# Integer-Programming-Assisted Path-Float-Based Method for Time-Cost Tradeoff Optimization in Project Planning

by

**Sasan Nasiri**

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Construction Engineering and Management

Department of Civil and Environmental Engineering

University of Alberta

# Abstract

In theory, time-cost tradeoff (TCT) optimization is a classic planning problem appealing to construction management; yet, existing analytical methods are found inadequate to make a significant impact in practice. Heuristic methods lack a theoretical basis to ensure arriving at optimum solutions in solving specific problems; on the other hand, mathematical programming requires cumbersome, complicated formulation. This study proposes a new algorithm for TCT optimization that takes advantage of a path-float based scheduling technique and integer programming (IP). The project duration can be shortened in each iteration based on path lengths; while IP is nested to inform on which activities on the critical path(s) to shorten by how long duration. The new TCT optimization approach streamlines critical path analysis in each cycle by elimination of backward pass and finds optimal or near optimal solutions in terms of lowest project cost or shortest project duration. Since only a part of the network (critical paths) is modelled in IP formulation in each intermediate cycle, the complexity of IP formulation plus the search space is substantially reduced. Case studies are used to verify the proposed method and demonstrate its application. The proposed method can be automated to tackle large project networks commonly encountered in (1) project planning and scheduling and (2) acceleration planning and workface planning in construction management.

# Preface

This thesis is an original work by Sasan Nasiri.

A version of the proposed algorithm streamlined for linear activity time-cost relationship has been published as a conference paper: Sasan Nasiri and Ming Lu (2019). "Path-Float Based Approach to Optimizing Time-Cost Tradeoff in Project Planning and Scheduling". *Computing in Civil Engineering 2019: Visualization, Information Modeling, and Simulation*. American Society of Civil Engineers. Dr. Ming Lu was the correspondence author of and was involved with concept formation and manuscript composition. In addition, we were invited and had accepted to submit an extended version of the aforementioned conference paper to Journal of Management in Engineering (ASCE).

# Acknowledgements

I would like to thank Professor Ming Lu, my supervisor, for his kind, generous, and unwavering support during my M.Sc. study at University of Alberta. I am deeply grateful to have had the opportunity to work under his guidance.

I would like to thank my industrial supervisor at Ledcor, Rod Wales, for his support, valuable advice, and comments on the research. In addition, I express my gratitude to Adam Person at Ledcor, for his detailed and helpful guidance.

# Table of Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| AOA | Activity-on-arrow |
| AON | Activity-on-node |
| CPM | Critical path method |
| FS | Finish-to-start |
| IP | Integer programming |
| PDM | Precedence diagram method |
| PERT | Program evaluation and review technique |
| PFCPM | Path-float based critical path method |
| RACPM | Resource-activity critical path method |
| TCT | Time-cost tradeoff |
| TLBO | Teaching-learning based optimization |

# Chapter 1. Introduction

## 1.1 General Introduction

Project planning and scheduling is an important task that is required at almost all levels of an organization (Moussourakis and Haksever, 2009). Various scheduling techniques such as Gantt Bar Chart, Program Evaluation and Review Technique (PERT), and Critical Path Method (CPM) have been generally accepted in both knowledge and practice of construction management (Olawale and Sun, 2010). CPM has been widely used for scheduling during the past six decades and has become a standard practice. A project is broken down into distinct activities, which must be completed in a logical and technologically constrained sequence (precedence relationships) in order to finish the project. Activity-on-node (AON) diagram is the graphical presentation of these activities. Precedence relationships along with activity durations are used as structured problem definition for CPM to determine the shortest project duration. However, CPM is a duration-oriented technique (Hegazy, 2002) and suffers from several limitations. To a certain extent, CPM ignores project costs and is unbale to confine the schedule to a specified duration (Hegazy, 2002).

In construction industry, the aim is to ensure that a project finishes both on time, and within a budget. Therefore, time and cost, among other objectives, are the most important decision-making parameters (Olawale and Sun, 2010). In order to survive the price competition in the current market, contractors must try to minimize project delivery costs as much as possible (Zheng and Ng, 2005). TCT is an

important tool for overcoming CPM's limitations in terms of ignored cost and schedule compression (Hegazy, 2002).

The cost of a project has two components: direct cost and indirect cost. Direct costs are generally attributed to specific activities in the project while indirect costs are accounted by the project as a whole. In general, indirect cost increases linearly with the increase of project duration (Ahuja, 1984). In addition, an owner may award a bonus for early completion or ask for liquidated damages for late completion (Ahuja et al., 1994). Several options (alternatives, or modes) for execution of an activity can be proposed, each having a different direct cost and duration. Generally, activity direct cost increases as duration of an activity decreases (Hegazy, 2002). As a result of these options, there can be numerous possible combinations to complete a project. A tradeoff between project time and project cost becomes apparent when comparing these combinations. If cutting project cost is of major concern, each activity can be performed at its lowest possible cost. If shortening project time is of primary concern, each activity can be performed at its shortest possible duration. Between these limits, there are two global optimums on the project's time-cost curve. One is the minimum total project duration and the least project cost corresponding to that minimum total project duration. The other is the minimum total project cost and the least project duration from all durations corresponding to that minimum total project cost.

In TCT, the main goal is to find these two global optimums. TCT is a combinatorial bi-objective optimization problem (Xiong and Kuang, 2008) and several techniques have been provided for it, each having their own advantages and limitations which will be comprehensively reviewed in Section 2.3. To summarize, when it comes to large-sized real project networks, existing methods suffer from one or more of the following limitations:

- Complex, error-prone, and time-consuming formulation and modelling,
- Intensive computation.

As a result, practicality of these methods is undermined and TCT implementation is extremely rare in the real world.

In this study, an iterative methodology inspired by the classic cost slope method for TCT analysis has been developed. In particular, a path-based scheduling approach proposed by Lu et al. (2017) has been employed instead of traditional CPM for identifying critical activities in TCT, which considerably streamlines the scheduling analysis by eliminating the backward pass of classic CPM. Furthermore, integer programming is nested in each iteration to formulate activity selection and enable us to implement this method for large-sized networks. The proposed TCT framework results in optimal or near-optimal solutions, provides feasible alternative solutions, and can be readily implemented in practice.

## 1.2 Research Motivation and Objectives

There is a tradeoff regarding existing solutions for TCT, which is optimality versus formulation, computational effort, and ease of use, or in other words, practicality. As the methods become more oriented toward mathematical optimization, the amount of required effort in formulation substantially increases. Literature published in the field of operations research and industrial engineering often tend to focus on finding the absolute optimum solution to TCT. However, in construction, although optimality is valued, efficiency and practicality of the method have their own significance. Number of feasible crashing permutations increases exponentially in large projects (Bettemir and Birgönül, 2017), which is usually the case in real world construction projects. In construction, we must be able to perform TCT for

various types of projects with widely different characteristics and in some instances, under a limited time. Exact methods are complex for construction planners to perform (Bettemir and Birgönül, 2017) and typical project management teams may lack the expertise or time to implement exact optimal TCT algorithms (Moussourakis and Haksever, 2004). As a result, TCT is not widely accepted in practice and in many cases, the opportunity to save more cost is missed and experience-based rules of thumb are used for construction projects.

On the other hand, methods that do not guarantee optimal solutions, e.g. heuristic methods (Liu et al., 1995), lack the mathematical rigor and can result in missing the optimal solution by a large margin.

Therefore, having identified this tradeoff, the purpose of this thesis is to provide a practical algorithm that minimizes the amount of computational and modelling effort. At the same time, this algorithm must be "optimal enough" to ensure near optimal solutions.

## 1.3 Thesis Organization

The present thesis consists of five chapters. Chapter 1 provides an introduction and overview of the problem and identifies objectives of this research. In Chapter 2, a comprehensive literature review of TCT problem, categories of TCT, TCT history, and existing methods for TCT are presented. In Chapter 3, path finding methods required for the proposed methodology and the generic methodology proposed for curvilinear time-cost relationship is presented. Subsequently, the methodology is streamlined for linear and multilinear activity time-cost relationship. This is followed by analytical discussion of the algorithm and notes on the advantages. In Chapter 4, two case studies from various sources are solved by the proposed

algorithm and the results are compared with their existing solutions for cross-validation. The first case is a small project and detailed solution is provided to clearly demonstrate the use of the proposed methodology. For the second case, the results will be compared with an exact method. Finally, in the third case, proposed algorithm is performed for a project network of practical complexity. Chapter 5 concludes the thesis and summarizes advantages and limitations of the methodology along with future work.

# Chapter 2. Literature Review

This chapter is divided into three parts. In Section 2.1, terminology and description of TCT are provided to complete the problem statement in Chapter 1. Subsequently, various categories of TCT in literature are briefly identified in Section 2.2. Finally, a comprehensive review of the existing methods for TCT is provided in Section 2.3, with a focus on construction publications.

## 2.1 Time-Cost Tradeoff

Total cost of a project has two components. One is the direct costs that incur as a result from execution of specific activities (Hegazy, 2002). Therefore, direct costs can be attributed to particular activities. The other component is the indirect cost which cannot be attributed to specific activities. Indirect cost consists of project overhead and general overhead (Hegazy, 2002). Project overhead is the indirect costs that are specific to the project, such as wages and salaries of supervisors, safety, on site office expenses, etc. General overhead is the head office expenses that have been attributed to a particular project (Peurifoy and Oberlender, 2002). A method for division of general overhead is distributing it relative to total direct cost of projects.

The least direct cost required for completing an activity is called the normal cost, and the corresponding duration is called the normal duration; while, the shortest possible duration for completing an activity is called the crash duration, and the corresponding cost is called the crash cost (Hegazy, 2002). Some examples for methods of crashing include implementing overtime hours, faster installed material, multiple shifts, more resources, or different technology (Hegazy, 2002). In general, the less expensive to complete an activity, the longer is the duration

(Hegazy, 2002). Additional cost incurred by crashing could be attributable to mobilization of additional equipment, overtime work, or reduced productivity resulting from nightshift work or congested work space (Ahuja et al., 1994). The opposite of crashing is relaxation. Sometimes it may be necessary to relax activities, particularly for non-critical activities, to make a project more economical (Ahuja et al., 1994). In this thesis, the assumption is that each activity is initialized at the most relaxed option (normal).

Activity time-cost relationship can be categorized as continuous vs. discontinuous (Moussourakis and Haksever, 2004). Continuous time-cost relationship can be either linear, multilinear, or curvilinear (Ahuja et al., 1994). Discontinuous time-cost relationship can be discrete points, or linear with gaps in between (Moussourakis and Haksever, 2004). Graphical representation of these relationships is shown in Figures 2.1, 2.2, 2.3, 2.4, and 2.5. For continuous curvilinear time-cost relationship, the cost curve can be divided into many small, straight line sections in order to improve practicality (Ahuja et al., 1994). Activity time-cost relationship can be both convex and concave (Falk and Horowitz, 1972).

An example of continuous linear time-cost relationship is overtime work. As an example of continuous multilinear time-cost relationship, consider an earthwork job, where for the first few days, crashing may be done by using a loader of different capacity. Beyond that, the job may require two loaders instead of one. The mobilization cost of the second loader is not covered by the same slope. An example for discrete points is that for a tunneling project, either a drill jumbo or a mole (tunnel boring machine) may be used. By using a mole, duration is reduced and cost jumps up. For continuous curvilinear, consider a dike construction project using graders and dozers of certain capacities (one grader, or two graders, or two graders and one dozer, or three graders, or four graders) (Ahuja et al., 1994).

