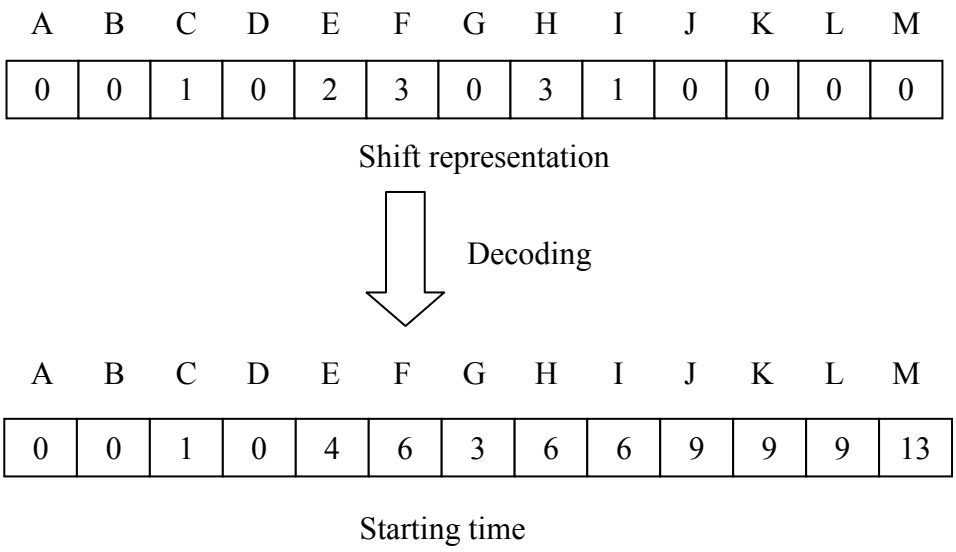


Figure 4.4: Decoding example.

Table 4.4: Pseudo code for the decoder.

Let

λ_p = the shift of activity p

EST_p =the early start of activity p

EFT_p = the early finish of activity p

d_p = the duration of activity p

Pre_p = the set of all predecessors of activity p

Begin

EST_1=0

$EFT_1 = EST_1 + d_1$

For activities p = 1 to n

$$EST_p = \max\{EFT_q | q \in Pre_p\} + \lambda_p$$

$$EFT_p = EST_p + d_p$$

End

It should be noted however that this representation doesn't guarantee a financially feasible schedule. That is, the negative cash balance may exceed the credit limit at any point along the project's duration. This can be treated by either adding a penalty in the objective function (penalization strategy) or by using the repair operator (repair strategy) as will be seen later.

Generation of the initial solution:

Initial solution is required for population based heuristics, like GA and SFLA, as well as the single based heuristics such as SA. The quality of the initial solution greatly affects the effectiveness and the efficiency of the heuristic. For the population based heuristics, the initial population should be diverse to avoid premature convergence. Thus, it's important to generate a diverse population that covers the search space of the problem.

The generation is done using an upper bound to the shift vector and a uniform random number generator. The adjusted total float, τ, can be used as the upper bound as it takes care of the total float of each activity as well as any Additional Shift Units, *ASU*, added to the total float of each activity if needed. To have a reasonable solution, not all activities are shifted. Thus the random generator code should randomly shift some of the activities as shown in Table 4.5. An example of a generated schedule for the example network is given in Figures 4.5 and 4.6.

It should be noted that the precedence relation constraint is feasible in all of the generated schedules but the repair operator should be used to repair any financial infeasibility.

Table 4.5: Pseudo code for the initial solution generator.

Let

P_{shift} = the probability of shifting an activity

τ = the adjusted total float vector.

Begin

Generate a random number for each activity

For i=1 to p

If rand(i)< P_{shift}

$$\lambda_i = \lfloor rand[0,1].\tau_i \rfloor$$

Else

$$\lambda_i = 0$$

End if

End

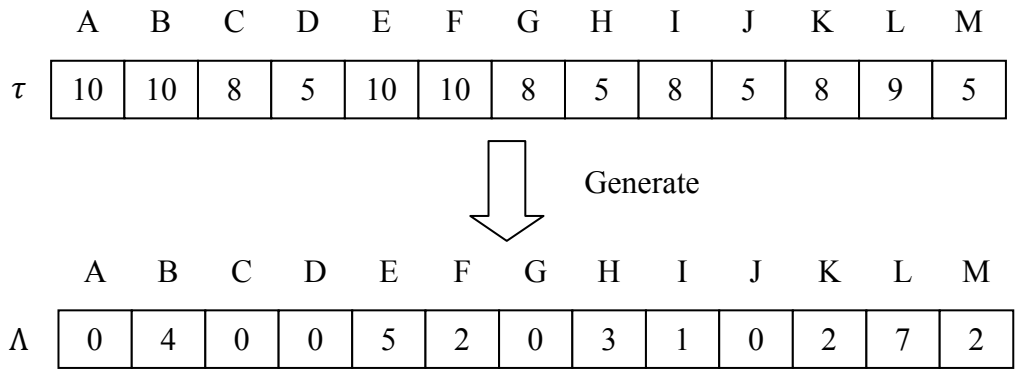


Figure 4.5: Generated schedule with *ASU* =5.

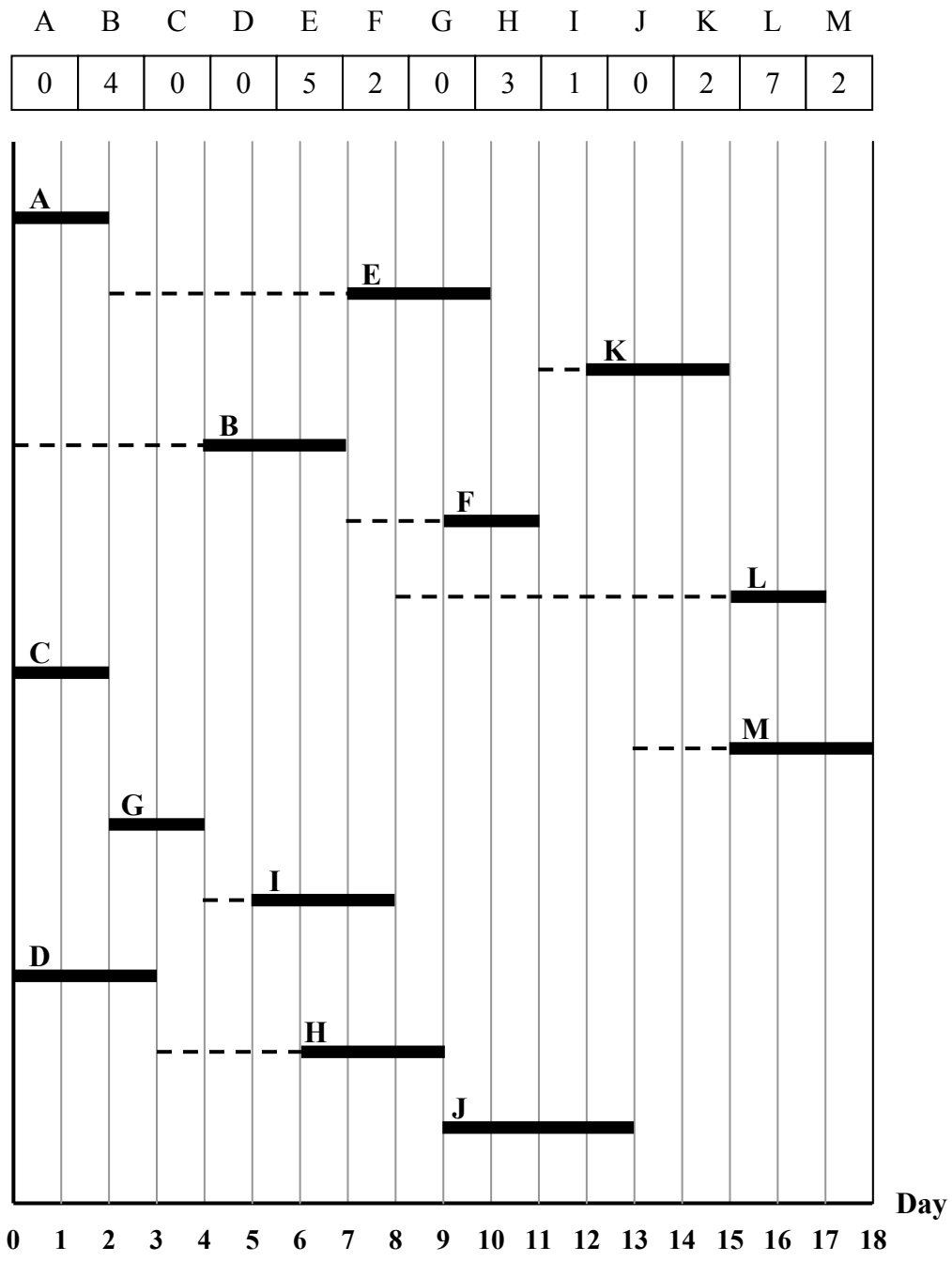


Figure 4.6: Example of a generated schedule.

The objective function:

The objective of the finance-based scheduling is to minimize the project's duration Z as given in (3.13). The objective (fitness function) when using the repair strategy is

$f(S) = Z$ (Repair used)

In this case the repair operation will always keep solution(s) feasible with respect to the financial constraint. However, a penalty should be added to the objective function if the penalization strategy is used. The objective (fitness function) function will look like

$f(S) = Z + \Phi\{max(0, B_{max} - W)\}$ (Repair not used)

Where Φ is a penalty factor, B_{max} is the maximum negative balance and W is the specified credit limit.

Repair algorithm:

As seen before, the encoding method guarantees feasible schedules with respect to the precedence relation constraint but doesn't warrant the credit limit feasibility. The repair function will repair any financial infeasibility by shifting the starting time of some activities in order to keep the balance within the credit limitation. For a given infeasible schedule, the repair algorithm will identify the period in which the credit limit constraint was violated. The algorithm will then chose an activity to be shifted according to a certain criterion. Activities will be shifted until the credit limit constraint for that period is satisfied .The algorithm will then move to the next period to make sure it's financially feasible. This is done for all the periods until the end of the project. The steps of the algorithm are shown in Algorithm 4.1. There are different shifting criteria such as giving priority to the minimum shift or a random activity random amount shift.

Algorithm 4.1: Repair

//Given: an infeasible schedule with respect to the credit limit constraint

//Output: a feasible schedule with respect to the credit limit constraint

Let:

EST_p =the early start of activity p

EFT_p = the early finish of activity p

c_p = cost of activity p

Repair ()

{

Calculate the cash flow of the schedule, identify the period that violates the credit limit constraint and the "amount" of cash exceeding the limit

While (not the last period)

{

While (amount > 0)

{

Set of activities in that period ={ EST_p<end of period & EFT_p>start of period}

Calculate the minimum shift needed to improve for each activity

Calculate the maximum shift available to improve for each activity

Calculate the shift required to return to feasibility for each activity

$$Shift_p = \lceil (amount/c_p) \rceil$$

Choose one activity to shift according to the shift criterion

Update the shift vector

}

Check next period

}

4.2.2 Genetic Algorithm

In this subsection, GA will be described by explaining the different operators used as illustrated in the flow chart in Figure 4.7.

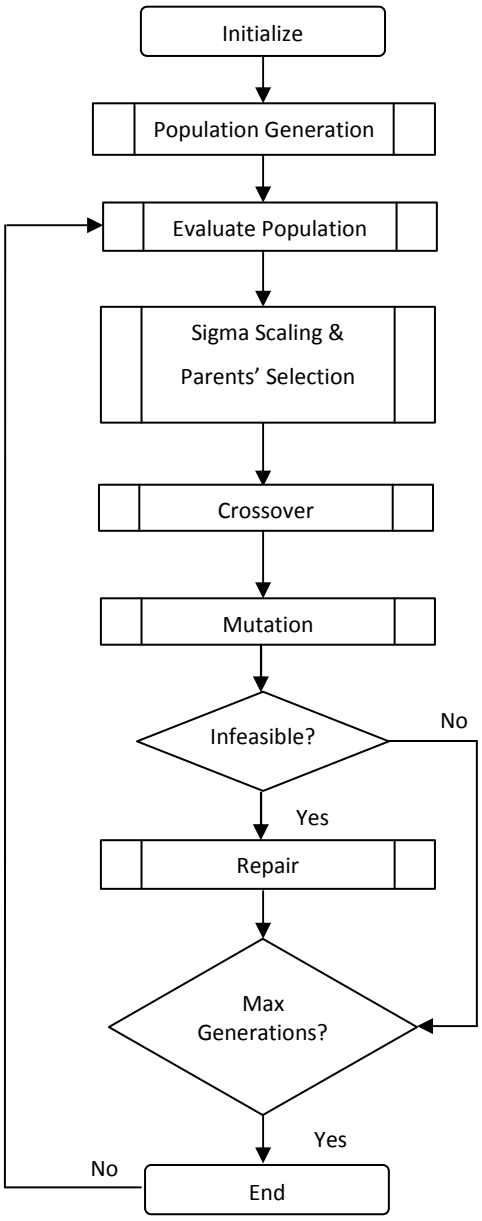


Figure 4.7: Flow chart for the GA.

Initialize:

The GA algorithm is initialized by setting its parameters such as the population size, the maximum number of generations, the probability of crossover and the probability of mutation.

Population Generation:

A population is generated using the generator presented in section 4.2.1. The number of solutions generated is equal to the population size determined in the initialization step.

Evaluate Population:

Each individual in the population is evaluated using the objective function presented in section 4.2.1.

Sigma Scaling and Parents' Selection:

The purpose of selection is to emphasize the fitter individuals in the population in hopes that their offspring will in turn have even higher fitness. Sigma scaling is a scaling mechanism that is applied to the population before the selection process. Originally, GA used fitness−proportionate selection, in which the "expected value" of an individual (i.e., the expected number of times an individual will be selected to reproduce) is that individual's fitness divided by the average fitness of the population. However, this method of selection might lead to a "premature convergence." In other words, fitness−proportionate selection early on often puts too much emphasis on "exploitation" of highly fit individuals at the expense of exploration of other regions of the search space.

Later in the search, when all individuals in the population are very similar (the fitness variance is low), there are no real fitness differences for selection to exploit, and the evolution process will stop. To address such problems, GA researchers have experimented with several "scaling "methods for mapping "raw" fitness values to expected values so as to make the GA less susceptible to premature convergence. Sigma scaling is one of these methods which keeps the selection pressure (i.e., the degree to which highly fit individuals are allowed many offspring) relatively constant over the course of the run rather than depending on the fitness variances in the population. Under sigma scaling, an individual's expected value is a function of its fitness, the population mean, and the population standard deviation

$$ExpVal(i,t) = \begin{cases} 1 + \dfrac{f(i) - \bar{f}(t)}{2\sigma(t)} & if\ \sigma(t) \neq 0 \\ 1.0 & if\ \sigma(t) = 0 \end{cases}$$

Where $ExpVal(i,t)$ is the expected value of individual i at time t, $f(i)$ is the fitness of i, $\bar{f}(t)$ and $\sigma(t)$ are the mean fitness and the standard deviation of the population at time t respectively. If the $ExpVal$ is negative, it's replaced with a value of 0.1. So that individuals with very low fitness will have slimmer chances in reproduction.

At the beginning of a run, when the standard deviation of the population's fitness is typically high, the fitter individuals will not be many standard deviations above the mean, and so they will not be allocated the lion's share of offspring. Likewise, later in the run, when the population is typically more converged and the standard deviation is typically lower, the fitter individuals will stand out more, allowing evolution to continue (Mitchell, 1998).

Crossover:

A simple one point crossover is used as explained in Figure 4.8. A pair of parents is selected using the selection method described before. The rate of crossover is determined by a crossover probability parameter which is assumed to be 1. Thus, all the population will reproduce and will be replaced by the offspring.

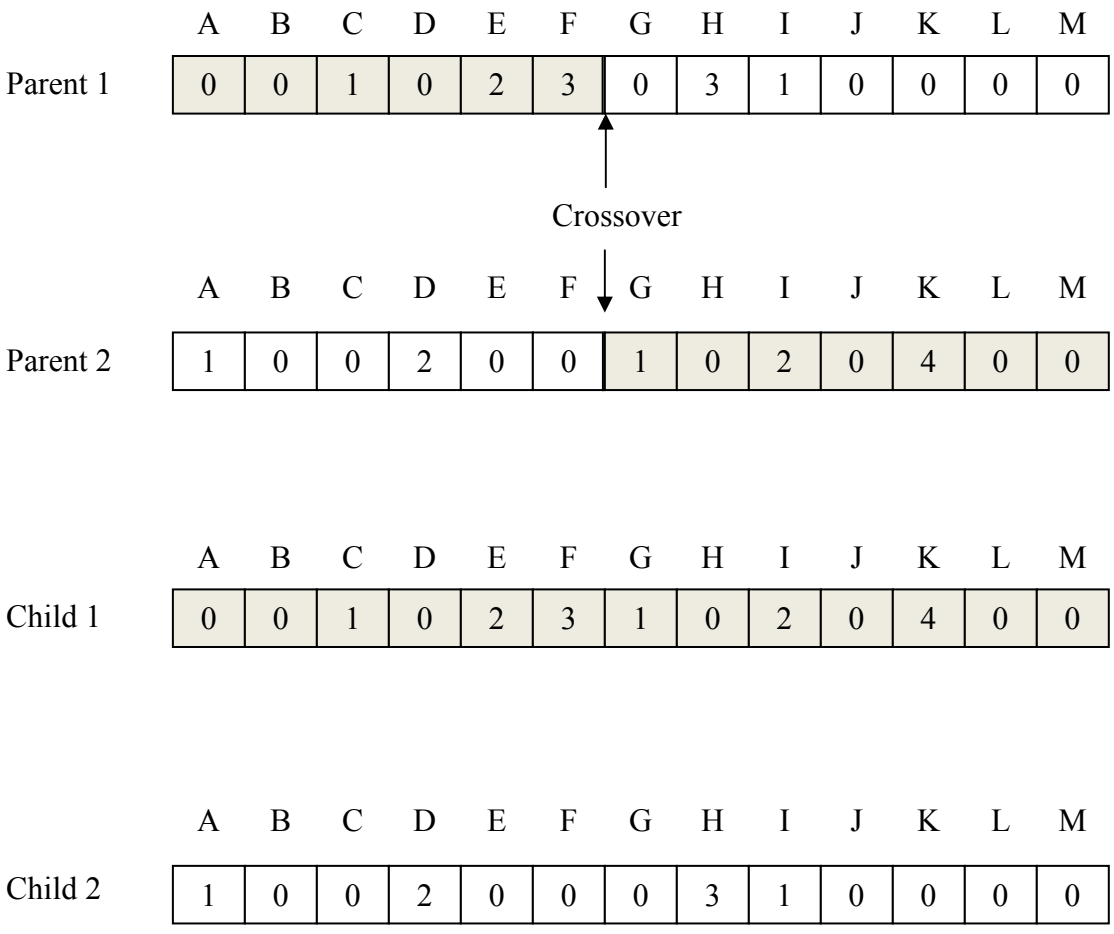


Figure 4.8: Crossover example.

Mutation:

In this step, the chromosome is mutated according to the mutation probability. The mutation (change) will affect one gene in the chromosome. This operation is necessary to keep a diverse population in order to avoid premature convergence. Figure 4.9 explains the mutation operator. Mutation is controlled using the mutation probability parameter.

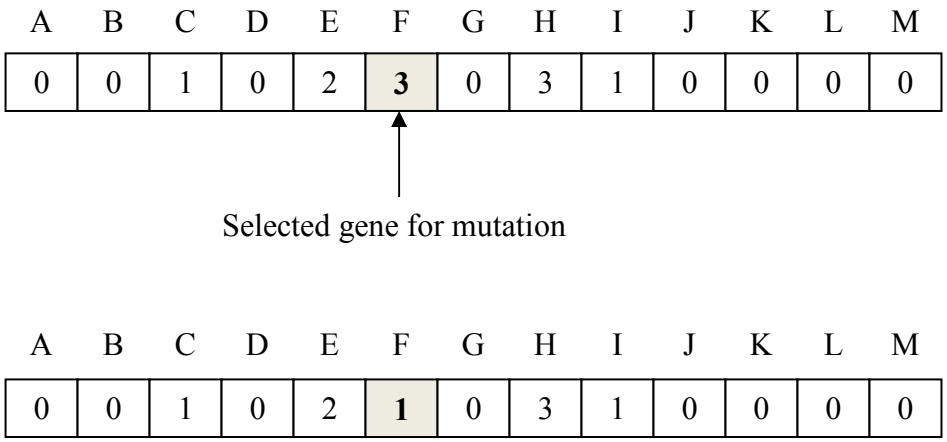


Figure 4.9: Mutation operation.

Repair:

The repair algorithm explained before is necessary to keep the population feasible with respect to the credit limit constraint. Several repair methods are proposed and it can be used interchangeably to keep the diversity of the population.

4.2.3 Shuffled Frog-Leaping Algorithm

In this subsection, the different steps of the SFLA, illustrated in Figure 4.10, are detailed.

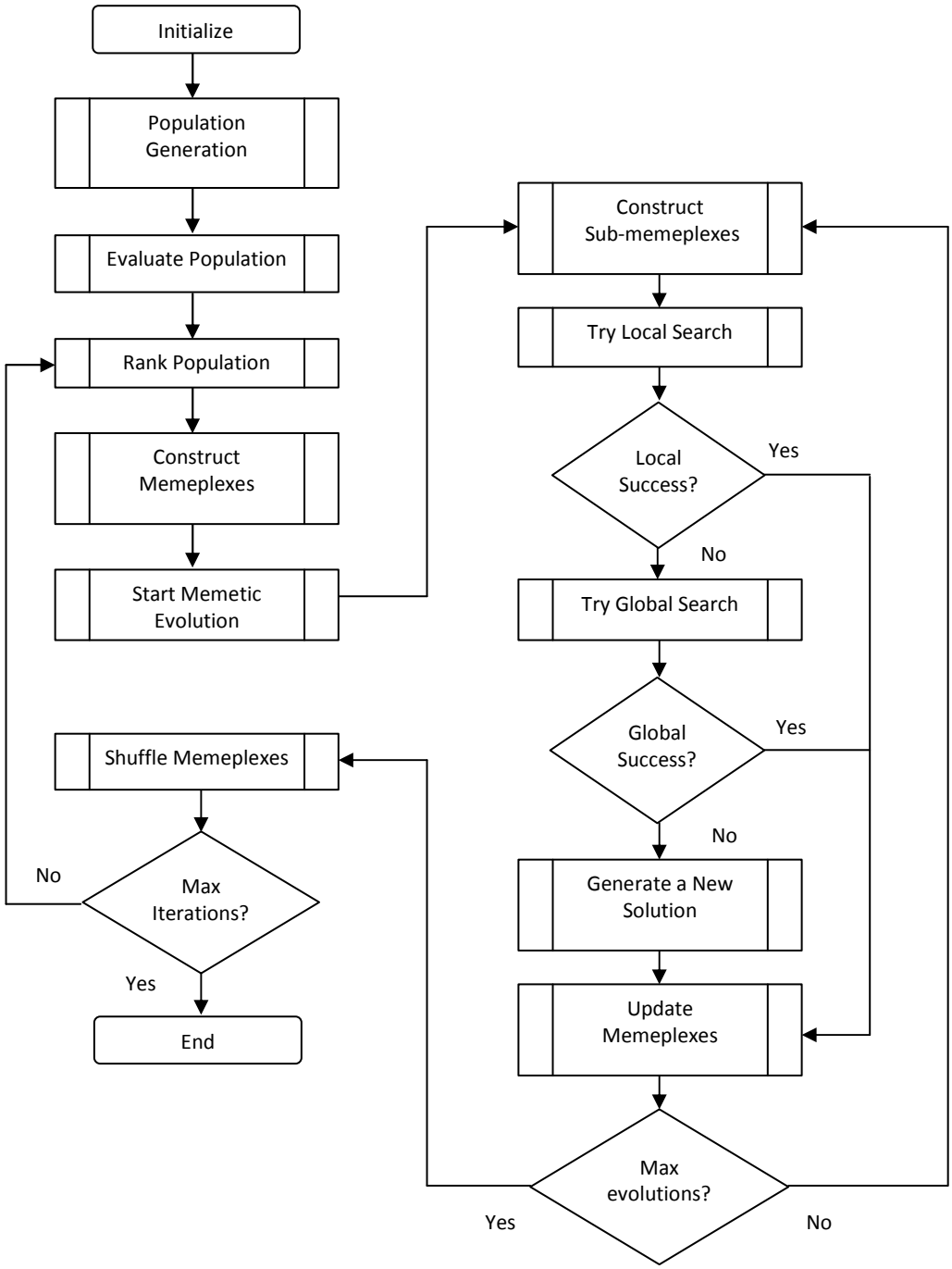


Figure 4.10: Flow chart for the SFLA.

Exploration

Step 0 **Initialize:** Set the number of memeplexes, Nm, and the number of frogs in each memeplex, Nn, the number of frogs in each sub-memeplex Nq, the number of memetic evolution steps, Ne, and the number of iterations, $iter$. The population size, F, in this case is equal to $F = Nm.Nn$.

Step 1 **Population Generation:** This step is done using the generator presented in section 4.2.1. The number of solutions generated is equal to F. I.e., $\Lambda_1, \Lambda_2, \dots, \Lambda_F$.

Step 2 **Evaluate Population:** Each frog in the population is evaluated using the objective function presented in section 4.2.1.

Step 3 **Rank Population:** Sort the F frogs in descending order of the fitness value. Record the best frog, Λ_{GB}, in the population.

Step 4 **Construct Memeplexes:** Construct memeplexes such that the ith memeplex has solutions $\Lambda_i, \Lambda_{i+Nm}, \Lambda_{i+2Nm}, \dots, \Lambda_{F-(Nm-i)}$. Figure 4.11 illustrates the memeplex construction.

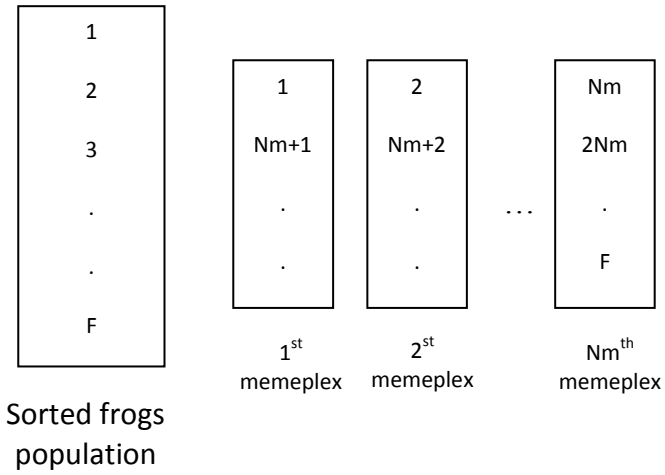


Figure 4.11: Construction of memeplexes.

Step 4 **Memetic Evolution:**

Step 4.1 **Construct Sub-memeplexes:** From each memeplex, randomly choose Nq frogs (solutions). This is accomplished by assigning a probability for each frog in the memeplex. The probability follows a triangular discrete distribution. The j^{th} frog in the memeplex will have the probability

$$P_j = 2(Nn + 1 - j)/Nn(Nn + 1) \qquad j = 1,2, \dots, Nn$$

Record the best and worst frogs (solutions) in the sub-memeplex as Λ_B and Λ_W respectively. Figure 4.12 illustrates the idea of sub-memeplex construction.

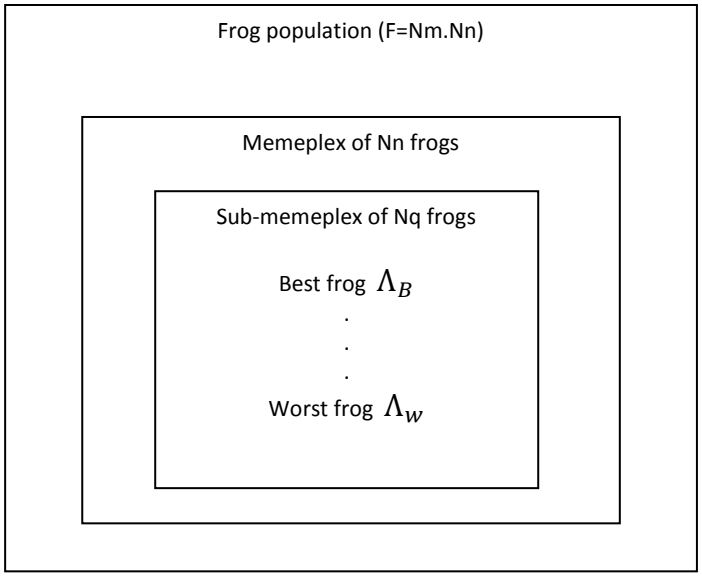


Figure 4.12: Construction of a sub-memeplex.

Step 4.2 **Try local search:** consider a temporary solution given by

$$\Lambda_{temp} = \Lambda_W + \lfloor rand.(\Lambda_B - \Lambda_W) \rfloor$$

Where $0 \leq rand \leq 1$ is a random number and $\lfloor \quad \rfloor$ is the floor function.

Evaluate the temporary solution.

Step 4.2 **Local success?** In this step we check if $f(\Lambda_{temp}) < f(\Lambda_W)$

Set $\Lambda_W = \Lambda_{temp}$ and go to step 4.6

Otherwise go to step 4.3

Step 4.3 **Try global search:** a temporary solution given by

$$\Lambda_{temp} = \Lambda_W + \lfloor rand.(\Lambda_{GB} - \Lambda_W) \rfloor$$

Step 4.4 **Global success?** In this step we check if $f(\Lambda_{temp}) < f(\Lambda_W)$

Set $\Lambda_W = \Lambda_{temp}$ and go to step 4.6

Otherwise go to step 4.5

Step 4.5 **Generate a new solution:** randomly generate a new solution to replace the worst solution Λ_W.

Step 4.6 **Update memeplexes:** update the worst solution in each memeplex.

Step 4.7 **Max evolutions?** check the if evolution counter =Ne

Go to step 5.

Otherwise go to step 4.1.

Step 5 **Shuffle Memeplexes:** merge all the memeplexes together into one pool.

Step 6 **Max Iterations?** check if the maximum number of iterations has been reached.

Stop

Otherwise go to step 3.

The mechanism of the memetic evolution in the SFLA is illustrated in Figure 4.13.