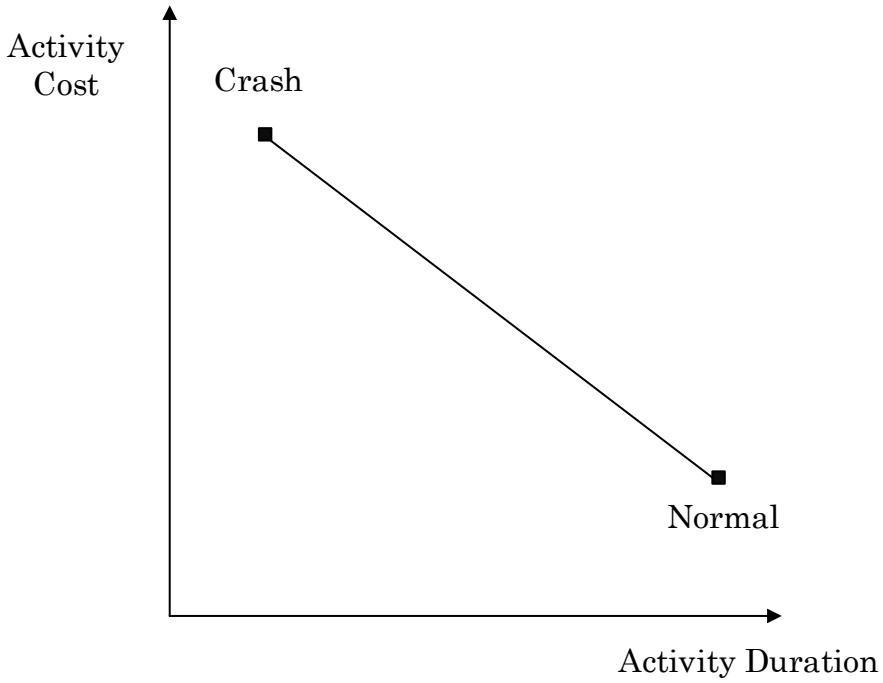



Figure 2.1 Linear continuous activity time-cost relationship

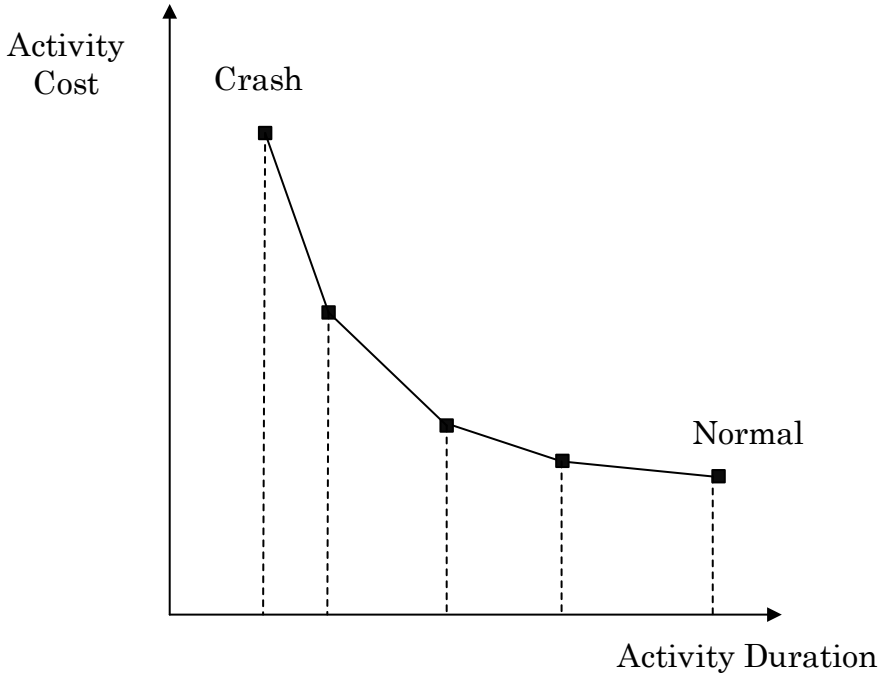



Figure 2.2 Multilinear continuous activity time-cost relationship

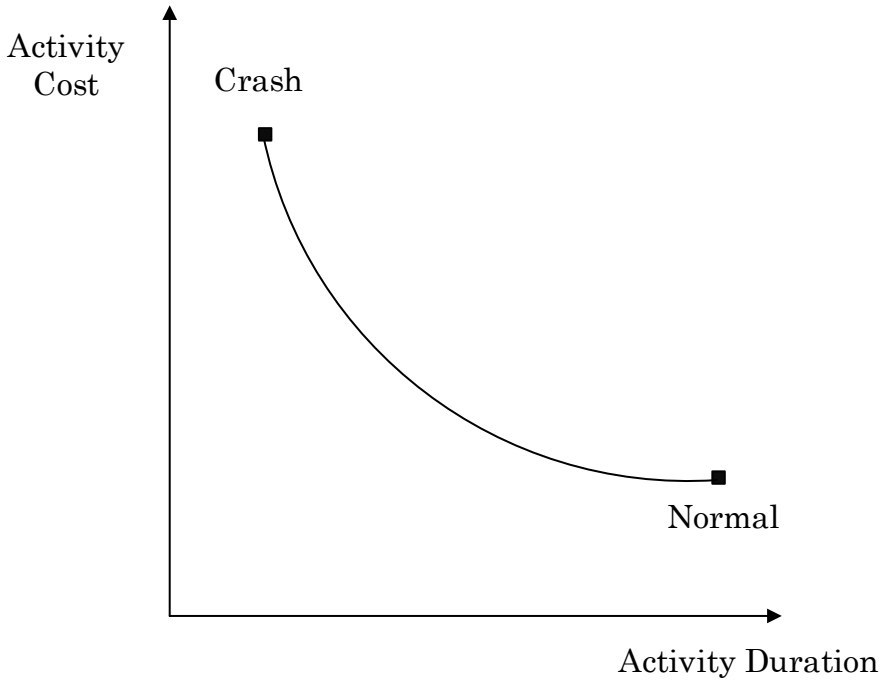



Figure 2.3 Curvilinear continuous activity time-cost relationship

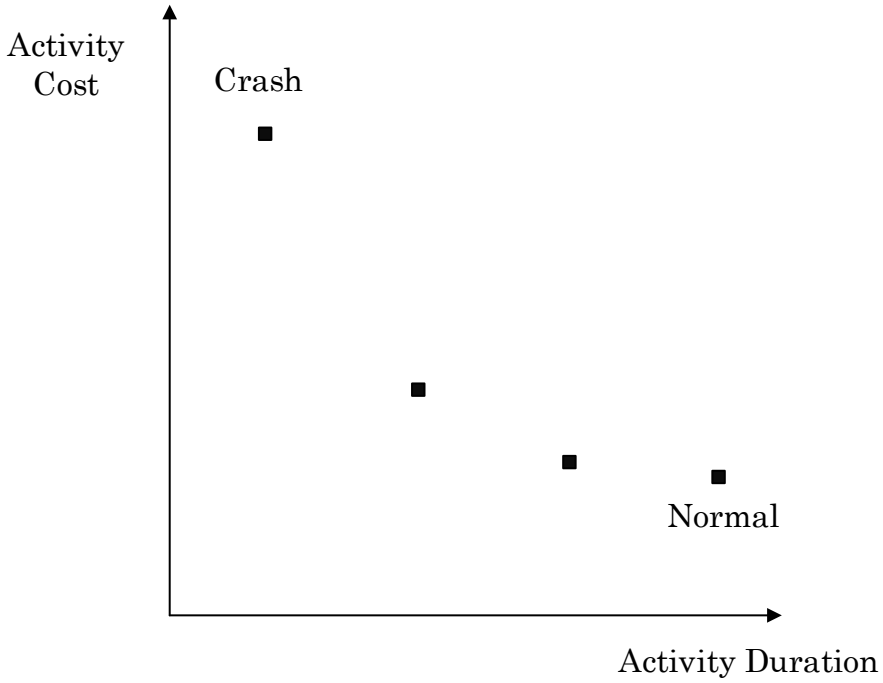



Figure 2.4 Discrete point activity time-cost relationship

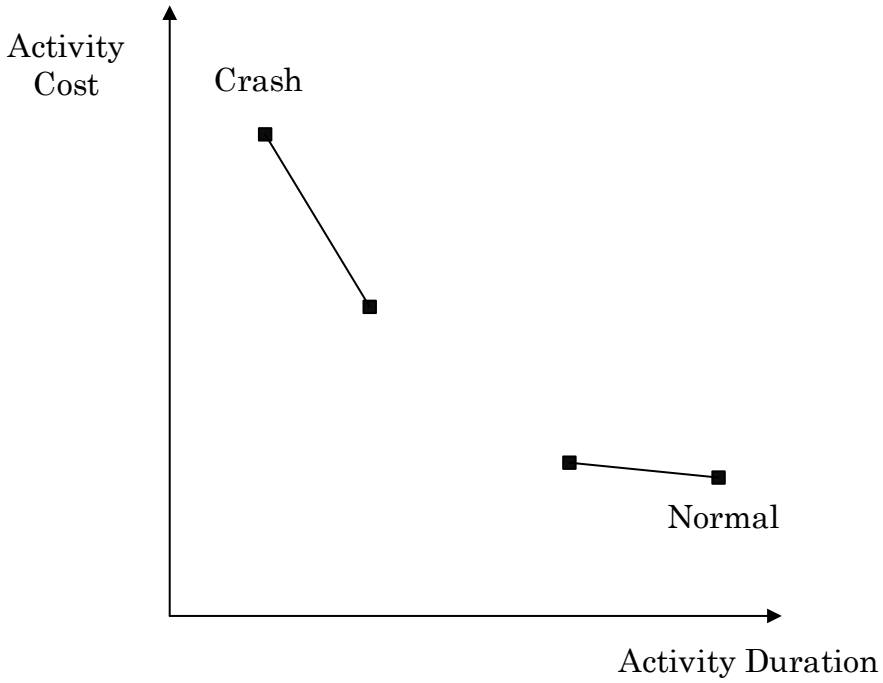



Figure 2.5 Discrete linear with gaps activity time-cost relationship

Construction projects are seldom entirely identical (Zheng and Ng, 2005) and for each project, execution modes for activities are required to perform TCT. In the absence of a mature tool for recording and retrieving time-cost data of activities, it would be costly to establish a series of possible time-cost relationships for each activity (Zheng and Ng, 2005).

The exercise of determining the most economical project duration proves to be the most economical on projects of repetitive type such as single-span bridges, multistory buildings, prefabricated buildings, or installation of equipment in process plants. Hence, once the most economical duration is determined, it can result in cumulative savings from similar projects to be done in the future (Ahuja et al., 1994).

Discrete TCT is classified as NP-hard (non-deterministic polynomial time) with a large number of variables and constraints (De et al., 1997). There are many possible

permutations to perform construction activities. The number of combinations increases exponentially with the number of activities. Therefore, finding the optimal TCT solution is difficult and time-consuming (Hegazy, 2002). Evaluating each option requires recalculation of schedule using CPM (Hegazy, 2002). TCT is traditionally performed separate from resource allocation and levelling (Hegazy, 2002).

## 2.2 Categories of Time-Cost Tradeoff Problems

Traditional TCT assumes all values are deterministic. However, in reality, time and cost are uncertain as a result of factors such as weather, resource availability, or productivity (Zheng and Ng, 2005). Hence, a category of TCT is stochastic TCT that assesses the risk in terms of time and cost (Feng et al., 2000). Feng et al. (2000) provided a stochastic approach for TCT. However, this approach relies on sampling statistical input models and a large number of Monte Carlo simulations. The lack of historical time and cost data to define inputs presents a challenge for its implementation. Moreover, the method heavily relies on computing experiments and analyzing observed system responses. The stochastic approach adds an additional dimension of complexity to the already hard to apply TCT. Apart from the integration of pareto front optimization with Monte Carlo simulation, the method is limited in terms of theoretical advance to TCT. El-Kholy (2013) introduced a linear programming model of stochastic TCT where the variability of funding and uncertainty of project duration were considered simultaneously.

Another category of TCT considers resource planning in addition to time and cost. Robinson (1975) included resources into TCT and turned it into a dynamic programming problem. It proceeded to find the time-cost function of the project for different levels of resources.

Traditional TCT assumes constant value of money along the project span, regardless of direct or indirect costs. In reality, value of money decreases over time. As a result, another category of TCT problems is that to consider the time value of money. Ammar (2010) provided a method for TCT by accounting for discounted cash flows based on the net present value. Costs of activities were assumed to incur at their finish time. It subsequently used mathematical programming for optimization.

Another aspect in TCT is the potential quality loss that results from crashing activities. If some activities are excessively crashed, it can result in rework, modifications, or even project failure (Kim et al.,2012). Kim et al. (2012) addressed this issue by proposing a mixed linear integer programming model that incorporated potential quality loss cost. This quality loss cost was the estimated direct cost of rework and modifications related to nonconformance of activities.

## 2.3 Existing Methods

There is a large body of knowledge regarding TCT analysis. We start by reviewing the earliest examples of TCT. Kelley and Walker (1959) formulated TCT as a linear program in their seminal paper. The cost of each job was defined as a linearly decreasing function of its duration. The objective function was to minimize project direct cost which was defined as the sum of all direct costs. Constraints included activity duration upper and lower limits (crash and normal durations) and precedence relationships. Kelley (1961) defined the concept of utility of an activity and tried to minimize or maximize it for a whole project. Utility could be cost, quality, or resources and was modelled as a linear function. It used primal-dual algorithm (Dantzig et al., 1956) and a network flow algorithm to solve TCT with linear programming.

Fulkerson (1961) provided a network flow method for TCT which assumed a linear activity time-cost relationship. The algorithm used was a labeling process which was a systematic search for a "path" with certain properties from start to finish. Note that the definition of "path" described here is different from the usual definition, as an arc can be traversed against its direction. Prager (1963) provided a structured interpretation of the method presented by Fulkerson (1961) for civil engineers to enable them (assuming they are not familiar with mathematical programming and flow theory) to better understand the problem.