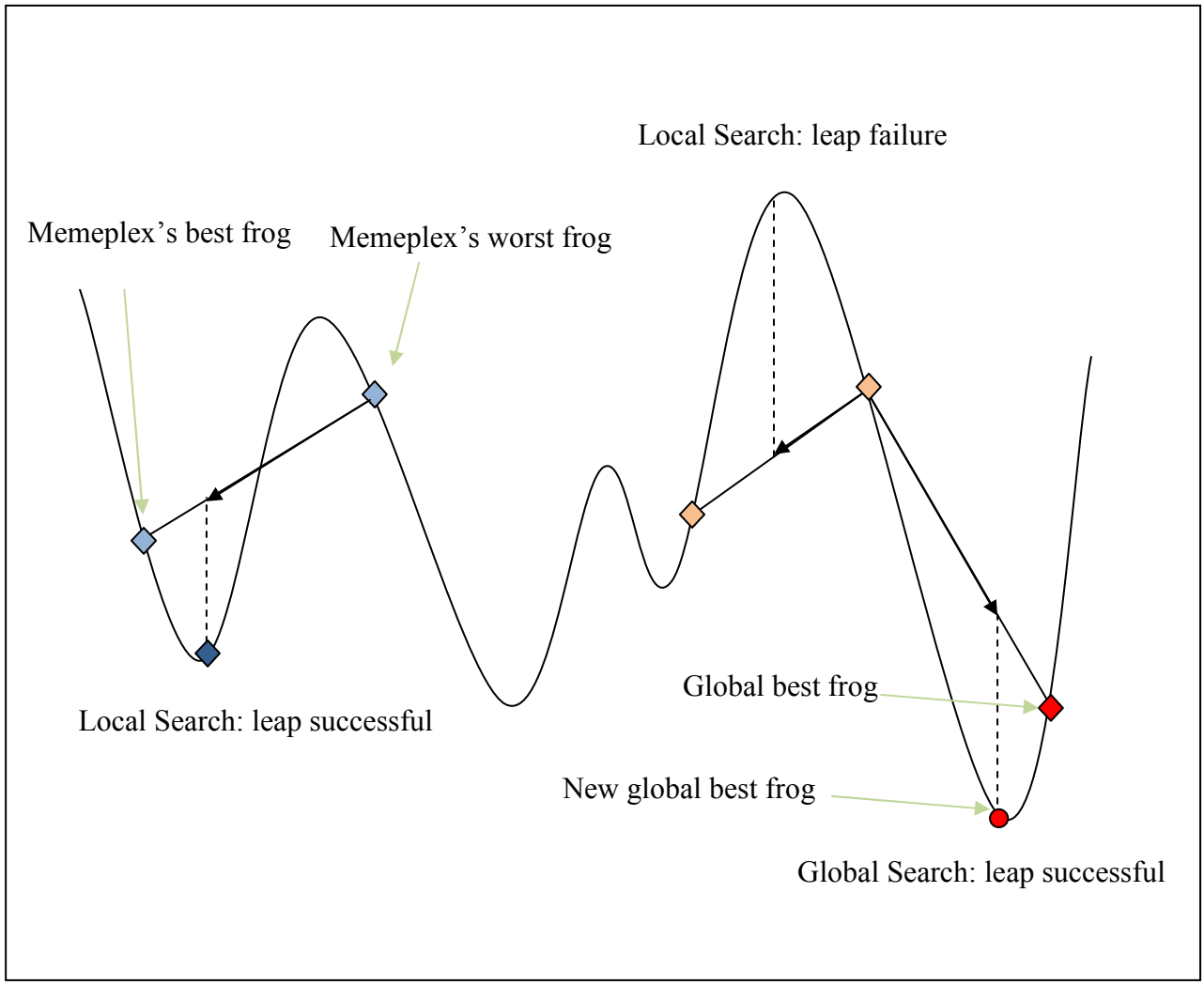


Figure 4.13: Memetic evolution in SFLA.

4.2.4 Simulated Annealing

This subsection will describe the mechanism of SA as shown in Figure 4.14.

Initialize:

Define the parameters of the SA such as the initial temperature, number of iterations per temperature and the minimum temperature allowed.

Generate Initial Solution:

One solution is generated using the generator presented in section 4.2.1.

Evaluate Solution's Energy:

The energy (fitness) of the solution is evaluated using the objective function presented in section 4.2.1.

Generate a Neighbor Solution:

A neighbor solution can be obtained from the current solution by changing some parts of the solution. This is done on the shift vector by adding a value between specified upper and lower limits for the change. It should be noted that the change should be small, thus the change shouldn't affect all activates. A pseudo code for generating a new neighbor is given in Table 4.6. An example for neighbor generation is shown in Figure 4.15.

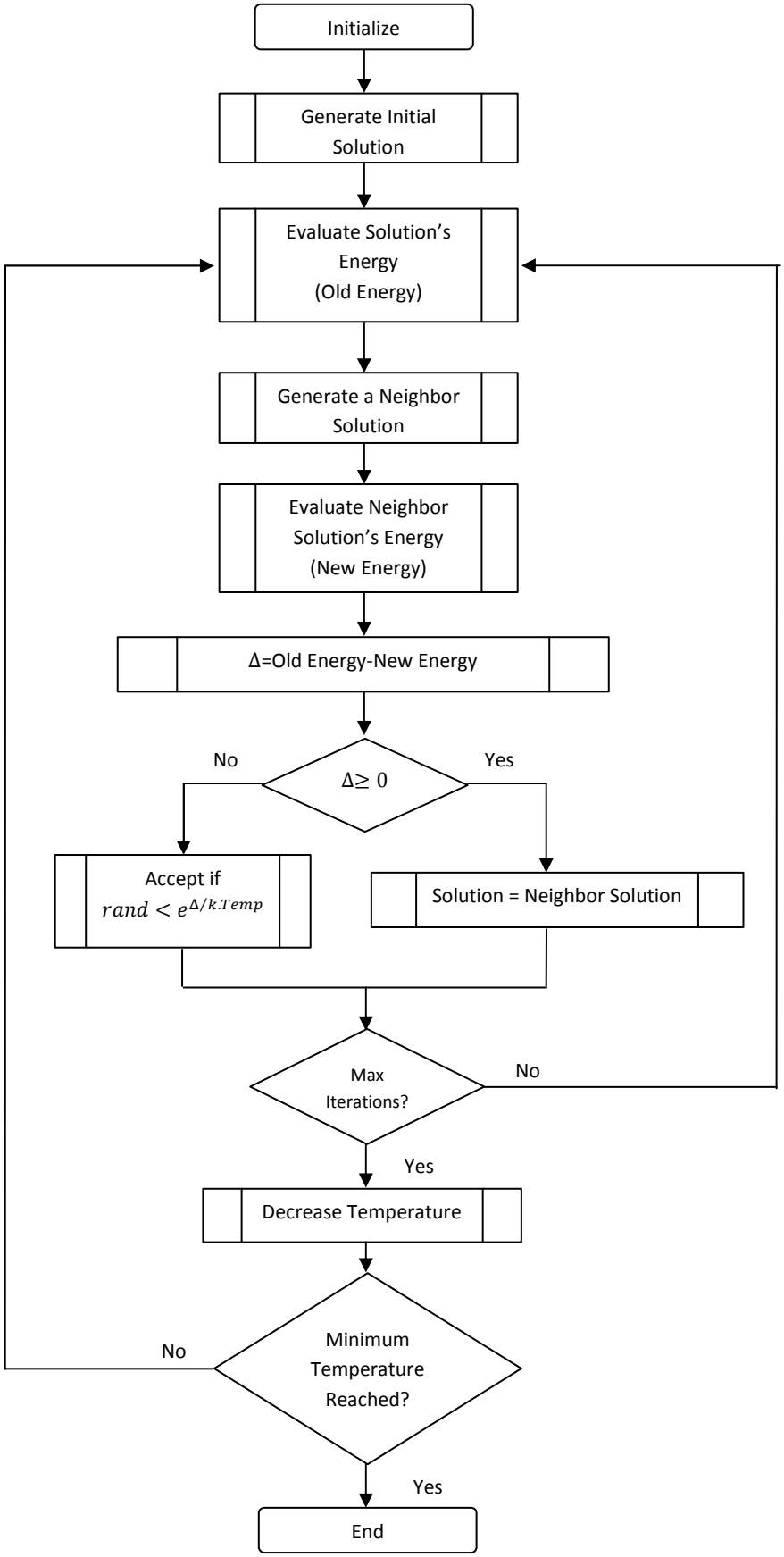


Figure 4.14: Flow chart for SA.

Table 4.6 : Pseudo code for generating a new neighbor.

Begin
Given the current solution
Given the possible allowed shifts for activities
Given the probability of each shift
Generate a random number for each activity
Using the random number, determine the amount of shift for each activity
Apply changes on the solution
Remove any negative values in the new solution
End

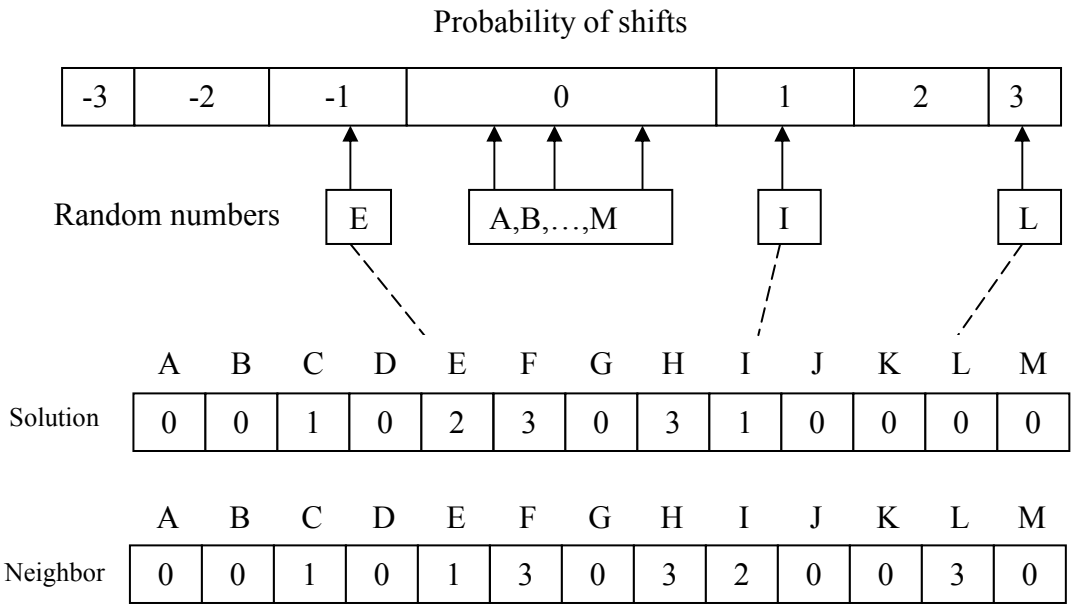


Figure 4.15: Generating a neighbor for a solution.

Decrease Temperature:

In the SA algorithm, the temperature is decreased gradually through the simulation process. The quality of the solution depends greatly on the speed of the cooling schedules. If the temperature is decreased slowly, better solutions are obtained but more computation time is needed. The most popular cooling function is the geometric cooling schedule in which the temperature is updated using the formula

$$Temp = \psi . Temp$$

Where $0 \leq \psi \leq 1$.

The process of annealing starts with the initial temperature set and the initial solution generated. The initial solution is evaluated and then a neighbor for the initial solution is generated and evaluated. If the neighbor solution has a better value according to the objective function, the neighbor will replace the initial solution. Otherwise, if the neighbor solution is not better, it will be accepted with a probability equal to $e^{\Delta/k.Temp}$. While k is the Boltzmann constant. A number of trails are made for each temperature, it can be seen that the acceptance of a worse solution depends on the current temperature. Worse solutions are accepted more at the beginning of the process when the temperature is high. After finishing the number of iterations set for a certain temperature, the temperature is cooled using the cooling schedule discussed earlier and the process of annealing goes on until the minimum temperature is reached.

Chapter 5

Multi Projects Finance Based Scheduling

Usually for a given company, contractors manage the financial aspects at the corporate level and not at the individual project's level. The contractor is generally concerned about the means of timely procuring cash for all ongoing projects. In this situation, finance based scheduling ensures that the resulting values of the negative cumulative balances of all projects do not add up to exceed the credit limit, while utilizing the positive cumulative balances that occur in some projects to execute others. Thus, concurrent projects can be related to the overall liquidity situation of contractors.

In this chapter, the approach adopted to design the heuristics is presented first. Later, multi projects case studies are presented with a solution sample.

5.1 Approach to the Multi Projects Scheduling Problem

In this section, the objective function is defined followed by the design of the GA, SA and SFLA heuristics. The representation used in the multi projects scheduling problem is presented earlier in section 4.2.1. Also, the generation of the initial solution is presented in the same section.

5.1.1 The Objective Function

In multi projects finance based scheduling, we assume that unit-price contracts between the client and the contractors charge a daily penalty on the late completion of any project. Thus, the objective of minimizing the project duration is broadened to be the profit maximization of all ongoing projects. The net profit of a single project is denoted by G, see Figure 3.1, and can be defined as the amount of positive cumulative balance at the end of the project's cash flow. The total net profit of a multi projects G_{total} can be defined as the total positive cumulative balance at the end of total cash flow of the multi projects. Using the penalization strategy, the objective function (fitness function) will be

$f(S) = G_{total} - \Phi\{max(0, TB_{max} - W_{total})\}$

Where Φ is a penalty factor, $TB_{max}$is the multi projects' total maximum negative cash flow and $W_{total}$is the specified multi projects' credit limit.

5.1.2 Genetic Algorithm

The GA used for the multi projects problem is similar to the one presented in section 4.2.2. However, the algorithm is modified to take care of multiple projects by generating a population for each project. For example, if we have two projects, two populations need to be generated. The fitness of the solution depends on both projects as G_{total} is the total net profit of both projects. This profit can be calculated by combining the cash flow of ith individual in the 1st population with the ith individual in the 2nd population. Selection is done based on the fitness value after applying the sigma scaling mechanism to it. The reproduction and mutation operators are applied for each population.

5.1.3 Shuffled Frog-Leaping Algorithm

The SFLA used here is based on the one presented earlier in section 4.2.3. Similar to the GA, the modification applied to the SFLA is to use multiple frog populations instead of one. Each population will have its own memeplexes and sub-memeplexes. The evaluation is done based on both populations and each population is sorted according to the evaluated fitness. The final solution will consist of the best frog from each population which is the first frog after the stopping criteria is reached.

5.1.4 Simulated Annealing

Muti projects SA is based on the one presented in section 4.2.4. An initial shift vector is generated for each project. The energy (solution) is evaluated based on both initial shift vectors. Then, a neighbor is generated for the shift vector of each project and the energy of the neighbors is evaluated. The solution is accepted or rejected according to the steps explained earlier.

5.2 Case Studies

In this section, two multi projects' case studies are given. The first case study is for a contractor executing two concurrent projects, the first is a 25-activity project and the second is a 30-activity project. In the second case study, the concurrent projects consist of 125 and 120 activities respectively. The activity networks for the first case study are shown in Figure 5.1 and Figure 5.2. The networks for the 125 and 120 activities are a repeated version of the 25 and 30 networks respectively. The financial data and the contractual terms of the four projects are shown in Table 5.1. An example on the financial calculations for the rates of the cash outflows and inflows is given in Table 5.2. This information is input to the program and all the cash flow calculations are based on it. A solution for the 25-30 multi-projects case, represented by schedules obtained as an output from the program, at a credit limit of 75,000 is presented in Figure 5.3 and Figure 5.4. In addition, the weekly expenditure and income for both of the projects are presented in Table 5.3. Finally, cash flow details for each project are presented in Tables 5.4, 5.5 and are illustrated in Figures 5.5, 5.6. The total cash flow details of both projects are presented in Table 5.6 and are illustrated in Figure 5.7. Further results and discussion are presented in the following chapter.