Most sources, such as Hegazy (2002), divide the existing methods into three groups, namely: heuristic methods, mathematical programming methods, and evolutionary algorithms. However, another branch of solutions is maximum flow and minimum cut theory (Jiang and Zhu, 2010). Methods that relate to our proposed algorithm will be discussed in depth. However, as numerous solutions for TCT exist, other methods will not be comprehensively discussed and will just be mentioned (especially highly cited publications).

Since early 1960s, heuristic methods and mathematical programming (optimization) methods had been traditionally used as the two main categories of solutions for TCT (Ammar, 2018; Hegazy, 2002). With advances in the artificial intelligence branch of computer science and growth of computer technology, evolutionary algorithms had emerged and many recent TCT publications focused on the use of these algorithms (Hegazy and Ayed, 1999; Jiang and Zhu, 2010).

Heuristic methods can be described as simple rules of thumb (Hegazy, 2002) that require less computational effort than mathematical programming (Liu et al., 1995). Ahuja et al. (1994) provided an example of an iterative heuristic method. Compression procedure outlined in Ahuja et al. (1994) used a relatively similar

method to ours, but without the IP formulization. At first, activities that would have never become critical were identified using the criticality theorem. Criticality theorem denotes that if the crash duration (all activities completely crashed) of path C1 is greater than the normal duration of path C2, activities that belong only to path C2 do not need to be considered for crashing and can be eliminated from the analysis. This method only crashed the activities on critical path(s). Hegazy (2002) provided a simple heuristic approach called the cost-slope method which assumed linear time-cost relationship. This method calculated the cost slope of each activity and in each iteration, critical paths were shortened by a manual choice of activities to crash (simply comparing the cost slope of a single activity on all critical paths to choose candidate activities). In addition, if float times were introduced, those activities would be relaxed to reduce the cost.

Mathematical programming methods convert the problem into standard mathematical optimization models and use linear, integer, or dynamic programming to obtain the optimal solution (Ammar, 2018). Basically, most formulations minimize total cost as the objective function and apply certain time constraints and other resource constraints (Jiang and Zhu, 2010).

Ahuja et al. (1994) presented a linear programming approach for TCT which provided exact solutions. In this approach, all activities were initialized at the crash level and would be relaxed. Objective function represented the total cost of the project, with activity durations as the variables in the optimization. Constraints included:

- For each activity, duration must have been between crash and normal durations.

- For each activity, a constraint denoting that the start time of an activity plus activity duration should have been less than (allowing float time) or equal to the end time of activity.

Formulation was then updated to handle multilinear or curvilinear time-cost relationships. Curvilinear was approximated by several linear parts. In this case, each linear part was represented by a constraint.

Liu et al. (1995) used a combination of linear programming and integer programming to solve TCT. Continuous curvilinear relationships were converted into piecewise linear (multilinear). All activities could only take integer durations. It used convex hull to eliminate crash options that were feasible but incurred too much cost and used linear programming to identify a general pattern of the project's overall time-cost relationship. From this curve, regions could be selected for the integer programming part to find the exact solutions. As a result, the methodology could produce nearly optimal solutions.

Deckro et al. (1995) used a mathematical programming formulation by employing a quadratic time-cost relationship for each activity. Objective function represented the sum of cost of all activities while constraints included precedence relationships, upper and lower bounds for activity durations, and a target completion date for the project.

Moussourakis and Haksever (2004) used a flexible mixed integer programming model that could handle a mix of continuous and discrete time-cost relationships with the assumption that continuous portions of cost curve were piecewise linear. Activity durations were selected as optimization variables and constraints included precedence relationships.

Jiang and Zhu (2010) proposed an integer linear programming model to calculate the normal project duration (all activities executed at normal duration and cost) with each precedence relationship and finish time of an activity presented as a constraint. For this part, the objective function was the project finish time. Subsequently, it proceeded to use a second formulation assuming all activities to be linear continuous. The objective function of this part was the project cost while each precedence relationship and activity finish time was imposed as a constraint. Finally, binary linear programming was used to adjust the solution obtained previously given nonlinear continuous activities.

Moussourakis and Haksever (2009) provided three mixed integer linear programming models for TCT that could handle continuous time-cost relationships (both convex and concave). A piecewise linearization method that could approximate nonlinear cost functions was presented. Each model focused on a different objective (minimizing project completion date, minimizing project cost, or minimizing project cost under early completion bonus or late completion penalty).

Reducing project duration results in reduction of total float of non-critical activities which limits the schedule flexibility and increases the risk of project delays (Al Haj and El-Sayegh, 2015). Al Haj and El-Sayegh (2015) proposed a nonlinear integer programming model which accounted for the impact of total float loss and considered a new type of tradeoff between time, cost, and flexibility. It introduced total float unit cost for non-critical activities. The objective function was to minimize total cost which included float unit cost as well. This would result in a solution with higher duration and cost but with lower risk.

Hegazy and Ayed (1999) developed an Excel model for discrete TCT by using nonlinear integer programming and designating the method index (each discrete mode of an activity was represented by a method index) as the variables.

As a more recent example of mathematical programming in TCT, Zou et al. (2016) used mixed integer linear programming for TCT analysis of repetitive projects. Decision variables included start time of activities, idle time of activities, and a binary variable representing crews used by activities. The problem could be solved by either an exact model or an approximate one.

Evolutionary methods are optimization search procedures that mimic natural evolution (Hegazy, 2002). Examples of evolutionary algorithms include genetic algorithms, particle swarm, and ant-colony systems (Elbeltagi et al., 2005). Feng et al. (2000) presented a hybrid approach that combined simulation technique and genetic algorithm to solve stochastic TCT.

Zheng and Ng (2005) used fuzzy sets theory to model managers' behavior in predicting time and cost of a certain option within an activity. It then used genetic algorithms to establish the optimal time-cost profile under different risk levels.

Yang (2007) presented a particle swarm optimization algorithm for TCT that could handle nonlinear, discrete, or piecewise discontinuous activity time-cost relationships.

Xiong and Kuang (2008) used ant colony optimization for TCT. It used modified adaptive weight approach (Zheng et al., 2004) that utilized useful information from current set of solutions to generate weight for each objective and apply a search pressure towards the ideal point.

Pathak and Srivastava (2014) presented an integrated artificial neural network and hybrid meta heuristic approach for nonlinear TCT. Hybrid meta heuristic is an evolutionary multi objective optimization technique.

Togan and Eirgash (2019) used a multi-objective optimization model based on teaching-learning based optimization (TLBO) for TCT to find a set of Pareto front solutions. TLBO is a nature-inspired algorithm and uses a population of solutions to proceed to the global solution (Rao et al., 2011).

Falk and Horowitz (1972) used an algorithm that solved a sequence of maximal flow subproblems which yielded a global solution.

Liu and Rahbar (2004) used maximal flow and minimal cut theory. The maximum possible flow from start to finish on a path is equal to the minimum flow capacity of components on that path. If there are no connections between paths from start to finish, the minimum cut set capacity is defined as the sum of minimum flow of each path. This methodology modeled the cost slope of each activity as its flow capacity. It then started an iterative algorithm. In each cycle, each activity's capacity was zero for non-critical activities, $\infty$ for activities without crashing options, and cost slope for activities on critical paths. Subsequently, the minimal cut set on the critical paths would be determined. This is similar to selection of a set of activities on critical paths that result in the least combined cost slope. Some other methods that have used maximal flow and minimal cut set in an iterative manner are Phillips and Dessouky (1977) and Hochbaum (2016).

Project's duration depends on the length of the critical path which needs to be crashed in TCT. In practice, only a small number of paths are dominant while others are redundant (Ammar, 2018).

Siemens (1971) proposed a path-based method that considered a desired project duration. It developed a time-cost matrix, with each row representing an activity and each column representing a path. It proceeded to manually select activities for crashing in an iterative crashing algorithm (in each step, one activity was crashed). This algorithm gave rules for choosing the paths and activities for crashing without providing a programming formulation. It defined an effective cost slope for each activity by dividing the actual cost slope of activities by the number of inadequately shortened paths. This effective cost slope had to be revised whenever any path had been adequately shortened. Activity with the lowest value of effective cost slope would have been selected in each step for crashing and the procedure would repeat until all paths were adequately shortened. This method might cause unnecessary crashing. Goyal (1975) provided a path-based method based on Siemens' (1971) method with addition of de-shortening. Siemens and Gooding (1975) adjusted the methodology of Siemens (1971) for nonlinear convex cost slopes which included de-shortening of activities [similar to Goyal (1975)]. Both algorithms are difficult to implement and require repeated calculations even for small networks (Goyal, 1996). Goyal (1996) combined the algorithms to take advantage of both. However, it resulted in unnecessary crashing and required de-shortening.

Baker (1997) presented a path-based network flow approach for linear TCT. In each iteration, the project duration would be reduced by one time-unit. It used a linear programming formulation, minimizing the cost with a variable representing the duration of an activity. Subsequently, maximal flow and minimal cut set was employed. Duration of some activities might be increased in this method.

Ammar (2018) proposed a path-based approach for discrete TCT. For each activity, several binary variables were defined, each representing one of the discrete crash options. The summation of these variables for each activity were constrained to be

equal to one, ensuring that only one mode had been selected. In addition, duration and cost of each activity were defined using these variables and cost or duration coefficients. Then, a mathematical programming formulation was used to minimize the total cost (which was the addition of costs of all activities). Path length of each existing path was defined as the sum of duration of all activities on that path. Therefore, as constraints, path lengths must have been less than the desired project duration. In addition, overlap was allowed as a fixed value or as a percentage of activity duration. Ammar (2018) also used the criticality theorem presented by Ahuja et al. (1994). An interpretation of this theorem is that if a path has a normal length less than or equal to the crash project duration (shortest possible duration), this path can never become critical and can be excluded from TCT optimization. In this way, a number of redundant paths could be removed from TCT.

Su et al. (2015) introduced the concepts of safety float and interference float to CPM and calculated path lengths based on safety float and free float. It provided a mathematical programming approach for continuous time-cost relationship, minimizing total cost of all activities subject to typical constraints (precedence relationships, upper and lower bounds for activity duration, and limiting the project duration to a desired length). In addition, for discrete TCT, binary variables were selected for each execution mode of each activity. The summation of these variables for each activity were constrained to be equal to one [similar to Ammar (2018)]. However, for schedule constraints, each precedence relationship was required to be formulated [as opposed to Ammar (2018)].

Bettemir and Birgönül (2017) proposed a path-based iterative approach for discrete TCT that converged to optimum or near optimum solutions. Activity selection for crashing was performed manually and scheduling in each iteration required a backward pass.

Heuristic methods are easy to understand and can provide acceptable solutions (Hegazy, 2002). However, these methods lack mathematical rigor, assume linear time-cost relationship (Hegazy, 2002), and do not guarantee optimum solutions (Hegazy, 2002; Ammar, 2018).

Regarding mathematical programming methods, formulating constraints and objective function is time-consuming and prone to errors. For large networks, the effort that is required to check and verify the program's formulation could be substantial. Mathematical programming knowledge is necessary to formulate these models correctly. Few construction planners are trained to perform this type of formulation, especially for large networks (Liu et al., 1995). Exact solution algorithms for TCT are known to be exponential in the worst case and the solution time would increase exponentially as the problem size increases (Moussourakis and Haksever, 2004). Mathematical programming can be ineffective when dealing with a large number of variables or nonlinear objective functions (Jiang and Zhu, 2010). To summarize, programming models require complex formulation, are computationally intensive, can only be applied to small networks, can get stuck in local minimums, and mostly assume linear time-cost relationship (Hegazy, 2002).

Evolutionary algorithms provide robust search algorithms, can handle discrete activity time-cost relationships, and are applicable to large problems. However, random search is time-consuming and non-deterministic [i.e. we cannot tell when or if the optimal solution is obtained (Hegazy, 2002)]. Evolutionary algorithms are search-based and computing intensive, which generally ignores the structure of time-cost tradeoff (Jiang and Zhu, 2010).

Maximum flow and minimum cut methods are applicable to small project networks but they are ineffective while dealing with a large number of activities (Jiang and Zhu, 2010).

# Chapter 3. Methodology

The proposed TCT methodology is tailored for the most general alternative of continuous time-cost relationship, which is curvilinear. This scope can readily be modified to handle linear and multilinear cases with more efficiency. Adjusted methods for linear and multilinear time-cost relationships are presented in Sections 3.3.1 and 3.3.2, respectively.

Discrete time-cost relationship has a combinatorial nature as the search space is not continuous. However, one way to solve the discrete TCT is to convert it into multilinear. In this way, method presented for multilinear TCT can be used with a minor adjustment. The adjustment is that when a discrete activity is crashed, all of its available crash time (from the current activity duration to the next highest activity duration) is consumed at once. This may result in a suboptimal solution but still provides various alternative solutions and requires less effort. To arrive at the optimum solution in discrete TCT, pure mathematical programming methods may be more useful. The path-based integer programming solution presented by Ammar (2018) is recommended for optimal discrete TCT which has been elaborated in Section 2.3.