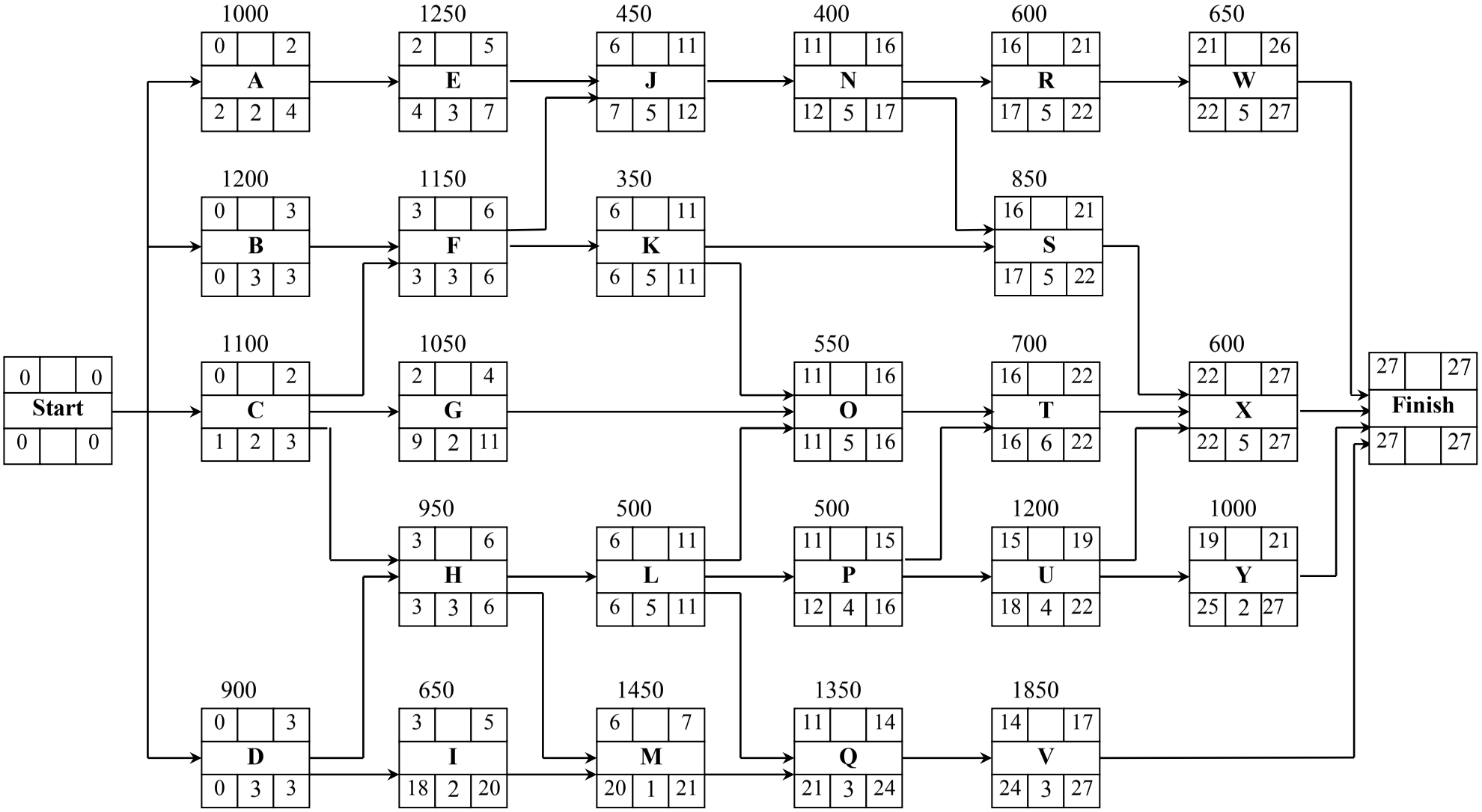


Figure 5.1: Activity network for the 25-activity problem.

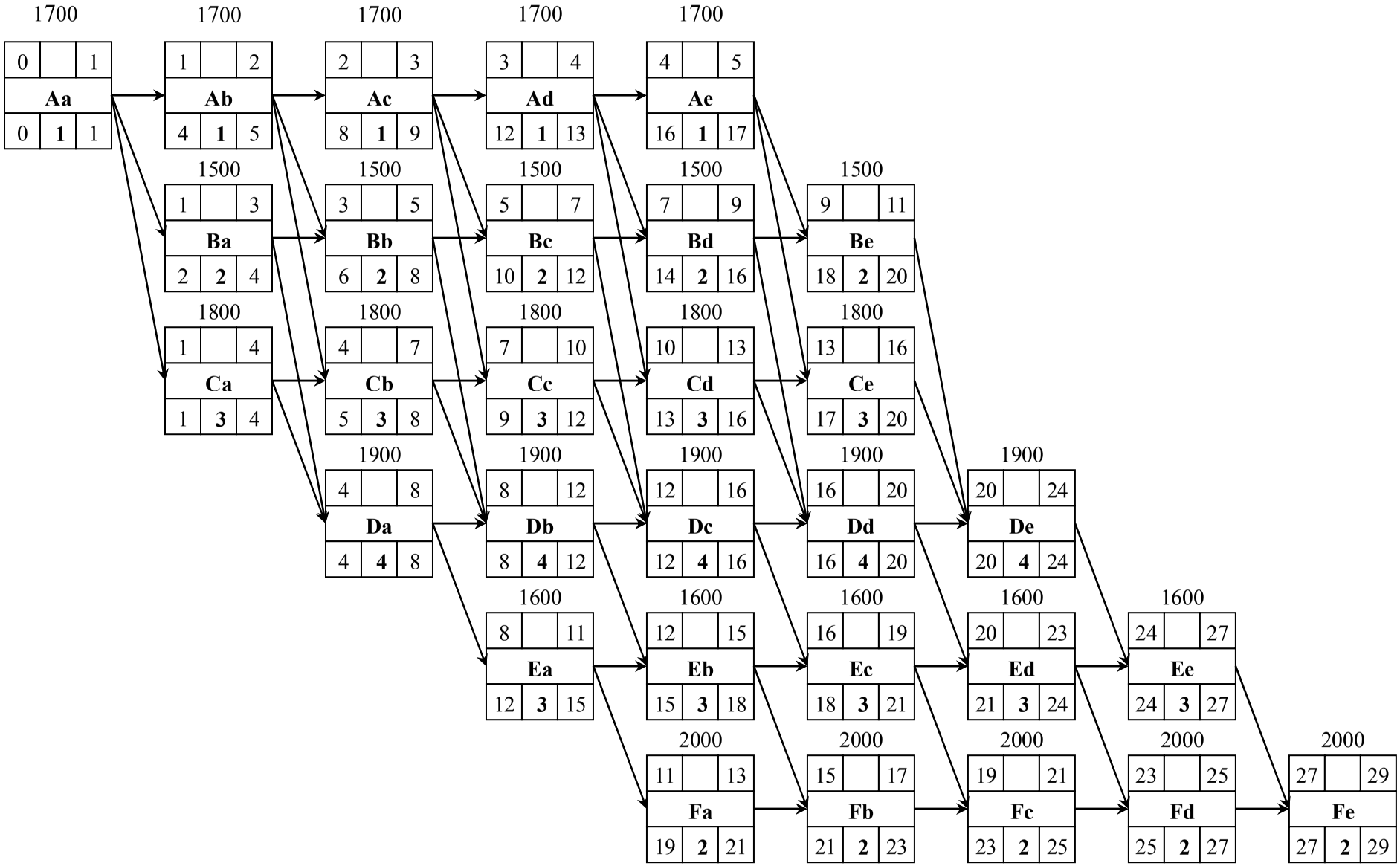


Figure 5.2: Activity network for the 30-activity project.

Table 5.1: The financial data and the contractual terms of the four projects.

Category	Item	25-activity project	30-activity project	125-activity project	120-activity project
Interest rate	Interest rate percentage per week	0.8	0.8	0.8	0.8
Financial data	Original duration (days)	27	29	135	89
	Original duration (Weeks)	6	6	27	18
	Overheads percentage	17	15	15	15
	Mobilization costs percentage	8	5	10	10
	Tax percentage	2	2	2	2
	Markup percentage	12	20	10	10
	Bond premium percentage	4	1	1	1
Contract terms	Advance payment percentage of total bid price	9	10	2	2
	Weeks to retrieve advance payment	a	a	a	a
	Retained percentage of pay requests	6	5	5	5
	Lag to pay retained money after last payment (Weeks)	0	0	0	0
	Weeks to submit pay requests regularly	1	1	1	1
	Lag to pay payment requests (weeks)	1	1	1	1
	Late completion penalty per day	1500	1000	1000	1000

a: Number of weeks encompassing the total project duration.

Table 5.2: Factor calculations for the 30-activity project.

Activity	Duration in days	Direct Cost per day	Total direct cost	Price per day
Aa:Ae	1	1700	1700	2537.69
Ba:Be	2	1500	3000	2239.14
Ca:Ce	3	1800	5400	2686.97
Da:De	4	1900	7600	2836.24
Ea:Ee	3	1600	4800	2388.42
Fa:Fe	2	2000	4000	2985.52
Note: The prices in this table do not include the financing cost				

Total cash outflow = 132,500;

Overheads = 19,875;

Mobilization costs = 7,618.8;

Cash outflow + Overheads + Mobilization = 159,993.8;

Taxes = 3199.9

Taxes + cash outflow + Overheads + Mobilization = 163,193.7;

Bond Premium = 1,958.3;

Total Bid Price = 197,790.7;

Factor to determine price based on cash outflow (197,790.7÷132,500)=1.4927

Activity	Cost per day	Price per day	Days								
			5	10	15	20	25	30	35	40	
A	1000	1501.3									
B	1200	1801.5									
C	1100	1651.4									
D	900	1351.2									
E	1250	1876.6									
F	1150	1726.5									
G	1050	1576.3									
H	950	1426.2									
I	650	975.8									
J	450	675.6									
K	350	525.4									
L	500	750.6									
M	1450	2176.9									
N	400	600.5									
O	550	825.7									
P	500	750.6									
Q	1350	2026.7									
R	600	900.8									
S	850	1276.1									
T	700	1050.9									
U	1200	1801.5									
V	1850	2777.4									
U	650	975.8									
X	600	900.8									
Y	1000	1501.3									
Direct cost per week			21850	8750	11150	14350	12950	3700	--	--	--
Earned value per week			24480.7	11380.7	13780.7	16980.7	15580.7	4752.4	--	--	--

Figure 5.3 : A solution for the 25-activity project at a shift of two weeks and a credit limit of 75,000.

Activity	Cost per day	Price per day	Days							
			5	10	15	20	25	30	35	40
Aa	1700	2537.7								
Ab	1700	2537.7								
Ba	1500	2239.1								
Ca	1800	2686.9								
Ac	1700	2537.7								
Bb	1500	2239.1								
Cb	1800	2686.9								
Da	1900	2836.2								
Ad	1700	2537.7								
Bc	1500	2239.1								
Cc	1800	2686.9								
Db	1900	2836.2								
Ea	1600	2388.4								
Ae	1700	2537.7								
Bd	1500	2239.1								
Cd	1800	2686.9								
Dc	1900	2836.2								
Eb	1600	2388.4								
Fa	2000	2985.5								
Be	1500	2239.1								
Ce	1800	2686.9								
Dd	1900	2836.2								
Ec	1600	2388.4								
Fb	2000	2985.5								
De	1900	2836.2								
Ed	1600	2388.4								
Fc	2000	2985.5								
Ee	1600	2388.4								
Fd	2000	2985.5								
Fe	2000	2985.5								
Direct cost per week			--	--	23600	24700	24300	28700	20000	11200
Earned value per week			--	--	27578.4	28678.4	28278.4	32678.4	23978.4	14382.7

Figure 5.4: A solution for the 30-activity project at a shift of two weeks and a credit limit of 75,000.

Table 5.3: Weekly expenditure and income of the individual and combined projects.

End of Week	Expenditures	Amount		Sum for two projects	Income	Amount		Sum for two projects
		25-Act. Project	30-Act. Project			25-Act. Project	30-Act. Project	
0	Mobilization & bond	11010.1	---	11010.1	Advance payment	9829.6	--	9829.6
1	Direct cost	21850	--	--	Earned value	--	--	--
	Overhead and Tax	2620.2[a]	--	--	Deductions	--	--	--
	Total	24470.2	--	24480.7	Net	--	--	--
2	Direct cost	8750	--	--	Earned value	32803.4	--	--
	Overhead and Tax	2630.7[a]	--	--	Deductions	3606.4[c]	--	--
	Total	11380.7	9577.1[g]	20957.8	Net	29196.9	19779.1[h]	48976
3	Direct cost	11150	23600	--	Earned value	13136.3	--	--
	Overhead and Tax	2630.7[a]	3978.4[b]	--	Deductions	2426.4	--	--
	Total	13780.7	27578.4	41359.1	Net	10709.9	--	10709.9
4	Direct cost	14350	24700	--	Earned value	16739.4	35229.1	--
	Overhead and Tax	2630.7[a]	3978.4[b]	--	Deductions	2642.6	5057.9[d]	--
	Total	16980.7	28678.4	45659.1	Net	14096.8	30171.1	44268
5	Direct cost	12950	24300	--	Earned value	21543.6	36871.1	--
	Overhead and Tax	2630.7[a]	3978.4[b]	--	Deductions	2930.8	5140	--
	Total	15580.7	28278.4	43859.1	Net	18612.7	31731	50343.8
6	Direct cost	3700	28700	--	Earned value	19441.8	36274	--
	Overhead and Tax	1052.3[a]	3978.4[b]	--	Deductions	2804.7	5110.2	--
	Total	4752.4	32678.4	37430.7	Net	16637	31163.8	47800.9
7	Direct cost	--	20000	--	Earned value	5554.8	42842.2	--
	Overhead and Tax	--	3978.4[b]	--	Deductions	1971.5	5438.6	--
	Total	--	23978.4	23978.4	Additions	6553. 1[e]	--	--
					Net	10136.3	37403.5	47539.9
8	Direct cost	--	11200	--	Earned value	--	29855.2	--
	Overhead and Tax	--	3182.7[b]	--	Deductions	--	4789.2	--
	Total	--	14382.7	14382.7	Net	--	25065.9	25065.9
9	Direct cost	--	--	--	Earned value	--	16718.9	--
	Overhead and Tax	--	--	--	Deductions	--	4132.4	--
	Total	--	--	--	Additions	--	9889. 5[f]	--
					Net	--	22475.9	22475.9

a: Overheads/day = 12367.5/27=458.1; Tax/day = 1838.54/27 = 68.09;
b: Overhead/day = 19875/29=685.34; Tax/day = 3199.88 /29 = 110.34;
c: Retained percentage and advance payment retrieval = 28674.4×0.06 + 9829.62/6 = 3358.7;
d: Retained percentage and advance payment retrieval = 30153.8×0.05 + 19779.07/7 = 4333.3;
e: Paying the retained money back = 109217.98 × 0.06 = 6553.1;
f: Paying the retained money back = 197790.68 × 0.05 = 9889.5;
g: Mobilization and bond costs for the 30-activity project.
h: Advance payment of the 30-activity project.