A major difference in the proposed methodology and most of the existing methods in the literature is the path-based approach. Lu et al. (2017) proposed a simplified version of CPM, called path-float based critical path method (PFCPM). In this method, project duration was determined from calculation of all path lengths. Subsequently, path-floats could be identified and total float of each activity could be derived from path-floats. PFCPM circumvents the backward pass analysis of classic CPM. This is particularly advantageous as once all paths are identified, effect of

any change in the duration of any activity on the schedule can be easily determined. As our method relies on recalculation and updating of project schedule in each iteration, PFCPM is utilized in each step, which considerably reduces the effort required for scheduling.

## 3.1 Assumptions and Inputs

The following assumptions have been made for the proposed methodology:

- Each activity is initialized at the most relaxed option (normal).
- All redundant precedence relationships (arrows) in the AON have been removed.
- All values (activity durations and costs) are deterministic.
- Resource requirements are satisfied in each iteration on all the activities in the project. There are no resource availability constraints, and they are fully represented in input AON as per the method by Lu and Li (2003), elaborated in Appendix A.
- Time-cost relationship of all activities is continuous curvilinear. This assumption is modified for linear and multilinear algorithms (Sections 3.3.1 and 3.3.2).
- Duration of each activity can only take integer values (for practical purposes).
- Precedence relationships of activities are sufficiently specified prior to TCT analysis and remain unchanged during analysis. In other words, the AON network is defined and not subject to change in ensuing analysis.
- AON network has only finish-to-start (FS) precedence relationships without lags. Schemes to transform non-FS relationships with lags into FS relationships without lags, by Lu and Lam (2009), are presented in Appendix B.

Many existing methods for TCT rely on the assumption of convex time-cost relationship. However, our proposed method is not limited to such an assumption and can handle both convex and concave or a curve with convex and concave parts.

The following inputs are required:

- Crashing data for activities, which include the time-cost relationships of all activities.
- Precedence relationships of activities.

## 3.2 Path Finding

The first step in the proposed method is to find all possible paths from start to finish in the AON network. For small project networks, this task can be easily done manually, by visual inspection. However, for large-sized complex networks, identification of each existing path can become challenging, if not practically impossible. To find an automated solution, methods in graph theory can be used. If each node represents a vertex and each precedence relationship arrow represents a directed edge, an AON network can be interpreted as a directed acyclic graph (De et al., 1995). Therefore, the problem of finding all paths from start to finish in an AON network is equal to the problem of finding all simple paths between two vertices. A path is called simple if no vertex is traversed more than once (Danielson, 1968). Several algorithms have been proposed to tackle the path finding problem. It must be noted that some algorithms do not produce all of the possible paths and focus on finding minimum cost paths which are not useful for our problem.

Two essentially different techniques exist for path finding: the matrix technique and the routing technique (Fratta and Montanari, 1975). An example of the routing technique is provided by Kroft (1967) where at first, a string of all vertices with

their neighbors denoted in parenthesis right next to them was constructed (with start vertex and finish vertex being the first and last vertices in the string). Subsequently, an algorithm which consists of backtracking was employed to find all the existing paths. An example of matrix technique is provided by Danielson (1968) where a modified adjacency matrix was introduced to find all possible paths. Rubin (1978) presented a basic and time-consuming algorithm which tested all the combinations of vertices comprising a path. If the edges connecting the vertices of a particular combination existed in AON, then that path was recorded as an existing path. It is obvious that this method is not particularly efficient. Fratta and Montanari (1975) provided another more efficient method which finds all simple paths by a vertex elimination technique based on matrix inversion.

The routing technique is useful for enumerating all simple paths between a pair of vertices. On the other hand, the matrix technique can provide all existing paths in the network between any pair of vertices. Since in the proposed methodology, we are only interested in finding the paths between start and finish nodes (only a pair of vertices), it is recommended to use the routing technique (Kroft, 1967) for practical large-sized project networks (100 or more activities). It must be noted that the routing technique can become exhaustive if the graph contains non-simple paths (Fratta and Montanari, 1975), which is not the case for AON networks.

## 3.3 Proposed Methodology

To handle curvilinear time-cost relationships, the nonlinear curve is replaced by straight lines for all practical purposes (Ahuja et al., 1994). Each line represents one time-unit, i.e. day or hour (Figure 3.1). It can be argued that by this assumption, the optimality of the solution is lost. However, in practice, it does not make much difference, as time-unit can be adjusted.

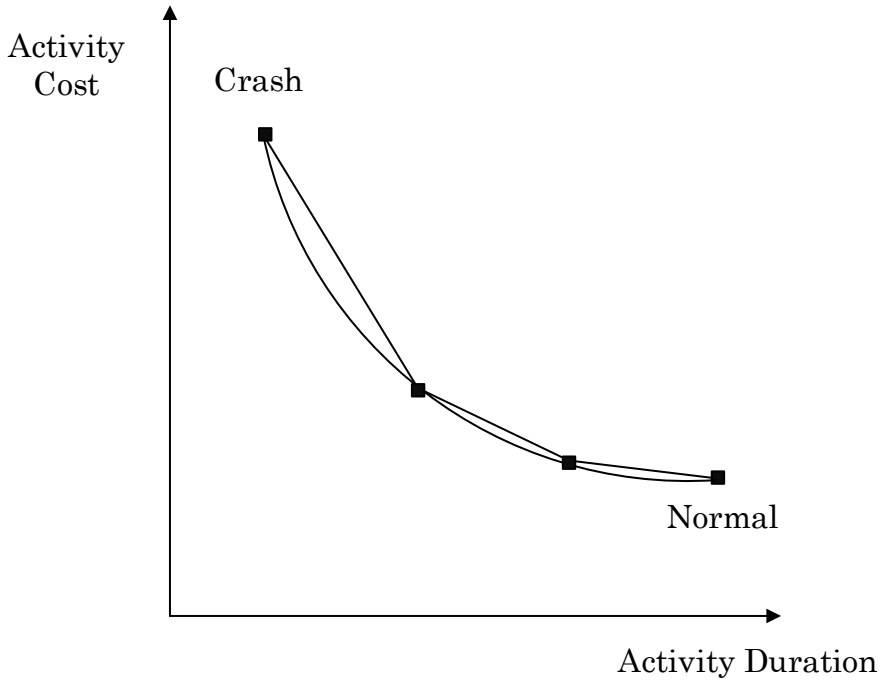



Figure 3.1 Approximation of time-cost curve

Consider an AON network having $N$ activities and $M$ paths. Let $A$ be the set of all activities which includes $a_1, a_2, \ldots, a_N$. Let $P$ be the set of all paths which includes $p_1, p_2, \ldots, p_M$. A path consists of activities ordered from start to finish. For instance, if $p_1$ consists of activities $a_1$, then $a_4$, then $a_7$, it is denoted as $p_1 = \{a_1, a_4, a_7\}$. For each activity $a_i$, let $d_i$ be the variable representing the current duration of the activity. $C_i(d_i)$ is defined as the direct cost function of the activity $a_i$, which represents the cost of the activity for duration $d_i$. $d_{in}$ and $c_{in}$ represent normal duration and normal cost of the activity and $d_{ic}$ and $c_{ic}$ represent crash duration and crash cost of the activity. In other words:

$$c_{in} = C_i(d_{in}) \tag{3.1}$$

$$c_{ic} = C_i(d_{ic}) \tag{3.2}$$

Note that the cost considered in $C_i$ *(*$d_i$*)* includes any cost that can be attributed to a particular activity. Costs that cannot be attributed to a particular activity will be considered as indirect cost.

The proposed method consists of several steps divided into two parts. The first part is the initialization of the problem (step 1) and the second part is the iterative crashing cycles (step 2 to 8). Steps include:

1. Determine the number of paths *(M), P* and the set of activities on each path using methods explained in Section 3.2.
2. Calculate the available crash time *(AC)* for each activity using Eq. 3.3. Calculate the cost slope *(S)* for each activity having available crash time using Eq. 3.4.

$$AC_i = d_i - d_{ic};\ \forall a_i \in A \tag{3.3}$$

$$S_i = C_i(d_i) - C_i(d_i - 1);\ \forall a_i \in A, AC_i \neq 0 \tag{3.4}$$

3. Use current duration of activities *(*$d_i$*)* on each path to calculate the path length *(PL)* of each path using Eq. 3.5.

$$PL_j = \sum_{a_i \in p_j} d_i\ ; \forall p_j \in P \tag{3.5}$$

4. Critical path length *(*$PL_{cr}$*)* is equal to the largest path length (Eq. 3.6). Let $P_{cr}$ be the set of critical path(s) which is defined by Eq. 3.7.

$$PL_{cr} = \max(PL_1, PL_2, \dots, PL_M) \tag{3.6}$$

$$P_{cr} = \left\{\forall p_j \in P \middle|\ PL_j = PL_{cr}\right\} \tag{3.7}$$

5. Determine the project duration (which is equal to $PL_{cr}$) and total cost. Total cost is the sum of direct costs of all activities plus indirect cost.

6. In this step, the set of activities for crashing in the current cycle must be selected. To ensure the minimum direct cost is added in the crashing cycle, the following IP formulation will be used. If no solution can be obtained, the crashing cycles must be terminated. $x_i$ is a binary variable representing if an activity should be crashed or not (1 for crashing, 0 for not crashing).

Minimize
$$\sum_{a_i \in P_{cr}} S_i \times x_i \,; AC_i \neq 0 \tag{3.8}$$

Subject to
$$x_i = 0,1; \forall a_i \in P_{cr}, AC_i \neq 0 \tag{3.9}$$

$$\sum_{a_i \in p_j} x_i \geq 1; \forall p_j \in P_{cr}, AC_i \neq 0 \tag{3.10}$$

7. For activities that have been selected for crashing ($x_i=1$), reduce the current duration ($d_i$) by one time-unit.
8. Go to step 2 for the next cycle.

Note that after the first cycle, available crash times and cost slopes will be updated in step 2 only for crashed activities. These values will remain the same for all other activities. In addition, only path length of critical path(s) requires updating in each iteration (step 3). Eq. 3.8 represents the combined cost slope of all activities that are chosen to be crashed. To reduce the total project duration, all critical paths must be crashed simultaneously. Eq. 3.10 ensures that at least one activity from each critical path is crashed.

Figure 3.2 shows the flowchart of the proposed algorithm. Each iteration of this algorithm provides a feasible solution, which is a project duration with a minimized corresponding total cost. After termination of algorithm, global optimums (minimum total cost and minimum total duration) can be selected from the set of feasible solutions.

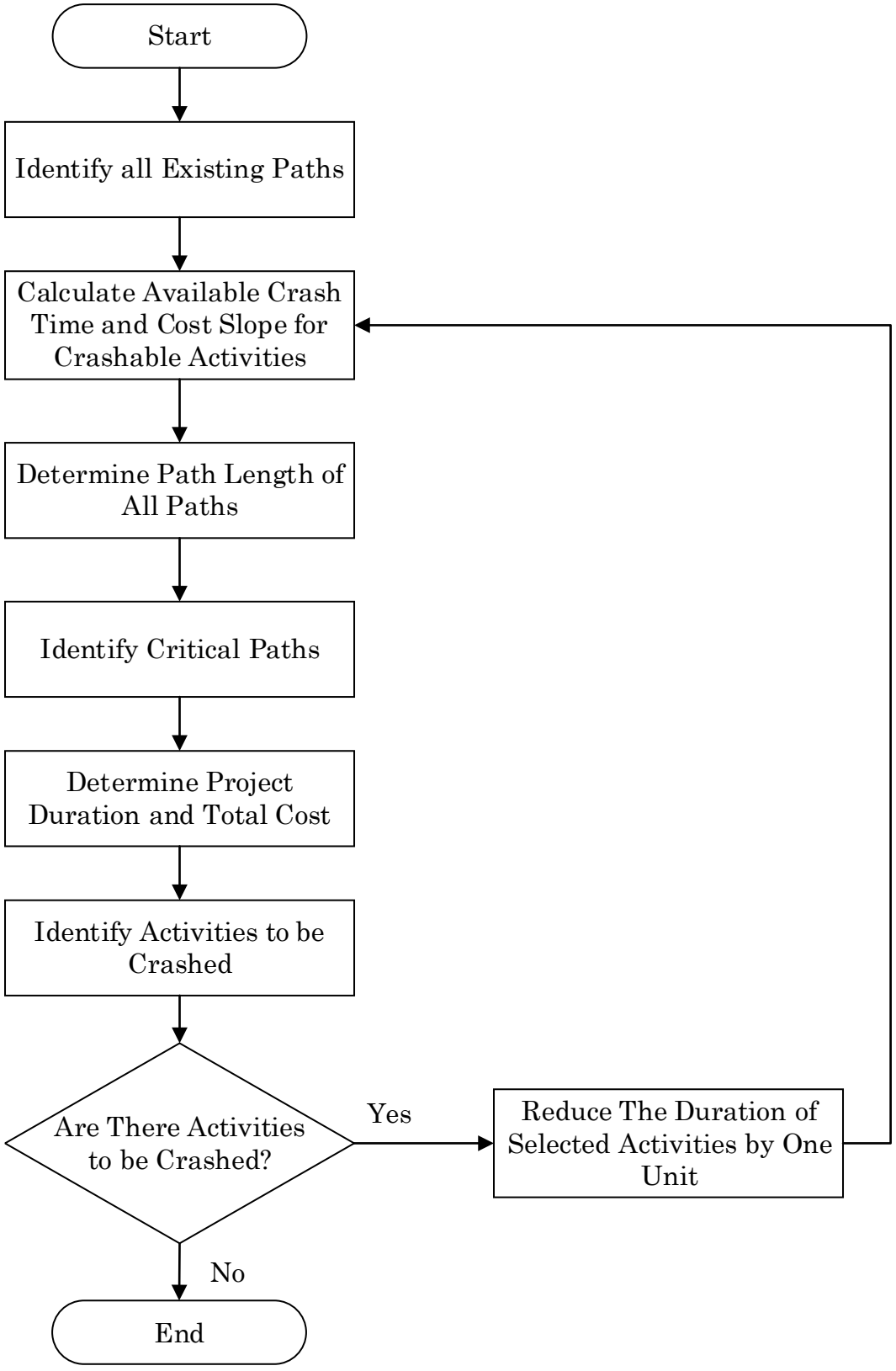



Figure 3.2 Flowchart of proposed method for curvilinear time-cost relationship

### 3.3.1 Adjusted Method Subject to Linear Time-Cost Relationship

The proposed algorithm can be streamlined to handle linear activity time-cost relationships (Figure 3.3). In this case, as the cost slope *(S)* of each activity is constant, it should only be calculated once and does not need to be updated in each iteration. Nomenclature and definitions are the same as before.

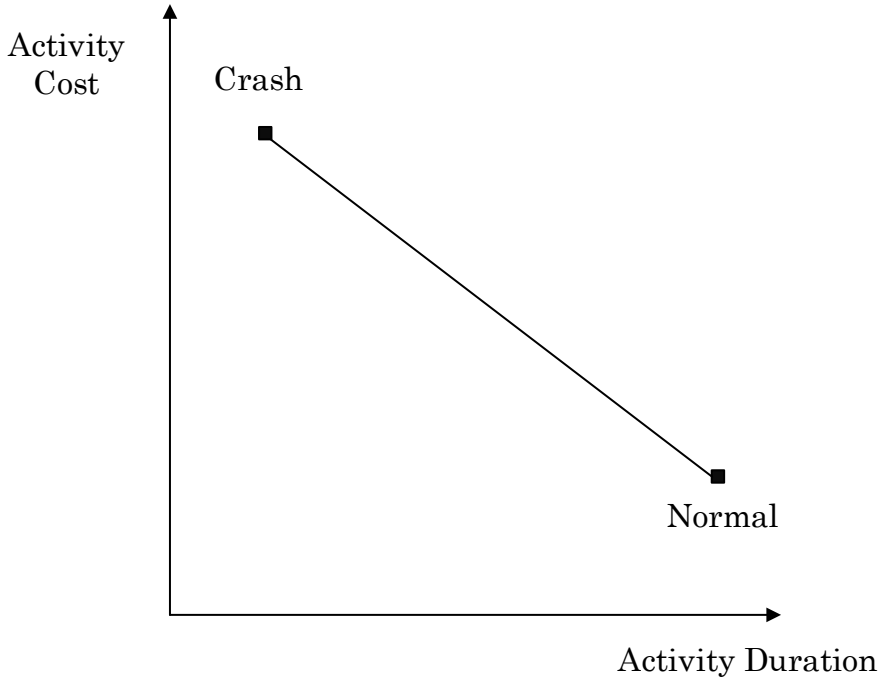



Figure 3.3 Linear time-cost relationship

The proposed method consists of several steps divided into two parts. The first part is the initialization of the problem (step 1 and 2) and the second part is the iterative crashing cycles (step 3 to 9). Steps include:

1. Determine the number of paths *(M), P* and the set of activities on each path using methods explained in Section 3.2.

2. Calculate the available crash time *(AC)* for each activity using Eq. 3.11. Calculate the cost slope *(S)* for each activity having available crash time using Eq. 3.12.

$$AC_i = d_{in} - d_{ic};\ \forall a_i \in A \tag{3.11}$$

$$S_i = \frac{c_{ic} - c_{in}}{AC_i};\ \forall a_i \in A, AC_i \neq 0 \tag{3.12}$$

3. Use current duration of activities *($d_i$)* on each path to calculate the path length *(PL)* of each path using Eq. 3.13.

$$PL_j = \sum_{a_i \in p_j} d_i\ ; \forall p_j \in P \tag{3.13}$$

4. Critical path length *($PL_{cr}$)* is equal to the largest path length (Eq. 3.14). Let $P_{cr}$ be the set of critical path(s) which is defined by Eq. 3.15.

$$PL_{cr} = \max(PL_1, PL_2, \dots, PL_M) \tag{3.14}$$

$$P_{cr} = \left\{\forall p_j \in P \middle|\ PL_j = PL_{cr}\right\} \tag{3.15}$$

5. Let $P_{nc}$ be the set of non-critical paths (Eq. 3.16). Determine next-to-critical path length ($PL_{ntc}$) using Eq. 3.17. Determine path-float of next-to-critical path ($PF_{ntc}$) using Eq. 3.18.

$$P_{nc} = P - P_{cr} \tag{3.16}$$

$$PL_{ntc} = \max\left\{PL_j \middle|\ p_j \in P_{nc}\right\} \tag{3.17}$$

$$PF_{ntc} = PL_{cr} - PL_{ntc} \tag{3.18}$$

6. Determine the project duration (which is equal to $PL_{cr}$) and total cost. Total cost is the sum of direct costs of all activities plus indirect cost.

7. In this step, the set of activities for crashing in the current cycle must be selected. To ensure the minimum direct cost is added in the crashing cycle, the following IP formulation will be used. If no solution can be obtained, the crashing cycles must be terminated. $x_i$ is a binary variable representing if an activity should be crashed or not (1 for crashing, 0 for not crashing).

Minimize $$\sum_{a_i \in P_{cr}} S_i \times x_i \,; AC_i \neq 0 \tag{3.19}$$

Subject to $$x_i = 0,1; \forall a_i \in P_{cr}, AC_i \neq 0 \tag{3.20}$$

$$\sum_{a_i \in p_j} x_i \geq 1; \forall p_j \in P_{cr}, AC_i \neq 0 \tag{3.21}$$

8. Determine the crash duration ($CD$) using Eq. 3.22. For activities that have been selected for crashing ($x_i=1$), reduce the current duration ($d_i$) by the amount of $CD$. Update $AC$ for activities that have been crashed using Eq. 3.23.

$$CD = min \begin{cases} PF_{ntc} \\ min\,\{AC_i |\, x_i = 1\} \end{cases} \tag{3.22}$$

$$AC_i = d_i - d_{ic};\ \forall a_i \in A, x_i = 1 \tag{3.23}$$

9. Go to step 3 for the next cycle.

Figure 3.4 shows the flowchart of the proposed algorithm.

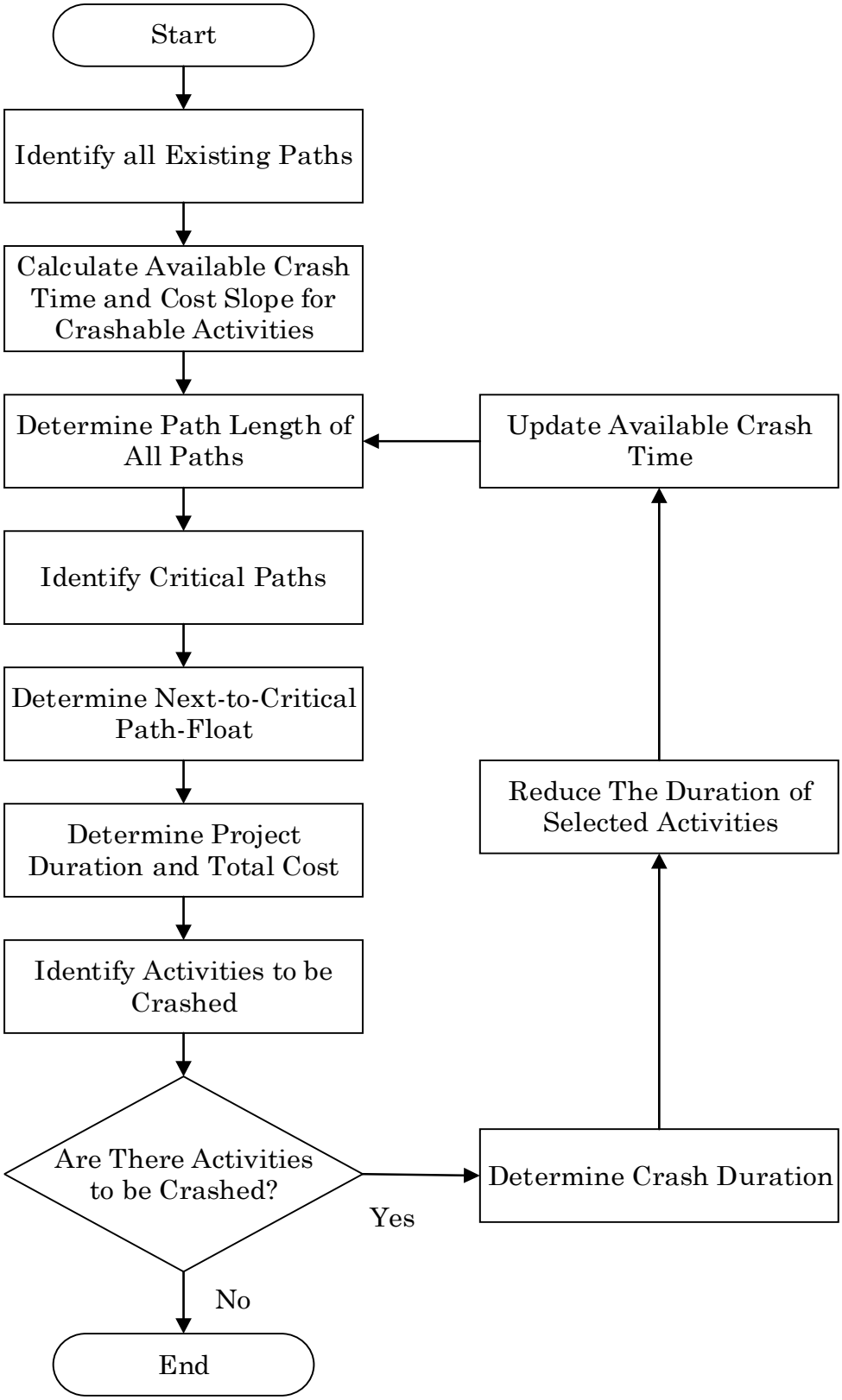



Figure 3.4 Flowchart of proposed method for linear time-cost relationship

### 3.3.2 Adjusted Method Subject to Multilinear Time-Cost Relationship

The proposed algorithm can be simplified to handle multilinear activity time-cost relationships. For each activity $a_i$, time-cost curve is divided into several sections ($K_i$ sections), each having a constant cost slope. Let $D_i$ be the set of durations for activity $a_i$ at which the cost slope changes and includes $d_{i0}, d_{i1}, \ldots, d_{iKi}$ (Figure 3.5). Note that $d_{i0}$ is the same as $d_{in}$ and $d_{iKi}$ is the same as $d_{ic}$.