Table 5.4: The cash flow parameters of the 25-activity project.

Financial parameter	Weeks							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Expenditure *ET*	-11,010.1	-24,480.7	-11,380.7	-13,780.7	-16,980.7	-15,580.7	-4,752.3	0
Financing costs *I*	0	101	-262.3	-124.7	-164.1	-178.5	-136.4	0
Cumulative balance *B*	-11,010.1	-25,762.2	-37,405.3	-22,113.9	-28,548.9	-30,211.4	-16,487.3	149.6
Income *P*	9,829.6	0	29,196.9	10,709.9	14,096.8	18,612.7	16,637.0	10,136.3
Net balance *N*	-1,180.4	-25,762.2	-8,208.3	-11,404	-14,452.1	-11,598.6	149.6	10,285.9

Table 5.5: The cash flow parameters of the 30-activity project.

Financial parameter	Weeks									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Expenditure *ET*	--	--	-9,577.1	-27,578.4	-28,678.4	-28,278.4	-32,678.4	-23,978.4	-14,382.7	0
Financing costs *I*	--	--	0	-46.3	-262.3	-246.8	-224.4	-105	-238.9	0
Cumulative balance *B*	--	--	-9,577.1	-17,422.7	-46,363.4	-44,717.6	-45,889.3	-38,942.8	-16,027	9,038.8
Income *P*	--	--	19,779.1	0	30,171.1	31,731	31,163.8	37,403.5	25,065.9	22,475.9
Net balance *N*	--	--	10,202	-17,422.7	-16,192.3	-12,986.5	-14,725.5	-1,539.3	9,038.8	31,514.8

Table 5.6: The cash flow parameters of the two projects.

Financial parameter	Weeks									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Expenditure *ET*	-11,010.1	-24,480.7	-20,957.8	-41,359.1	-45,659.1	-43,859.1	-37,430.7	-23,978.4	-14,382.7	0
Financing costs *I*	0	101	-262.3	-171.1	-426.4	-425.4	-360.8	-238.9	-105	0
Cumulative balance *B*	-11,010.1	-25,762.2	-37,405.3	-39,536.7	-74,912.4	-74,929	-62,376.7	-38,793.2	-5,741.05	19,324.8
Income *P*	9,829.6	0	48,976	10,709.9	44,268	50,343.8	47,800.9	47,539.9	25,065.9	22,475.9
Net balance *N*	-1,180.4	-25,762.2	1,993.6	-28,826.7	-30,644.4	-24,585.1	-14,575.8	8,746.6	19,324.8	41,800.85

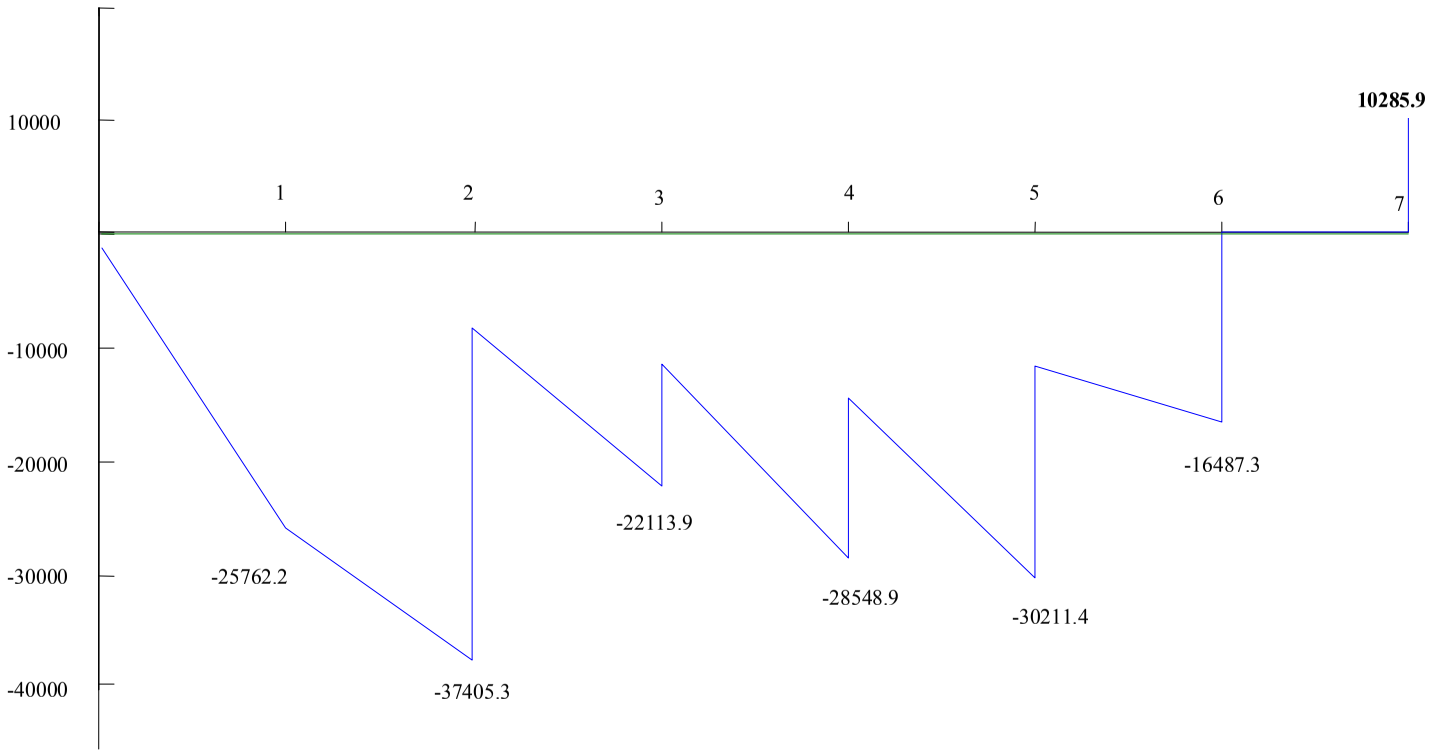


Figure 5.5: Cash flow of the 25-activity project.

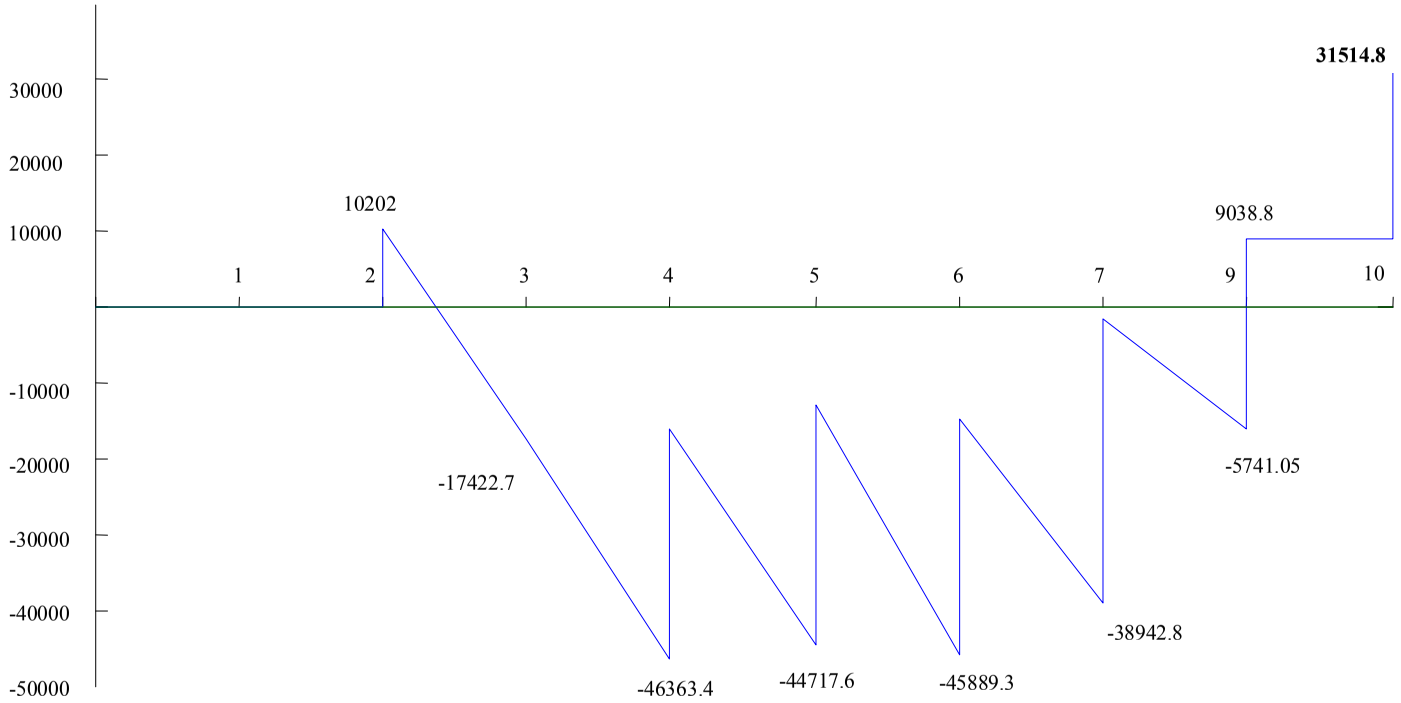


Figure 5.6: Cash flow of the 30-activity project.

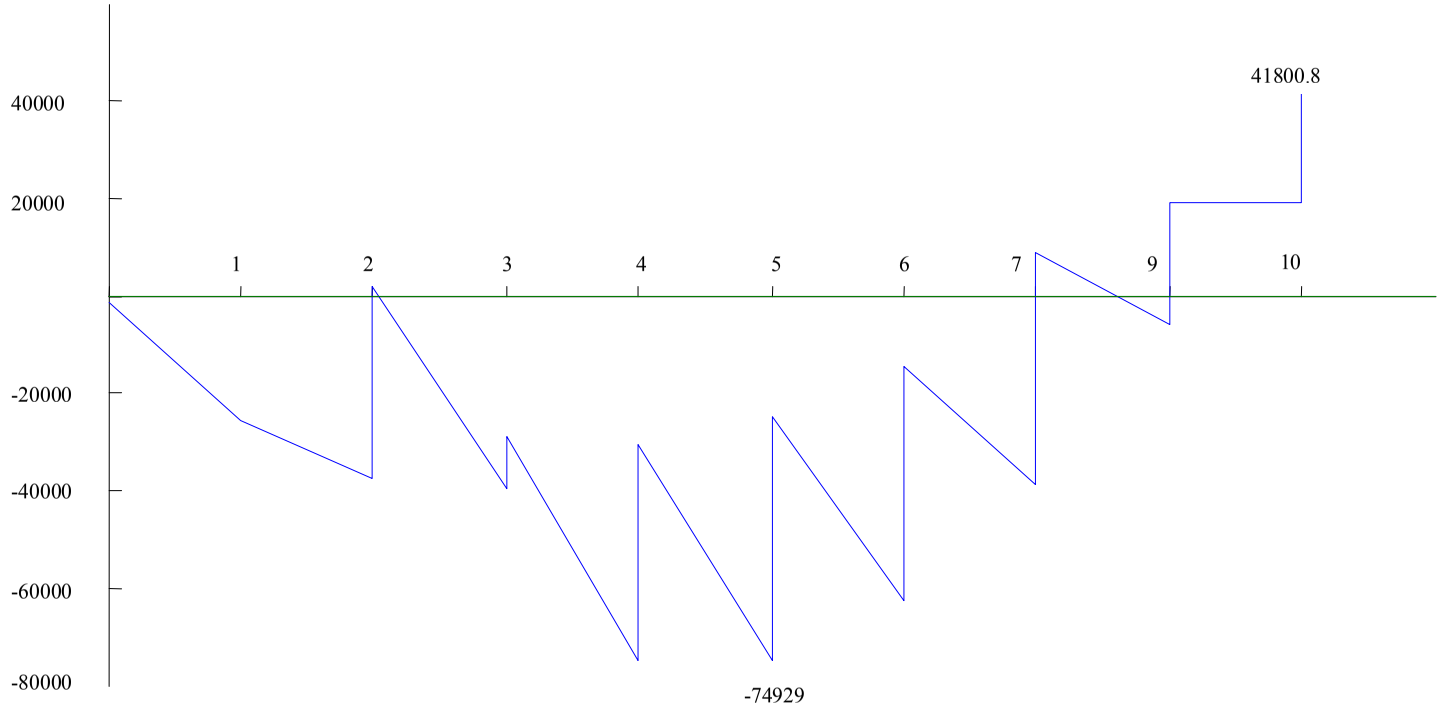


Figure 5.7: Total cash flow of the two projects.

Chapter 6

Results and Discussion

In this chapter we report our computational experience and benchmark comparisons between the solution methods discussed in Chapter 4. The solutions reported are obtained using integer programming, genetic algorithm, genetic algorithm with repair, shuffled frog leaping algorithm, shuffled frog leaping algorithm with repair, simulated annealing and simulated annealing with repair.

The optimal solutions obtained using the integer program are solved using Lingo 10 optimization package from LINDO systems. In addition, the results of the meta-heuristics are obtained by solving the problems coded on Matlab 7.4.0. All experiments took place on a 2 GHz intel machine with 1GB of RAM. The benchmark comparisons among solution methods are presented for both the single project and multi projects problems. Single project example networks consist of five 30-activity networks presented in Appendix 1. In the single project finance based scheduling problem, contracts are assumed to follow the unit-price contracts with a 10% overhead percentage, 20% markup percentage and a 10% retainage. The payments are assumed to have a one week lag. On the other hand, the multi projects example networks are illustrated in Chapter 5. The contracts are assumed to follow the cost-plus contracts with the contractual terms presented in Table 5.1.

6.1 Meta-heuristics' Parameters

In order to obtain the most suitable parameter values that suit our problem, a large number of experiments for each meta-heuristic were conducted. The final parameter values adopted for each meta-heuristic are given next.

Genetic Algorithm

For the single project problem, the population size was set to 500. The crossover and mutation probabilities were set to 1 and 0.2 respectively. The evolutionary process was kept running until either the optimal solution of the IP is reached or until there are no improvement in 10 consecutive generations whichever occurs first. The maximum number of generations was set to 100. For the multi projects problem, the population size was increased to 2000. The crossover and mutation probabilities were set to 1 and 0.05 respectively. The number of generations was increased to 1000.

Shuffled Frog Leaping Algorithm

For the single project problem, the population size was set to 600. The memeplex and submemeplex sizes were set to 40 and 20 respectively. The number of evolutions per iteration was set to 15 and the maximum number of iterations was 100. For the multi projects problem, the population size was increased to 4000 and the number of iterations was set to 150.

Simulated Annealing

For the single project problem, The initial temperature was set to 1 and the cooling schedule factor, ψ,was set to 0.95. The Boltzmann constant was set to 1 and the iterations per temperature were 600. The minimum temperature was set to 0.0001. For the multi projects problem, The initial temperature was set to 1 and the cooling schedule factor, ψ,was set to 0.98. The Boltzmann constant was set to 2000 and the iterations per temperature were 600. The minimum temperature was set to 0.0001.