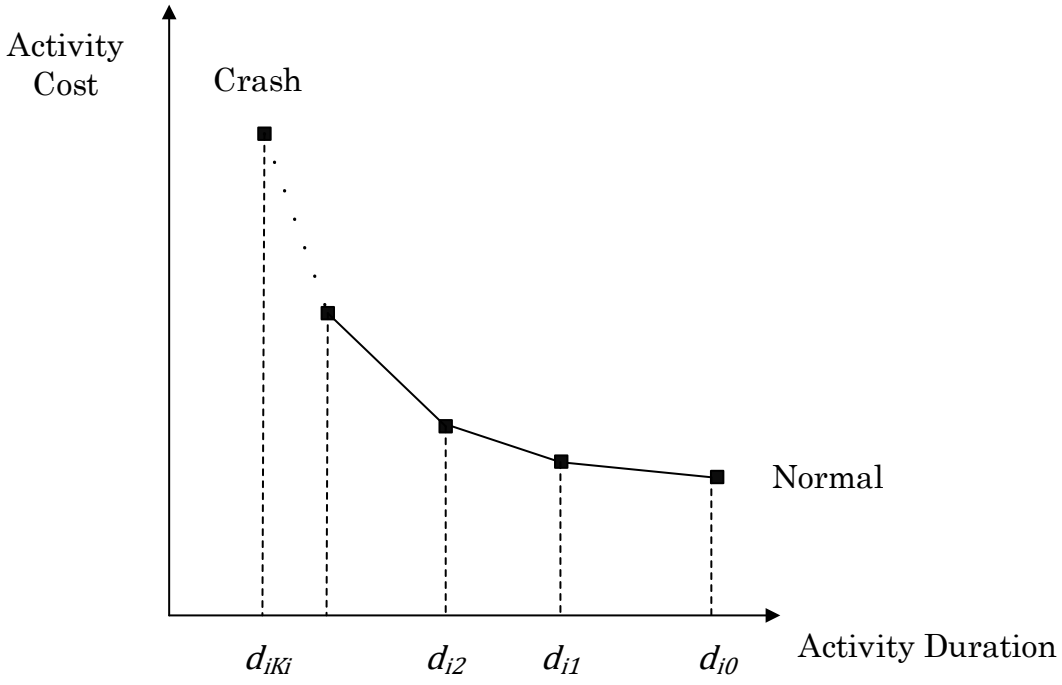



Figure 3.5 Multilinear time-cost relationship

The proposed method consists of several steps divided into two parts. The first part is the initialization of the problem (step 1) and the second part is the iterative crashing cycles (step 2 to 9). Steps include:

1. Determine the number of paths *(M), P* and the set of activities on each path using methods explained in Section 3.2.

2. $d_{ih}$ is defined as the largest value in $D_i$ that is smaller than the current duration of activity (Eq. 3.24). Note that $d_i$ is the current duration of activity $a_i$. Calculate the available crash time *(AC)* for each activity using Eq. 3.25. Calculate the cost slope *(S)* for each activity having available crash time using Eq. 3.26.

$$d_{ih} = \max\{d_{il} | \, d_{il} \in D_i, d_{il} < d_i\} \tag{3.24}$$

$$AC_i = d_i - d_{ih};\ \forall a_i \in A \tag{3.25}$$

$$S_i = \frac{C_i(d_i) - C_i(d_{ih})}{AC_i};\ \forall a_i \in A, AC_i \neq 0 \tag{3.26}$$

3. Use current duration of activities *($d_i$)* on each path to calculate the path length *(PL)* of each path using Eq. 3.27.

$$PL_j = \sum_{a_i \in p_j} d_i \,; \forall p_j \in P \tag{3.27}$$

4. Critical path length *($PL_{cr}$)* is equal to the largest path length (Eq. 3.28). Let $P_{cr}$ be the set of critical path(s) which is defined by Eq. 3.29.

$$PL_{cr} = \max(PL_1, PL_2, \dots, PL_M) \tag{3.28}$$

$$P_{cr} = \left\{\forall p_j \in P \middle| \, PL_j = PL_{cr}\right\} \tag{3.29}$$

5. Let $P_{nc}$ be the set of non-critical paths (Eq. 3.30). Determine next-to-critical path length ($PL_{ntc}$) using Eq. 3.31. Determine path-float of next-to-critical path ($PF_{ntc}$) using Eq. 3.32.

$$P_{nc} = P - P_{cr} \tag{3.30}$$

$$PL_{ntc} = \max\left\{PL_j \middle| \, p_j \in P_{nc}\right\} \tag{3.31}$$

$$PF_{ntc} = PL_{cr} - PL_{ntc} \tag{3.32}$$

6. Determine the project duration (which is equal to $PL_{cr}$) and total cost. Total cost is the sum of direct costs of all activities plus indirect cost.
7. In this step, the set of activities for crashing in the current cycle must be selected. To ensure the minimum direct cost is added in the crashing cycle, the following IP formulation will be used. If no solution can be obtained, the crashing cycles must be terminated. $x_i$ is a binary variable representing if an activity should be crashed or not (1 for crashing, 0 for not crashing).

Minimize
$$\sum_{a_i \in P_{cr}} S_i \times x_i \,; AC_i \neq 0 \tag{3.33}$$

Subject to
$$x_i = 0,1; \forall a_i \in P_{cr}, AC_i \neq 0 \tag{3.34}$$

$$\sum_{a_i \in p_j} x_i \geq 1; \forall p_j \in P_{cr}, AC_i \neq 0 \tag{3.35}$$

8. Determine the crash duration ($CD$) using Eq. 3.36. For activities that have been selected for crashing ($x_i=1$), reduce the current duration ($d_i$) by the amount of $CD$.

$$CD = min \begin{cases} PF_{ntc} \\ min\,\{AC_i |\, x_i = 1\} \end{cases} \tag{3.36}$$

9. Go to step 2 for the next cycle.

Figure 3.6 shows the flowchart of the proposed algorithm. Note that cost slope of a crashed activity in an iteration will only change if the activity has used all of the crash time available for its corresponding section.

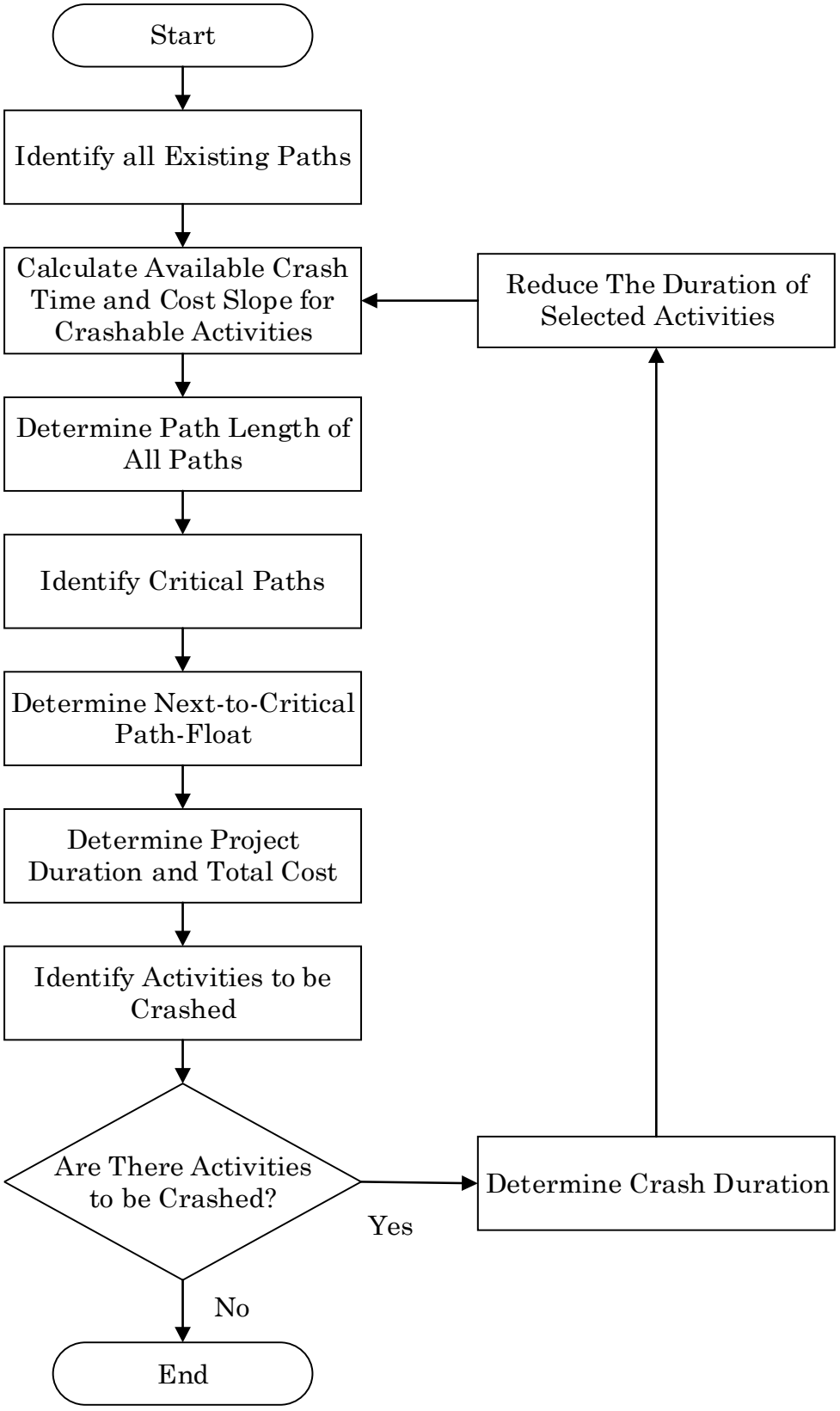



Figure 3.6 Flowchart of proposed method for multilinear time-cost relationship

## 3.4 Analytical Discussion

All concepts elaborated in this section apply to all three of the proposed methodologies (linear, multilinear, and curvilinear) unless stated otherwise.

Suppose there are only two paths in the AON network named $p_1$ and $p_2$ with $p_1$ having the longer path length. We want to limit the project duration to $\gamma$ such that $p_1 > \gamma > p_2$ (Figure 3.7). In this case, any activity that is part of $p_2$ but not a part of $p_1$ does not need to be considered for crashing. The obvious reason is that crashing such an activity reduces the path length of $p_2$ but does not reduce the project duration (in this case, $p_1$) and causes unnecessary added cost.

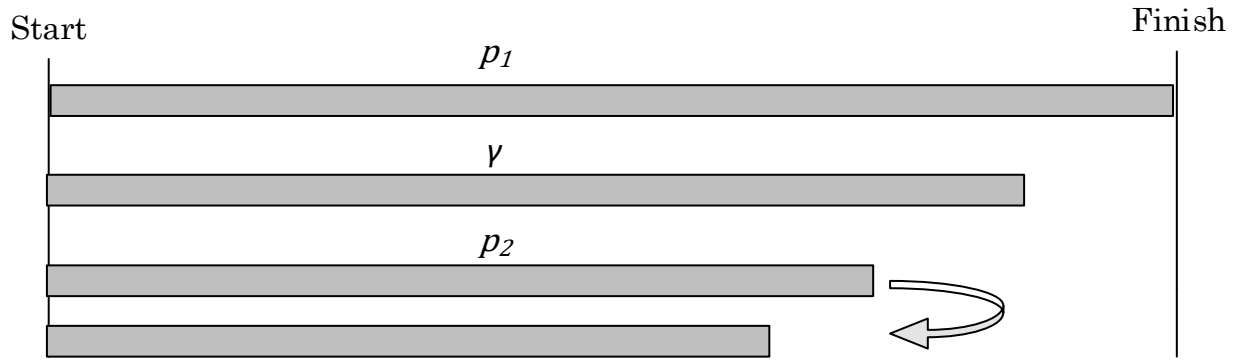



Figure 3.7 Unnecessary crashing

The proposed algorithm extends this concept to a project network comprising of numerous paths by only considering critical path(s) in each iteration. As a result, unnecessary crashing and the resultant added cost are avoided on a general project-wide scale, which contributes to optimality of the obtained results. In addition, search space is substantially reduced by considering only a part of the network as opposed to all of it.

To ensure that the best set of activities selected for crashing does not change during an iteration, search space (critical paths) and its characteristics (cost slopes) must remain stable during each cycle of crashing. Therefore, the amount of crashing duration is limited by (1) next-to-critical path-float and (2) the interval in which the cost slope of selected set of activities for crashing remains the same. Next-to-critical path is the path with the second largest path length after the critical path. Next-to-critical path-float ($PF_{ntc}$) has been previously defined using Eq. 3.18 and is illustrated in Figure 3.8.


Figure 3.8 Graphical representation of next-to-critical path

The main difference between the three algorithms (curvilinear, linear, and multilinear) is due to how the aforementioned limits on the crashing duration are imposed. Regarding the curvilinear algorithm, the activity durations can only take integer values. Therefore, next-to-critical path-float cannot be less than one time-unit. Furthermore, activities are approximated by one time-unit linear segments and in each iteration, selected activities are only crashed by one day. Hence, both limits are satisfied. In the linear algorithm, crashing duration is limited by the next-to-critical path-float. Note that in this case, the cost slopes of all activities are constant in all iterations. For the multilinear algorithms, crashing duration is limited by minimum of next-to-critical path-float and available crash duration of

activities until their respective cost slopes changes. Cost slope of linear and multilinear activities does not change for larger intervals, which enables us to take larger steps in each iteration without skipping the optimal solutions.

The objective function of IP formulation represents the combined cost slope of all activities chosen to be crashed in each iteration. In other words, objective function is equal to added direct cost if we reduce the duration of the project by one time-unit. The proposed algorithm divides the TCT into several steps and in each step, added direct cost is minimized by use of IP formulation. Figure 3.9 is a simplified visual representation of this minimization with red lines showing non-optimal (i.e. having a larger combined cost slope) crashing alternatives in each iteration. Note that unnecessary crashing does not reduce the project duration and is not represented by the red lines. To summarize, the proposed algorithm searches for the lowest possible cost in a stepwise manner. The resultant project time-cost curve represents the lowest project cost corresponding to any project duration. The global optimums (lowest total cost and lowest total duration) can be selected from the time-cost curve.

As a result of this minimization, the algorithm can achieve optimum or near-optimum results (illustrated in Figure 3.10). The obtained optimum has coincided with the true theoretical optimum on the test bed cases (a linear case, and a curvilinear case) from literature (refer to Sections 4.1 and 4.2). For practical projects, theoretical optimum remains unknown, but we can infer that the obtained solution approaches the true optimum in a small margin thanks to the proposed algorithm and the nested IP. This is deemed sufficient for practical applications in construction project management.

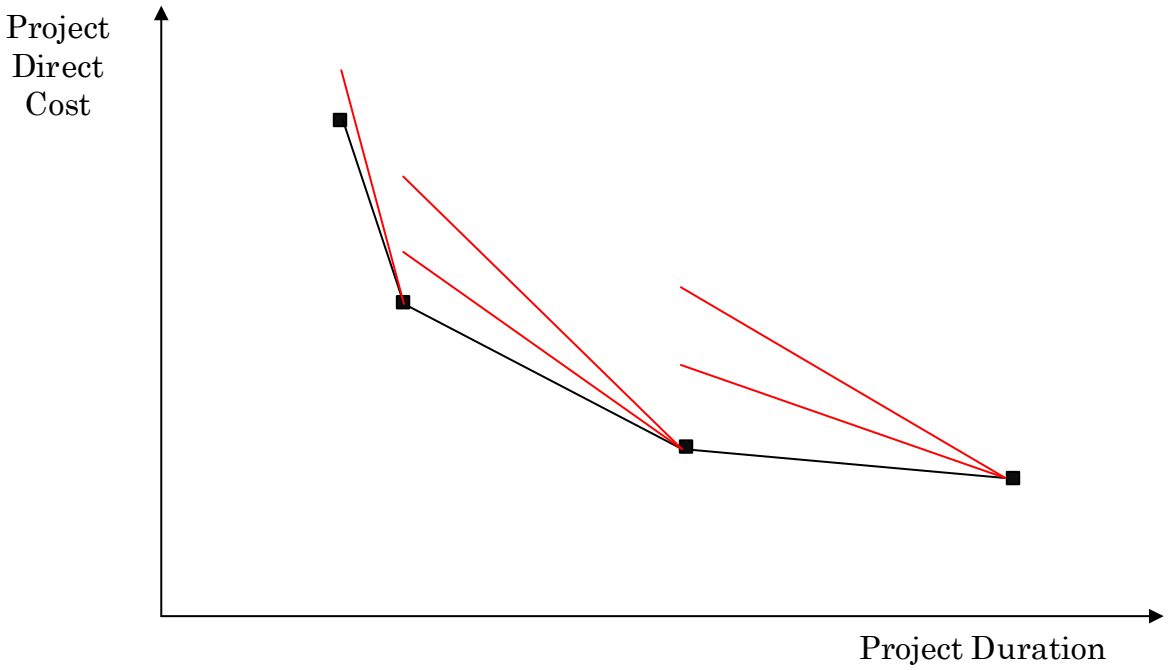



Figure 3.9 Minimization in each iteration

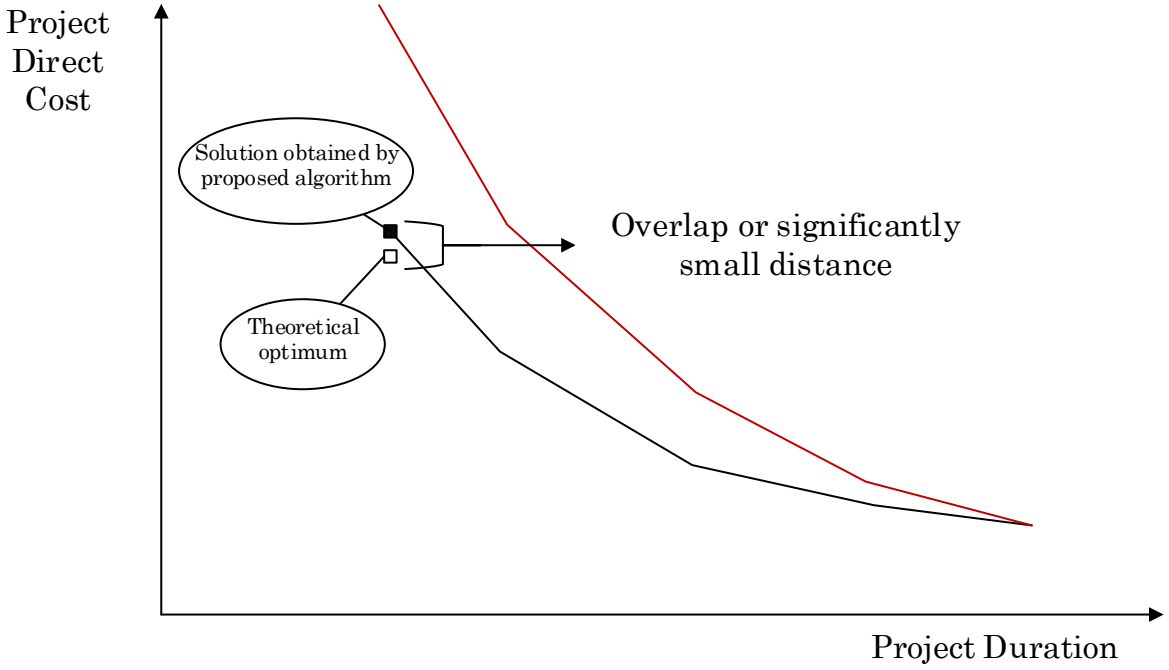
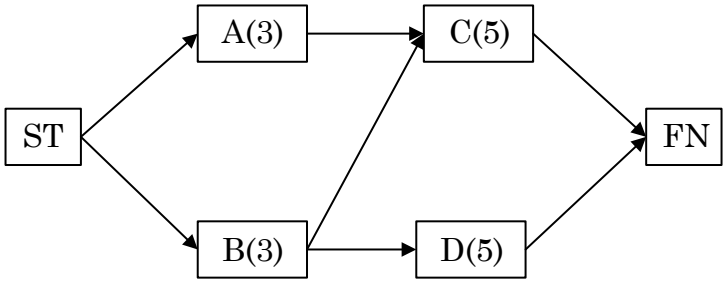
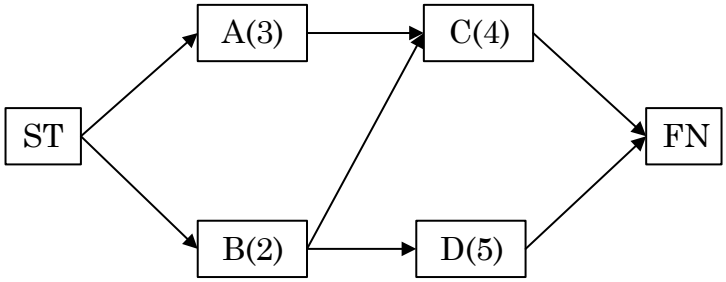



Figure 3.10 Optimality of proposed algorithm (obtained solution and theoretical optimum are expected to overlap, or their distance will be significantly small)

Eq. 3.10 represents the number of activities that are being crashed on each critical path, in each iteration. The value of this constraint must be larger than zero for all current critical paths to ensure that the project duration is reduced. If the value of this constraint is equal to one for all critical paths, it can be deduced that the path length of each critical path is reduced by the same amount. It must be noted that in each iteration, all activities in the selected set must be crashed by the same amount. As a result, if Eq. 3.10 is equal to one for all critical paths in an iteration, all current critical paths will be critical in the next iteration as well. This is mostly the case but there is an exception to it. Consider the network illustrated in Figure 3.11 with crash information presented in Table 3.1. In this network, path lengths of all three paths (AC, BC, and BD) is 8. Hence, all paths are critical and should be crashed. Based on Table 3.1, only activities B and C have non-zero available crash time. Therefore, activity C is the only option for path AC and activity B is the only option for path BD and both have to be crashed. After crashing, the new path lengths are updated as 7 days for AC, 6 days for BC, and 7 days for BD. It can be seen that path BC, which was critical at the initial stage is not critical anymore. Eq. 3.10 will take a value of two or more for a certain path if two or more activities on it are the only options with non-zero crash times on respective critical paths. In this case, these activities will be crashed in all the next iterations and analysis will stop once any of them runs out of available crash time, as this will result in a critical path that has no further crash options.




(a) AON with activity durations at initial stage (activity durations are given in parentheses)




(b) AON with activity durations after crashing (activity durations are given in parentheses)

Figure 3.11 Sample AON for analysis of Eq. 3.10

Table 3.1 Crash data for Figure 3.11

| **Activity** | **Duration (d)** | | **AC (d)** |
|---|---|---|---|
| | **Initial** | **After crash** | |
| A | 3 | 3 | 0 |
| B | 3 | 2 | 2 |
| C | 5 | 4 | 1 |
| D | 5 | 5 | 0 |

If the time-cost relationships of all activities are linear or convex (that is to say that cost slope of activity will remain the same or increase as the duration of activity decreases), the value of objective function can only increase or remain the same in each iteration. To assess this statement, we have to consider several different

scenarios. Assume C1 is a crashing cycle in the algorithm. C2 is defined as the crashing cycle right after C2. Note that there is no restriction on the two consecutive cycles of C1 and C2. In other words, C1 and C2 can be either the first, final, or intermediate crashing cycle. To compare the critical paths of these two consecutive iterations, two questions need to be answered:

- Are all the paths that were critical in C1 still critical in C2? Note that it is possible for a critical path to become non-critical after an iteration.
- Is there a critical path in C2 that was previously not critical in C1?

Based on the comparison of critical paths in C1 and C2, the scenarios are defined as presented in Table 3.2. Each scenario is examined and elaborated with an example having linear activity time-cost relationships. Note that linearity is only for simplicity and is not necessary. Examples with convex multilinear or convex curvilinear activity time-cost relationships can be readily provided.

Table 3.2 Scenarios for increasing value of objective function

| Scenario | Description |
|---|---|
| 1 | All paths that were critical in C1 are critical in C2 as well.<br>There is no new critical path in C2 that was not critical in C1. |
| 2 | At least one path that was critical in C1 is not critical in C2.<br>There is no new critical path in C2 that was not critical in C1. |
| 3 | All paths that were critical in C1 are critical in C2 as well.<br>There is a new critical path in C2 that was not critical in C1 |
| 4 | At least one path that was critical in C1 is not critical in C2.<br>There is a new critical path in C2 that was not critical in C1 |

For scenario one, the critical paths are the same in both C1 and C2. Therefore, the lowest value of objective function will be obtained in C1. For example, consider the AON network presented in Figure 3.12, with linear activity time-cost relationships and crash data of Table 3.3. Paths and path lengths are presented in Table 3.4. In

C1, paths AC and BD are critical and activities A and B will be crashed by one day with a combined cost slope of $500 per day. In C2, paths AC and BD are critical and activities A and D will be crashed by one day with a combined cost slope of $600 per day.

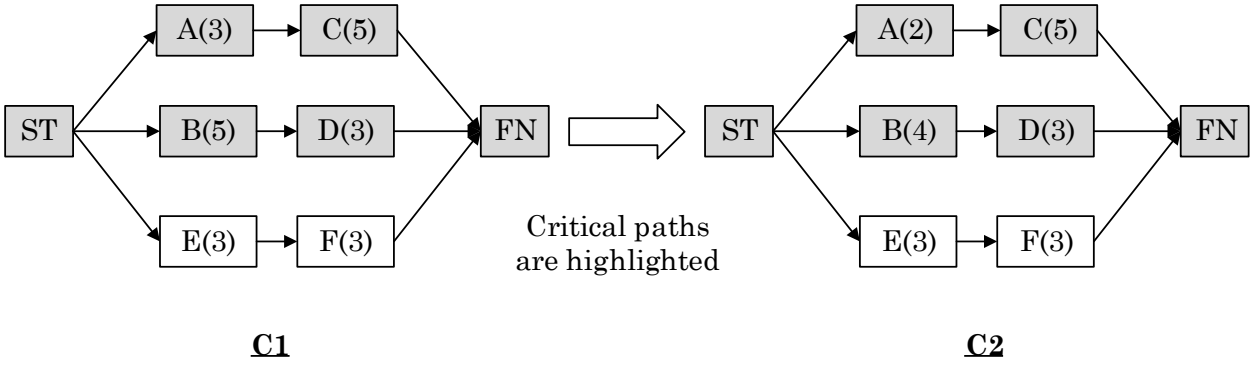



Figure 3.12 Sample AON for scenario 1

Table 3.3 Example for scenario 1 based on Figure 3.12

| | Duration (d) | | S | AC (d) | |
|---|---|---|---|---|---|
| **Activity** | **C1** | **C2** | **($/d)** | **C1** | **C2** |
| A | 3 | 2 | 200 | 2 | 1 |
| B | 5 | 4 | 300 | 1 | 0 |
| C | 5 | 5 | 200 | 1 | 1 |
| D | 3 | 3 | 400 | 1 | 1 |
| E | 3 | 3 | 200 | 1 | 1 |
| F | 3 | 3 | 250 | 1 | 1 |

Table 3.4 Paths lengths for scenario 1 with (critical paths highlighted)

| Path | Path Length (d) | |
|---|---|---|
| | C1 | C2 |
| AC | 8 | 7 |
| BD | 8 | 7 |
| EF | 6 | 6 |

For scenario two, the activities that were crashed in C1 on the path that has become non-critical in C2 will have to be crashed again in C2 (previously explained) and are part of the value of objective function in both iterations. The rest of the analysis is similar to scenario one. For example, consider the AON network presented in Figure 3.13, with linear activity time-cost relationships and crash data of Table 3.5. Paths and path lengths are presented in Table 3.6. In C1, paths AC, BGC, BD, and EF are critical and activities B, C, and E will be crashed by one day with a combined cost slope of $700 per day. In C2, path BGC has become non-critical and paths AC, BD, and EF are critical. Therefore, in C2, activities B, C, and F will be crashed by one day with a combined cost slope of $750 per day.