6.2 Comparisons and Discussion

The results obtained after solving the single project problems using the exact and heuristic solutions, with and without repair, are presented in this section. Later, the multi projects results using the heuristic solutions are presented. The heuristic results that are presented in this section are the best results obtained out of ten runs. All problems were solved using two different credit limits. Each problem was solved ten times and performance measures were recorded. The performance comparison between the meta-heuristics are based on three criteria: (1) the average and standard deviation of the solution; (2) the percentage deviation of the average from the optimal solution obtained from the integer program; and (3) the average processing time. In all experiments, the heuristic is stopped when the optimal solution is reached or when the terminating criteria of the heuristic is satisfied whichever comes first. (Elbeltagi, Hegazy & Grierson, 2005)

For the single project problem, the objective was to minimize the total duration of the project under a given credit limit. Table 6.1 shows the optimal results obtained from the integer program along with the heuristics' best results out of ten runs. It can be seen that for Networks 4 and 5 all heuristics have performed well and were able to reach the optimal solution. However, not all were able to reach the optimal for the rest of the networks. Further details of the solutions are summarized in Table 6.2 and 6.3 for solutions obtained without repair and with repair respectively. In each table, details of the solutions obtained using the constrained credit limit is presented including the different performance measures. Figure 6.1 shows the percentage deviation from the optimal solution for problems solved without repair. As shown in Figure 6.1, SFLA outperformed GA when it comes to solution quality but SA was the best among all of the three. The average processing time for each meta-heuristics is shown in Figure 6.2. The best heuristic in terms in solution quality, SA, is the slowest in the speed of convergence. This might be attributed to the mechanism of how the SA works. Furthermore, SFLA outperformed the GA in speed. When the repair algorithm is used with these heuristics, the solution quality improves substantially as all heuristics reached the optimal solution. Figure 6.3 shows the average processing time for the heuristics when the repair algorithm is used. In four of the five networks, the processing time of the GA was the least. It should be noted that the repair algorithm also helps in reaching the optimal solution regardless of the quality of the initial population. This is illustrated by an example on Network 4, using GA and GA-R, in Table 6.4. Without using the repair algorithm, the percent deviation from the optimal increases as the additional shift units, added to the

shift vector that is used to generate the initial population, are decreased from 5 to 0. However, the solution quality remains the same when the repair algorithm is used.

The results obtained for the multi projects problem are summarized in Table 6.5. The objective in this problem was to maximize the total profit of the combined projects while keeping the total maximum negative cash flow of both projects within the specified credit limit. Detailed results are presented in Table 6.6. According to the deviation percentages shown in Figure 6.4, SFLA was better than GA in the case of 125-120 projects while it was a little worse for the smaller problem of the 25-30 projects. Surprisingly, SA outperformed both the GA and the SFLA in both the quality of the solution and the processing time. The processing time for each case is shown in Figure 6.5. It should be noted that when the size of the problem gets larger, the solution quality gets affected as the average deviation of the solution becomes poorer when compared to smaller problems and the processing time required becomes larger.

It's interesting to observe that the performance of all heuristics is almost consistent among all problems. Despite being an old heuristic, SA outperformed both of the other heuristics for the cases of single and multi projects in solution quality. This shows that SA is still a great tool to solve problems with a similar structure as such heuristic exploits the structure of the problem as opposed to population based heuristics which tend to be explorative rather than exploitative. In addition, it can be concluded that the repair algorithm is important for both the GA and the SFLA as it improves the solution quality drastically. This improvement however will cost more computation time. The repair algorithm replaced the old method of discarding infeasible solutions by adding any required shifts to repair any financial infeasibility.

Table 6.1: Results of the single project problems.

Network	Original Duration* (Days)	Original Max. Negative Balance* $	Credit Limits $	Total Duration						
				IP	GA	GA-R**	SFLA	SFLA-R**	SA	SA-R**
1	43	70,197	45,000	45	46	45	46	45	45	45
			40,000	48	49	48	49	48	48	48
2	25	78,804	52,000	29	30	29	30	30	29	29
			42,000	35	36	35	35	35	35	35
3	29	63,805	49,000	30	31	30	31	30	30	30
			40,000	36	36	36	36	36	36	36
4	29	34,964	27,000	32	32	32	32	32	32	32
			25,000	34	34	34	34	34	34	34
5	23	60,345	46,000	25	25	25	25	25	25	25
			40,000	27	27	27	27	27	27	27

* Without Credit Limit

** With Repair

Table 6.2: Details of results obtained for single project problems without using repair.

Network	C.L.	IP Sol.	GA				SFLA				SA			
			Best Sol.	Average*	Dev. (%)	Avg. Time	Best Sol.	Average*	Dev. (%)	Avg. Time	Best Sol.	Average*	Dev. (%)	Avg. Time
1	40000	48	49	50.2 (0.63)	4.6	32.5	49	49.6 (0.97)	3.33	8.4	48	48.8 (0.42)	1.67	34.7
2	42000	35	36	36.2 (0.44)	3.5	19.5	35	35.9 (0.33)	2.54	4.9	35	35 (0)	0	23.8
3	40000	36	36	36.1 (0.33)	0.3	12.4	36	36.1 (0.33)	0.3	4.4	36	36 (0)	0	14.6
4	25000	34	34	34.7 (0.48)	2.1	12.5	34	34.3 (0.48)	0.9	5.1	34	34 (0)	0	14.2
5	40000	27	27	28.1 (0.74)	4.1	11.6	27	27.8 (0.42)	2.9	4.2	27	27 (0)	0	14.4

* Average (Standard Deviation)

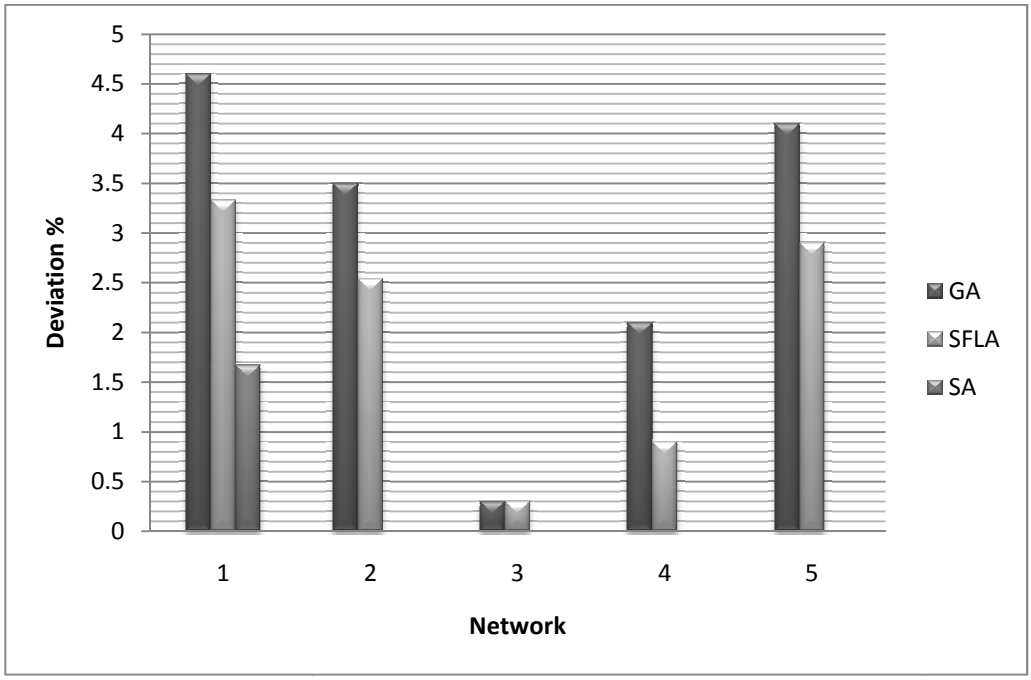


Figure 6.1: Percent deviation from the optimal solution (Single project problems without repair).

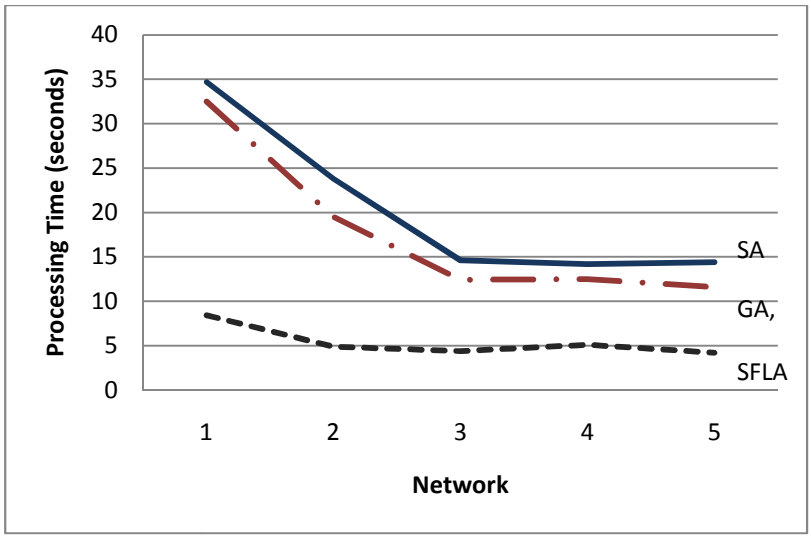


Figure 6.2: Average processing time (Single project problems without repair).

Table 6.3: Details of results obtained for single project problems using repair.

Network	C.L.	IP Sol.	GA-R				SFLA-R				SA-R			
			Best Sol.	Average*	Dev. (%)	Avg. Time	Best Sol.	Average*	Dev. (%)	Avg. Time	Best Sol.	Average*	Dev. (%)	Avg. Time
1	40000	48	48	48.6 (0.52)	1.25	48.5	48	48.5 (0.53)	1	80.9	48	48 (0)	0	61.4
2	42000	35	35	35 (0)	0	35.6	35	35 (0)	0	41.4	35	35 (0)	0	53
3	40000	36	36	36 (0)	0	10.2	36	36 (0)	0	6.8	36	36 (0)	0	29
4	25000	34	34	34 (0)	0	15.6	34	34 (0)	0	18.5	34	34 (0)	0	31.7
5	40000	27	27	27 (0)	0	16.8	27	27 (0)	0	19.2	27	27 (0)	0	27

* Average (Standard Deviation)

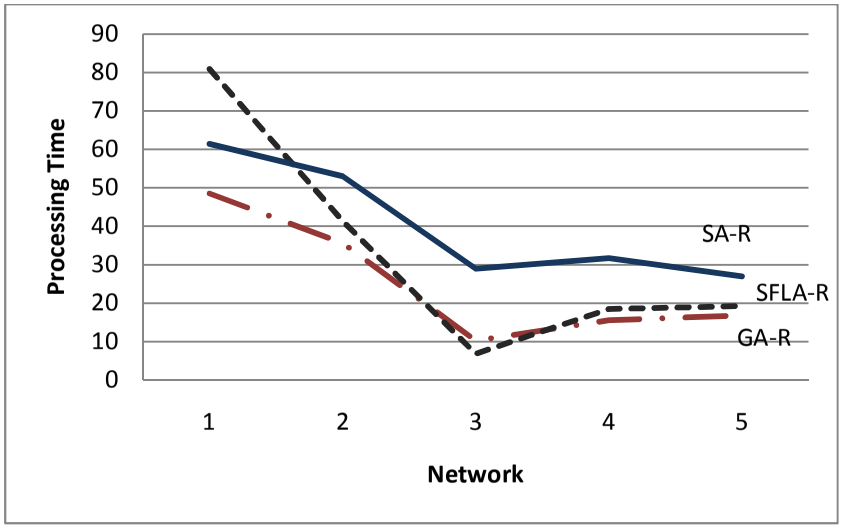


Figure 6.3: Average processing time (Single project problems with repair).

Table 6.4: Sensitivity of the GA & GA-R to the initial population in network 4.

		GA	GA-R
Additional Shift Units=5	**Average Solution (Days)**	34.7	34
	Average Deviation from Optimal (%)	2.1	0
Additional Shift Units=0	**Average Solution (Days)**	39.1	34
	Average Deviation from Optimal (%)	15	0

Table 6.5: Results of the multi projects problems (without repair).

Case* (Credit Limit)	Original Max. Negative Balance $	Original Duration (Days)	GA			SFLA			SA		
			Total Profit $	Max Negative Balance $	Duration	Total Profit $	Max Negative Balance $	Duration	Total Profit $	Max Negative Balance $	Duration
25-30 (70,000)	86,066	39	41,667	-69,803	39	41,728	-69,821	39	41,740	-69,946	39
25-30 (60,000)	86,066	39	36,547	-59,823	42	36,567	-59,678	42	36,585	-59,852	42
125-120 (90,000)	117,870	135	98,742	-89,956	135	101,143	-89,881	135	101,259	-89,908	135
125-120 (75,000)	117,870	135	68,974	-74,932	136	72,399	-74,912	138	75,101	-74,863	136

* Project1 size - Project2 size (Credit Limit)

Table 6.6: Details of results obtained for the multi projects problems (without repair).

Case	Credit Limit	GA				SFLA				SA			
		Best Sol.	Average (StDev)	Dev. (%)	Avg. Time	Best Sol.	Average (StDev)	Dev. (%)	Avg. Time	Best Sol.	Average (StDev)	Dev. (%)	Avg. Time
25-30	60,000	36,547	36,324.6 (532)	0.7	313.5	36546	35542.8 (857.5)	2.8	109.1	36585	36561 (19.2)	0.07	268
125-120	75,000	6,8974	62,444.6 (3649)	16.8	3269	72399	68132.1 (3570)	9.3	2842	75101	69407 (3886)	7.6	2070.1

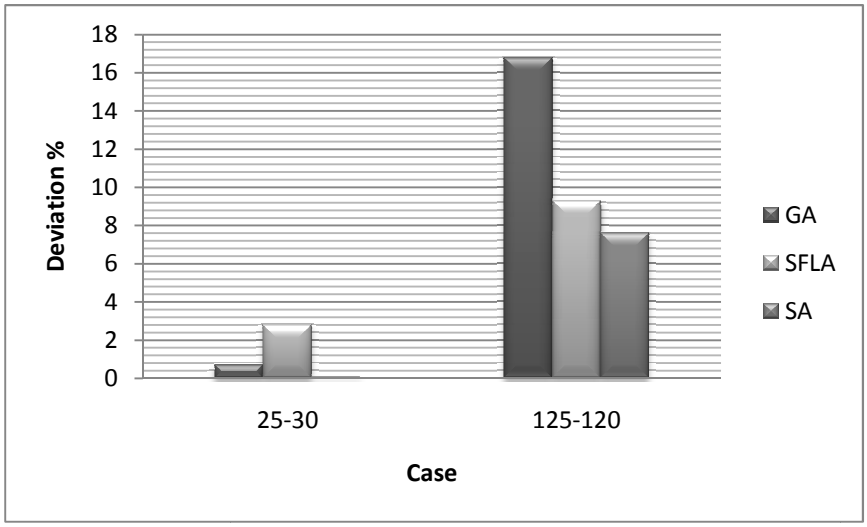


Figure 6.4: Percent deviation from the best solution (Multi projects).

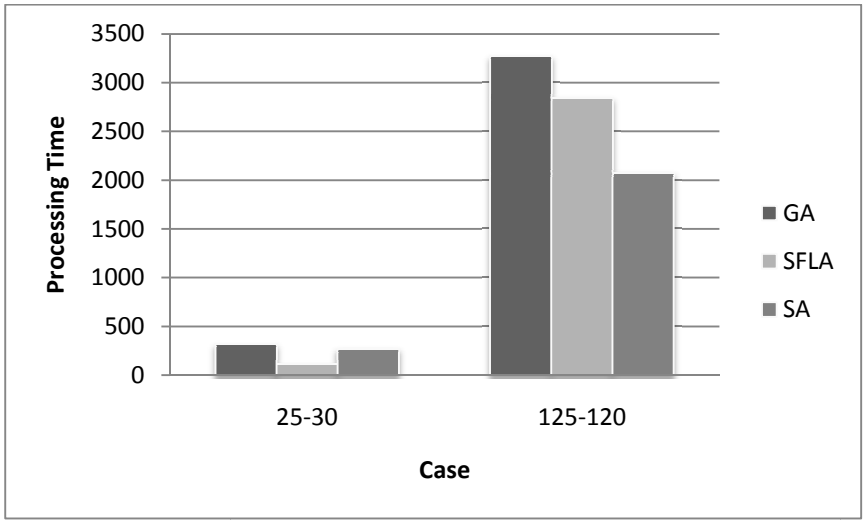


Figure 6.5: Average processing time (Multi projects).

Chapter 7

Conclusion and Future Research

In this thesis, we have modified the cash flow model of Au and Hendrikson (1986) by considering the daily cash flow. This has helped in formulating the problem as an integer program with accurate financing cost calculations. We utilized some modeling tricks in the integer programming model to accurately calculate the financing cost and the last payment received. Due to the NP-hardness nature of this problem, we proposed different heuristic solutions using genetic algorithm, shuffled frog leaping algorithm and simulated annealing. In these heuristics, we have used a representation scheme that guarantees the feasibility of the solution with respect to the precedence relation constraints. In addition, a repair algorithm was developed to repair any infeasible solution with respect the financial constraint. In addition, we redesigned the heuristics to deal with the multi projects finance based scheduling problem. Finally, a study was conducted to compare the performance and the quality of solutions produced by each heuristic.

A huge number of experiments were conducted to select the best parameters for each heuristic. Because of the stochastic nature of such heuristics, a number of runs were conducted and the average along with the standard deviation were used as performance measures of the solution quality. Moreover, the processing time of each heuristic was another performance measure which was recorded.

From the results of the study, we conclude that the repair algorithm is required for both the GA and the SFLA to get good quality solutions. In addition, SA outperformed the other heuristics in the solution quality while SFLA gave fairly better results than the GA and was also faster in terms of processing time. This shows that SFLA is a promising fast heuristic when compared to the GA. Furthermore, these results favored the use of one of the oldest heuristics, SA, for this type of problem.

Further research can be carried out to further investigate several issues. In what follows, we outline the most important issues:

1- **Integer Programming**: we managed to solve some instances of a 60-activity single project problem and it took over a day to get the solution. Further IP modeling should be considered to find a tighter set of constraints that will improve the solution time.

2- **Meta-heuristics**: as SA performed well, it's expected that other single-solution based heuristics like Tabu search will have good performance too. Other meta-heuristics such as ant colony and particle swarm can be applied. A comparative study among these heuristics may further improve the quality of the solution obtained for such problem.

3- **Uncertainty**: the uncertainty can be added to different parts in the finance-based scheduling problem like the activity duration.

4- **Multi-objective**: different objective functions can be considered.

References

Artigues, C., Demassey, S. and Neron, E. (2008). *Resource-constrained project scheduling: models, algorithms, extensions and applications*. ISTE Publishing Company.

Au, T., and Hendrikson, C. (1986). "Profit Measures for Construction projects". *Journal of Construction Engineering and Management*, 115(2), 302-316.

Barbosa P. S. F., Pimentel P. R. (2001). " A linear programming model for cash flow management in the Brazilian construction industry". *Construction Management and Economics*, 19(5), 469-479.

Chiu, H. and Tsai, D. (2002). An efficient search procedure for the resource-constrained multi-project scheduling problem with discounted cash flows. *Construction Management and Economics*, 20(1), 55-66.

Chung, G. and Lansey, K. (2009). Application of the Shuffled Frog Leaping Algorithm for the Optimization of a General Large-Scale Water Supply System. *Water Resources Management*. 23(4), 797-823.

Demeulemeester, E. and Herroelen, W. (2002). *Project scheduling: a research handbook*. Kluwer Academic Publisher.

Doersch, R. H. and Patterson, J. H. (1977). "Scheduling a project to maximize its net present value: A zero-one programming approach". *Management Science*, 23(8), 882-889.

Eiben, A. and James, E. (2003). *Introduction to evolutionary computing*. Berlin, Germany: Springer-Verlag.

Elazouni, A. (2009). "Heuristic method for multi-project finance-based scheduling". *Construction Management and Economics*, 27(2), 199-211.

Elazouni, A. and Metwally, F. (2007). "Expanding Finance-Based Scheduling to devise overall-optimized project schedules". *Journal of Construction Engineering and Management*, 133 (1), 86-90

Elazouni, A. and Metwally, F. (2005). "Finance-Based Scheduling: Tool to Maximize Project Profit Using Improved Genetic Algorithms". *Journal of Construction Engineering and Management*. 131(4), 400–412.

Elazouni, A. and Gab-Allah, A. (2004). "Finance-based scheduling of construction projects using integer programming". *Journal of Construction Engineering and Management*, 130(1), 15–24.

Elbeltagi, E., Hegazy, T. and Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*. 19(1), 43-53.

Eusuff, M., Lansey, K. & Pasha, F. (2006). Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering Optimization*, 38(2), 129-154.

Hegazy, T. and Ersahin, T. (2001). "Simplified Spreadsheet Solutions. II: Overall Schedule Optimization". *Journal of Construction Engineering and Management*, 127(6), 469-475.

Kaka, A. and Lewis, J. (2003). "Development of a company-level dynamic cash flow forecasting model (DYCAFF)". *Construction Management and Economics*, 21(7), 693-705.

Karshenas, S. and Haber, D. (1990). "Economic optimization of construction project scheduling". *Construction Management and Economics*, 8(2), 135-146.

Kazaz, B. and Sepil, C. (1996). Project Scheduling with Discounted Cash Flows and Progress payments. *Journal of Operational Research Society*, 47(10), 1262-1272.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671–680.

Li, S. (1996). "New Approach for Optimization of Overall Construction Schedule". *Journal of Construction Engineering and Management*, 122(1), 7-13.

Liu, S.S. and Wang, C.J. (2008). " Resource-constrained construction project scheduling model for profit maximization considering cash flow". *Automation in Construction*, 17(8), 966-974.

Michalewicz, Z. and Fogel, D. (2002). *How to solve it: modern heuristics*. New York : Springer.

Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution programs*. Berlin, Germany: Springer-Verlag.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT Press.

Navon, R. (1995). "Resource-based model for automatic cash-flow forecasting". *Construction Management and Economics*, 13(6), 501-510.

Padman, R., Smith-Daniels, D.E. and Smith-Daniels V.L. (1997). "Heuristic Scheduling of Resource-Constrained Projects with Cash Flows". *Naval Research Logistics*, 44(4), 365-381.

Russell, R. A. (1986). A Comparison of Heuristics for Scheduling Projects with Cash Flows and Resource Restrictions. *Management Science,* 32(10), 1291-1300.

Russell, R. A. (1970). Cash flows in networks. *Management Science*, 16(5), 357-373.

Sears, G. A. (1981). "CPM/Cost: An Integrated Approach". *Journal of the Construction Division*, 107(2), 227-238.

Smith-Daniels, D.E., Padman, R. and Smith-Daniels, V.L. (1996). "Heuristic scheduling of capital constrained projects". *Journal of Operations Management*, 14(3), 241-254.

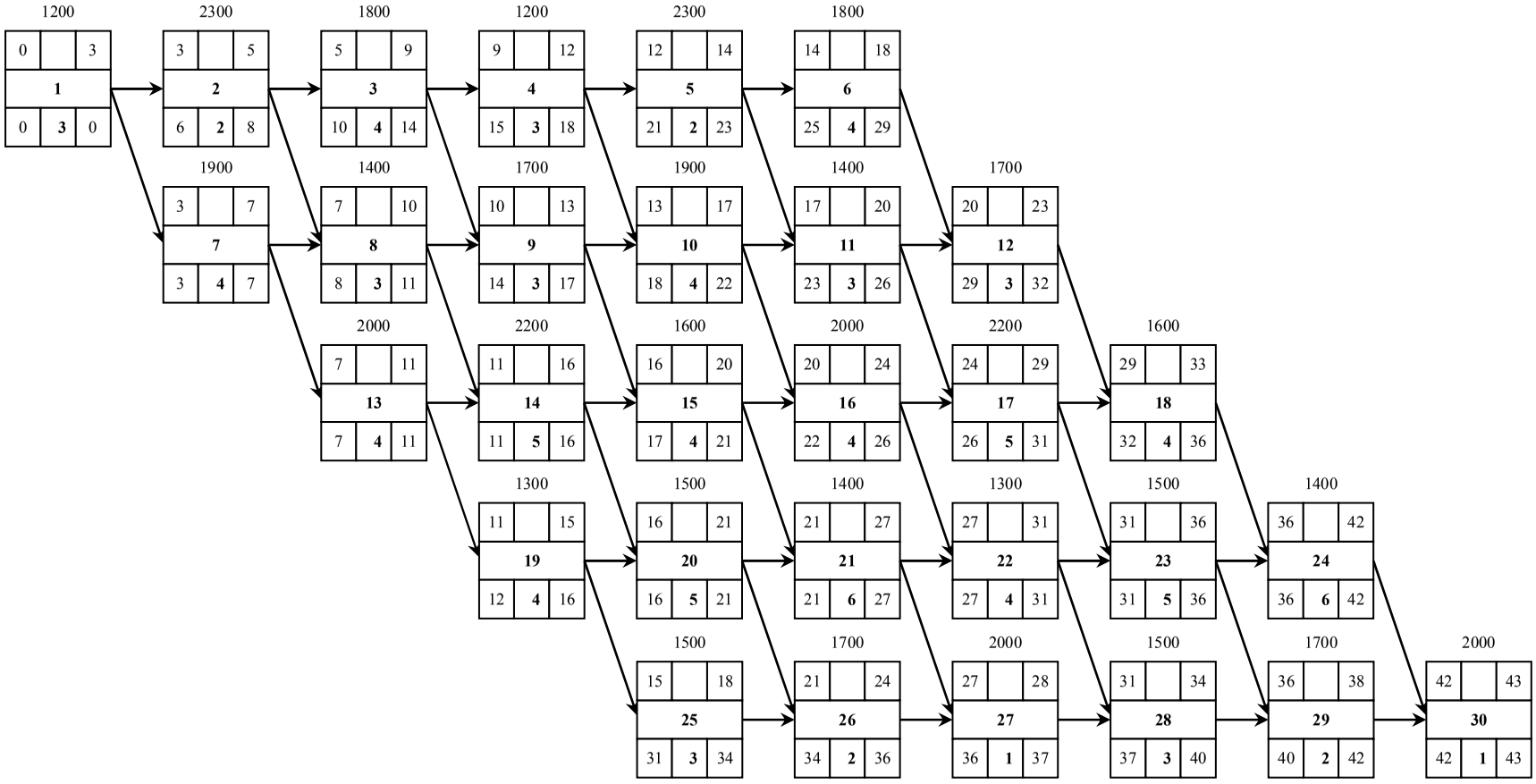
Smith-Daniels, D.E. and Smith-Daniels V.L. (1987). "Maximizing the net present value of a project subject to materials and capital constraints". *Journal of Operations Management*, 7(1-2), 33-45.

Uher, T. (2003). Programming and scheduling techniques. Sydney, NSW: UNSW Press.

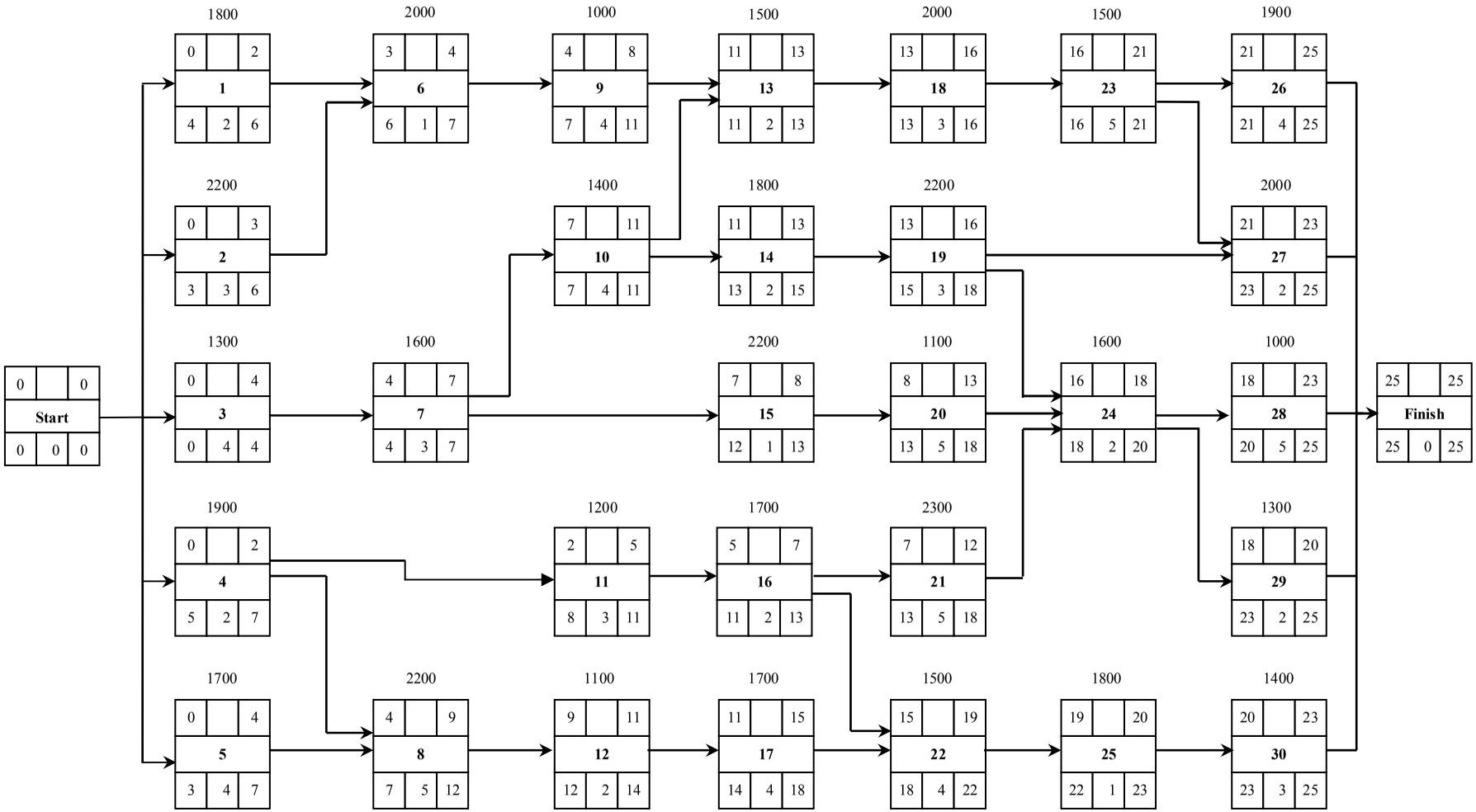
Warazawski, A. (2003). "The parametric analysis of the financing cost in construction projects". *Construction Management and Economics*, 21(4).

Williams, H. P. (1993). *Model building in mathematical programming*. New York: Wiley.

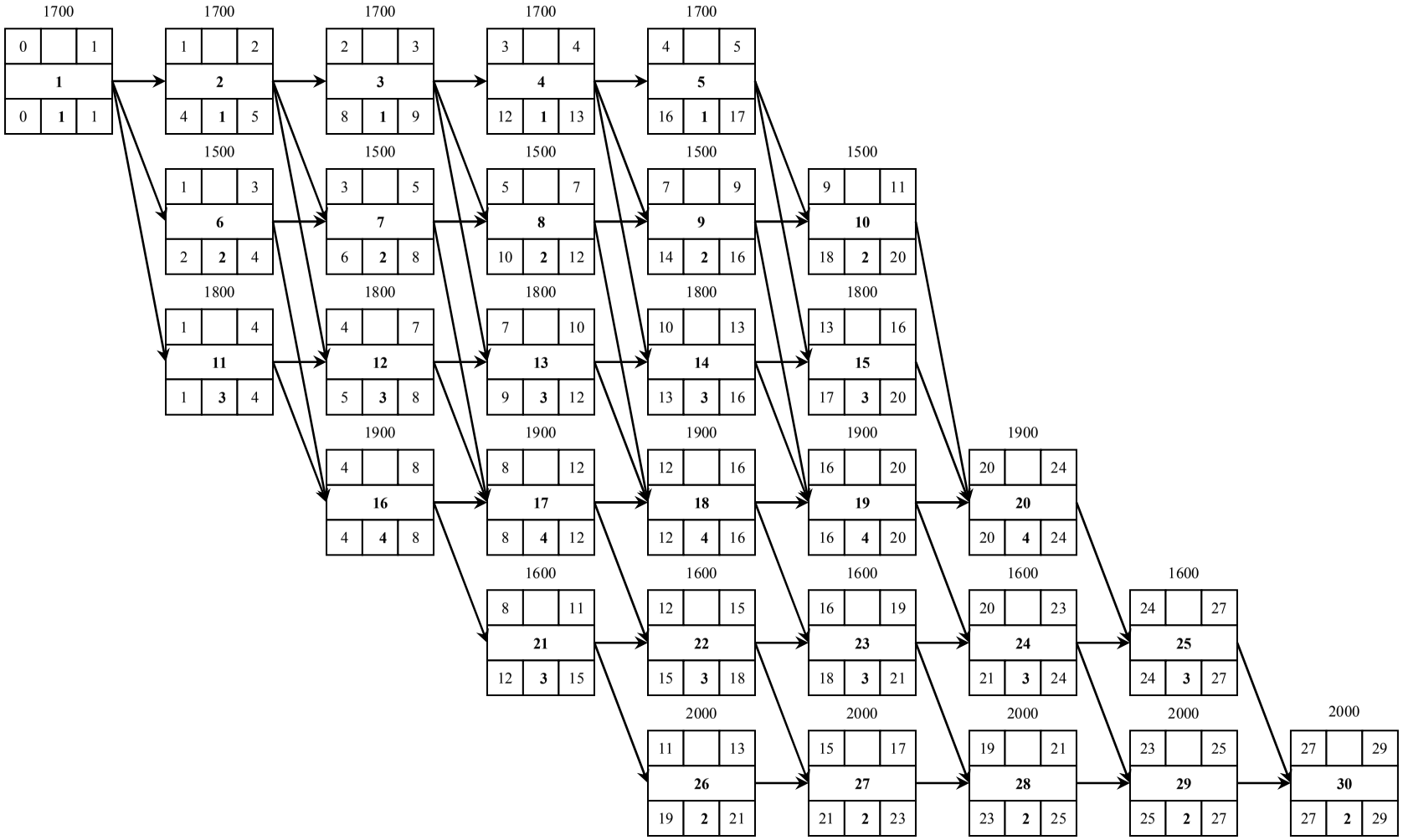
Appendix I



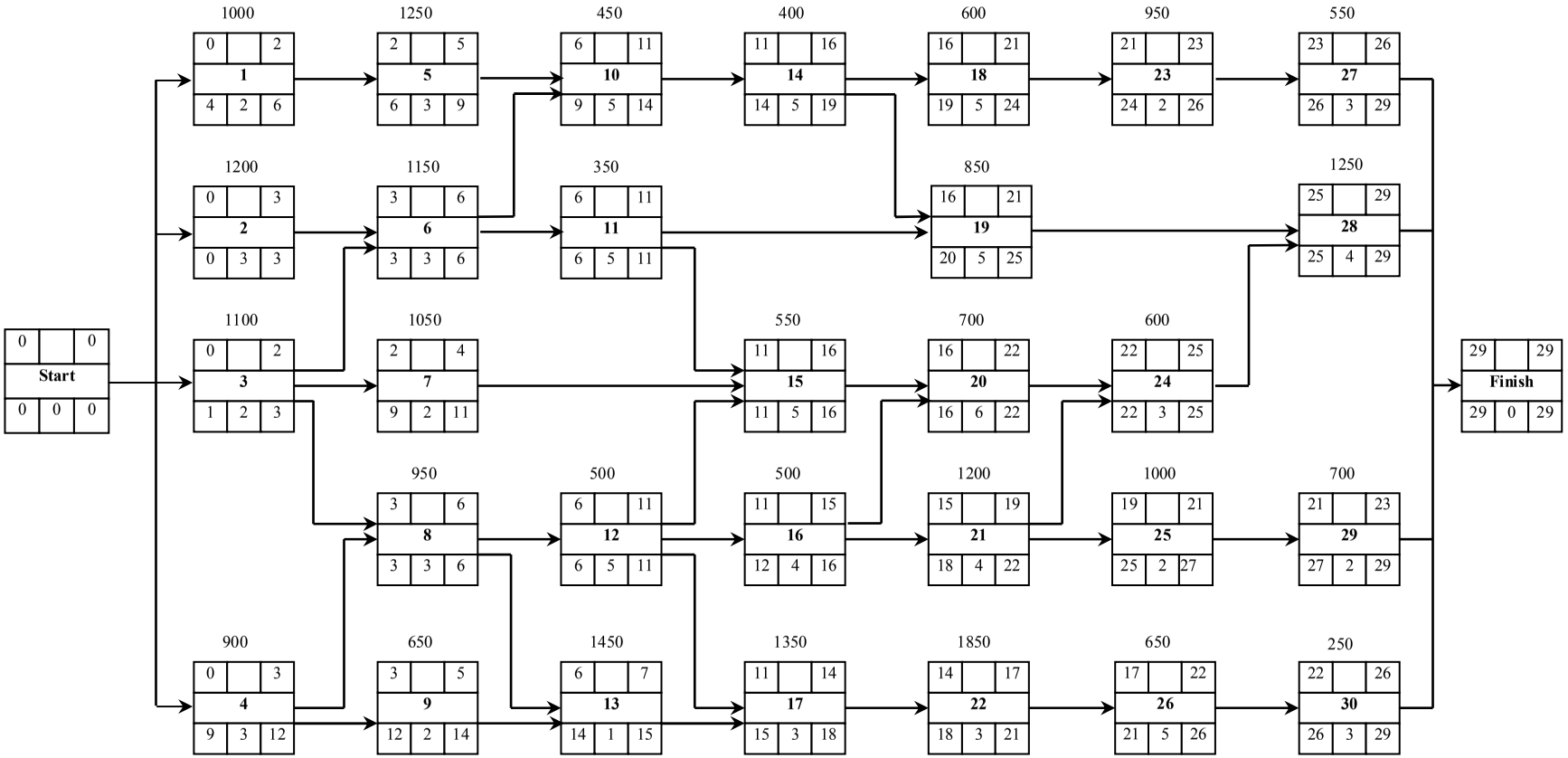
Network 1


Start
0 0 0
Finish
25 0 25
1 1800 0 2 4 2 6
2 2200 0 3 3 3 6
3 1300 0 4 0 4 4
4 1900 0 2 5 2 7
5 1700 0 4 3 4 7
6 2000 3 4 6 1 7
7 1600 4 7 4 3 7
8 2200 4 9 7 5 12
9 1000 4 8 7 4 11
10 1400 7 11 7 4 11
11 1200 2 5 8 3 11
12 1100 9 11 12 2 14
13 1500 11 13 11 2 13
14 1800 11 13 13 2 15
15 2200 7 8 12 1 13
16 1700 5 7 11 2 13
17 1700 11 15 14 4 18
18 2000 13 16 13 3 16
19 2200 13 16 15 3 18
20 1100 8 13 13 5 18
21 2300 7 12 13 5 18
22 1500 15 19 18 4 22
23 1500 16 21 16 5 21
24 1600 16 18 18 2 20
25 1800 19 20 22 1 23
26 1900 21 25 21 4 25
27 2000 21 23 23 2 25
28 1000 18 23 20 5 25
29 1300 18 20 23 2 25
30 1400 20 23 23 3 25

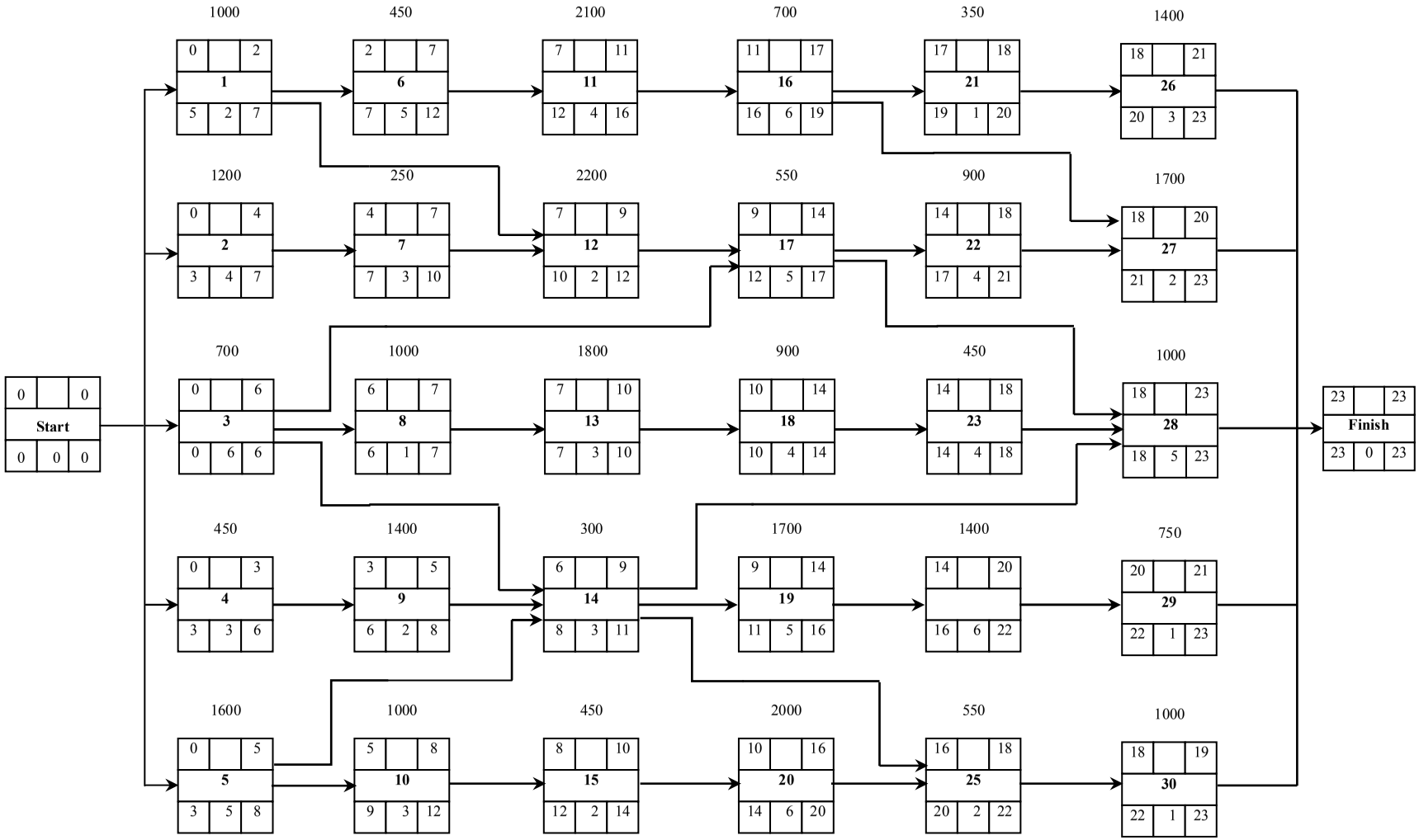
Network 2


1700
0 1
1
0 1 1
1700
1 2
2
4 1 5
1700
2 3
3
8 1 9
1700
3 4
4
12 1 13
1700
4 5
5
16 1 17
1500
1 3
6
2 2 4
1500
3 5
7
6 2 8
1500
5 7
8
10 2 12
1500
7 9
9
14 2 16
1500
9 11
10
18 2 20
1800
1 4
11
1 3 4
1800
4 7
12
5 3 8
1800
7 10
13
9 3 12
1800
10 13
14
13 3 16
1800
13 16
15
17 3 20
1900
4 8
16
4 4 8
1900
8 12
17
8 4 12
1900
12 16
18
12 4 16
1900
16 20
19
16 4 20
1900
20 24
20
20 4 24
1600
8 11
21
12 3 15
1600
12 15
22
15 3 18
1600
16 19
23
18 3 21
1600
20 23
24
21 3 24
1600
24 27
25
24 3 27
2000
11 13
26
19 2 21
2000
15 17
27
21 2 23
2000
19 21
28
23 2 25
2000
23 25
29
25 2 27
2000
27 29
30
27 2 29

Network 3


Start
0 0 0
0 0 0
Finish
29 29 29
29 0 29
1
1000
0 2
4 2 6
5
1250
2 5
6 3 9
10
450
6 11
9 5 14
14
400
11 16
14 5 19
18
600
16 21
19 5 24
23
950
21 23
24 2 26
27
550
23 26
26 3 29
2
1200
0 3
0 3 3
6
1150
3 6
3 3 6
11
350
6 11
6 5 11
19
850
16 21
20 5 25
28
1250
25 29
25 4 29
3
1100
0 2
1 2 3
7
1050
2 4
9 2 11
15
550
11 16
11 5 16
20
700
16 22
16 6 22
24
600
22 25
22 3 25
8
950
3 6
3 3 6
12
500
6 11
6 5 11
16
500
11 15
12 4 16
21
1200
15 19
18 4 22
25
1000
19 21
25 2 27
29
700
21 23
27 2 29
4
900
0 3
9 3 12
9
650
3 5
12 2 14
13
1450
6 7
14 1 15
17
1350
11 14
15 3 18
22
1850
14 17
18 3 21
26
650
17 22
21 5 26
30
250
22 26
26 3 29

Network 4


Start
Finish
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
25
26
27
28
29
30

Network 5

Vita

Anas AlSayed AlGhazi
P.O. Box 1415
Dhahran 31261
Saudi Arabia
Email: aalghazi@gmail.com
Nationality: Saudi

Education

2005-2009 M.Sc. in Systems Engineering (Operations Research), King Fahd University Of Petroleum & Minerals, Dhahran, Saudi Arabia.

1997-2003 B.Sc. (Honors) in Systems Engineering, King Fahd University Of Petroleum & Minerals, Dhahran, Saudi Arabia.

Professional Experience

2005-Present Faculty of Systems Engineering Department, King Fahd University Of Petroleum & Minerals, Dhahran, Saudi Arabia.

2003-2005 Assistant Consultant, MFB Consultants. Jeddah, Saudi Arabia.