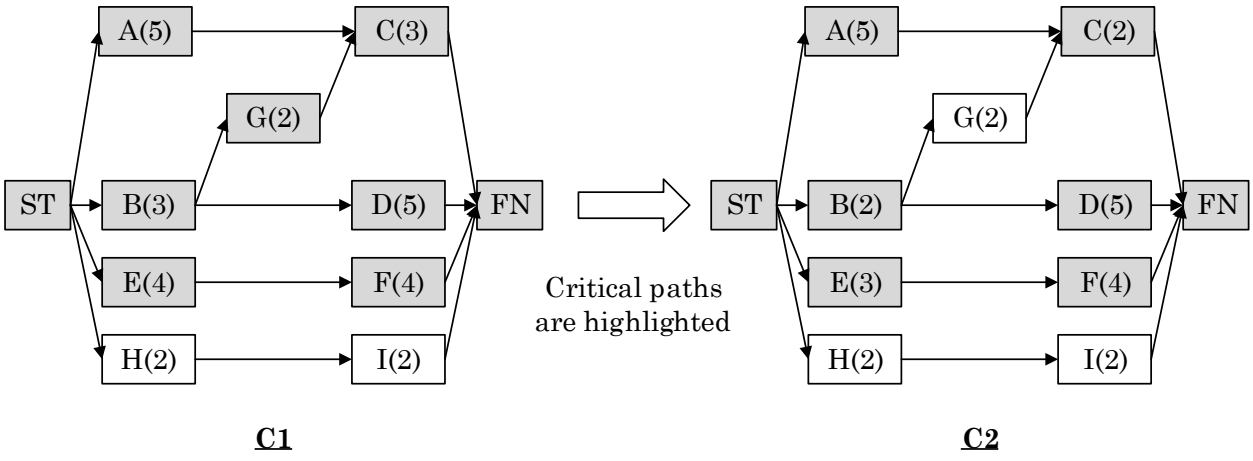



Figure 3.13 Sample AON for scenario 2

Table 3.5 Example for scenario 2 based on Figure 3.13

| | Duration (d) | | S | AC (d) | |
|---|---|---|---|---|---|
| **Activity** | **C1** | **C2** | **($/d)** | **C1** | **C2** |
| A | 5 | 5 | - | 0 | 0 |
| B | 3 | 2 | 300 | 2 | 1 |
| C | 3 | 2 | 200 | 2 | 1 |
| D | 5 | 5 | - | 0 | 0 |
| E | 4 | 3 | 200 | 1 | 0 |
| F | 4 | 4 | 250 | 1 | 1 |
| G | 2 | 2 | 100 | 1 | 1 |
| H | 2 | 2 | 200 | 1 | 1 |
| I | 2 | 2 | 300 | 1 | 1 |

Table 3.6 Paths lengths for scenario 2 with (critical paths highlighted)

| Path | Path Length (d) | |
|---|---|---|
| | C1 | C2 |
| AC | 8 | 7 |
| BGC | 8 | 6 |
| BD | 8 | 7 |
| EF | 8 | 7 |
| HI | 4 | 4 |

Scenario three needs to be examined in two situations. If the activity that is to be crashed on the newly formed critical path in C2 is part of the critical paths in C1, the objective function of C2 cannot be smaller than C1. Otherwise, that set of activities would have been selected in C1. If the activity that is to be crashed on the newly formed critical path in C2 is not part of the critical paths in C1, C1 critical paths still need to be crashed. Therefore, the part of C2's objective function corresponding to C1 critical paths can only increase (similar to scenario 1). In addition, the selected activity that is not part of the C1 critical paths has a cost

slope which will be added to the objective function in C2. For example, consider the AON network presented in Figure 3.14, with linear activity time-cost relationships and crash data of Table 3.7. Paths and path lengths are presented in Table 3.8. In C1, paths AC and BD are critical and activities A and B will be crashed by one day with a combined cost slope of $500 per day. In C2, paths AC, BD, and EF are critical and activities A, D and E will be crashed by one day with a combined cost slope of $800 per day.

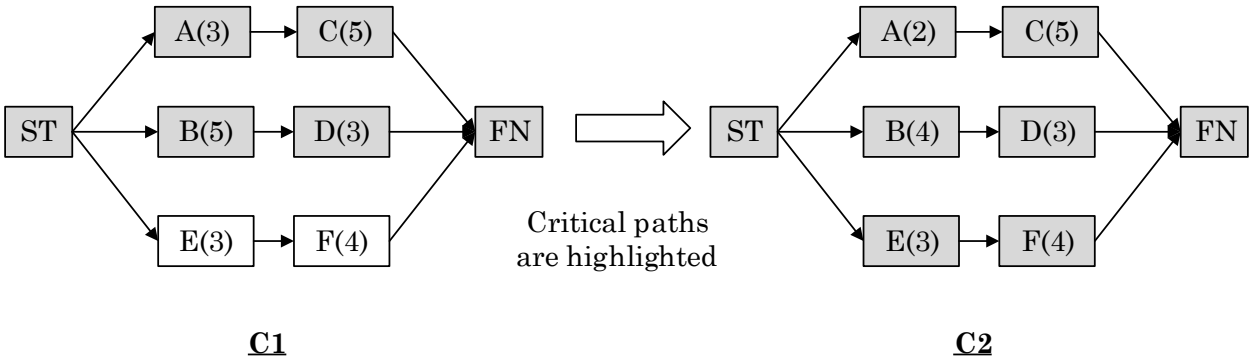



Figure 3.14 Sample AON for scenario 3

Table 3.7 Example for scenario three based on Figure 3.14

| | Duration (d) | | S | AC (d) | |
|---|---|---|---|---|---|
| **Activity** | **C1** | **C2** | **($/d)** | **C1** | **C2** |
| A | 3 | 2 | 200 | 2 | 1 |
| B | 5 | 4 | 300 | 1 | 0 |
| C | 5 | 5 | 200 | 1 | 1 |
| D | 3 | 3 | 400 | 1 | 1 |
| E | 3 | 3 | 200 | 1 | 1 |
| F | 4 | 4 | 250 | 1 | 1 |

Table 3.8 Paths lengths for scenario 3 with (critical paths highlighted)

| Path | Path Length (d) | |
|---|---|---|
| | C1 | C2 |
| AC | 8 | 7 |
| BD | 8 | 7 |
| EF | 7 | 7 |

For scenario four, the activities that were crashed in C1 on the path that has become non-critical in C2 will have to be crashed again in C2 (previously explained) and are part of the value of objective function in both iterations. The rest of the analysis is similar to scenario three. For example, consider the AON network presented in Figure 3.15, with linear activity time-cost relationships and crash data of Table 3.9. Paths and path lengths are presented in Table 3.10. In C1, paths AC, BGC, BD, and EF are critical and activities B, C, and E will be crashed by one day with a combined cost slope of $700 per day. In C2, path BGC has become non-critical and paths AC, BD, EF and HI are critical. Therefore, in C2, activities B, C, F, and H will be crashed by one day with a combined cost slope of $950 per day.

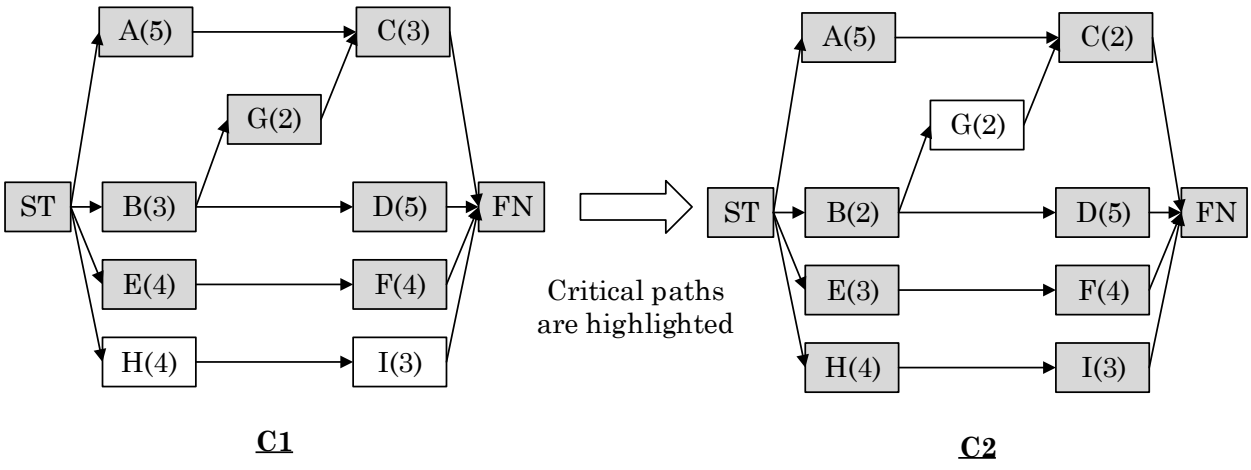



Figure 3.15 Sample AON for scenario 4

Table 3.9 Example for scenario four based on Figure 3.15

| Activity | Duration (d) | | S ($/d) | AC (d) | |
|---|---|---|---|---|---|
| | C1 | C2 | | C1 | C2 |
| A | 5 | 5 | - | 0 | 0 |
| B | 3 | 2 | 300 | 2 | 1 |
| C | 3 | 2 | 200 | 2 | 1 |
| D | 5 | 5 | - | 0 | 0 |
| E | 4 | 3 | 200 | 1 | 0 |
| F | 4 | 4 | 250 | 1 | 1 |
| G | 2 | 2 | 100 | 1 | 1 |
| H | 4 | 4 | 200 | 1 | 1 |
| I | 3 | 3 | 300 | 1 | 1 |

Table 3.10 Paths lengths for scenario 4 with (critical paths highlighted)

| Path | Path Length (d) | |
|---|---|---|
| | C1 | C2 |
| AC | 8 | 7 |
| BGC | 8 | 6 |
| BD | 8 | 7 |
| EF | 8 | 7 |
| HI | 7 | 7 |

If the time-cost relationship of an activity is concave, it means that the cost slope of that activity can decrease in two consecutive iterations. In this case, increase in the value of objective function cannot be ensured.

Indirect cost is defined as costs allocated to a project, which cannot be attributed to any certain activity. Indirect cost increases with the project duration (Bettemir and Birgönül, 2017). Indirect cost is divided into two parts; (1) a part that does not depend on the project duration (fixed) such as project office expenses and site installations, and (2) a part that increases with project duration (variable) such as

salaries of supervisors, medical, and safety personnel (Hegazy, 2002). Regarding TCT, most literature, such as Moussourakis and Haksever (2009) and Ahuja et al. (1994), ignore the fixed indirect cost and the variable part is modelled as a linear function of project duration. Some references, such as Ammar (2018), model both fixed and variable parts of indirect cost, assuming a constant indirect cost slope (IS) for the variable part. This method is illustrated in Figure 3.16. However, there is a more general realistic method to model indirect cost slope, which is to consider three different indirect cost slopes (illustrated in Figure 3.17). This method has not been used in TCT literature. As an example of a change in indirect cost slope, safety related costs can be considered. An expedited schedule gives rise to additional safety hazards and increases the safety cost. All three indirect cost models can be easily accommodated by the proposed method. If a single constant indirect cost slope (Figure 3.16) is selected, no change is required in the proposed methodology. However, if three indirect costs slopes are selected, a new step must be added in certain iterations. In this case, we must examine if the indirect cost slope has changed during a cycle. If so, direct cost, indirect cost, and the total cost corresponding to the project duration where the indirect cost slope changes must be recorded. Note that there is no need to stop and reselect the activities for crashing. The reason for recording this instance as an alternative solution is the effect that combined direct cost slope and indirect cost slope have on total cost slope (explained in the next paragraph).

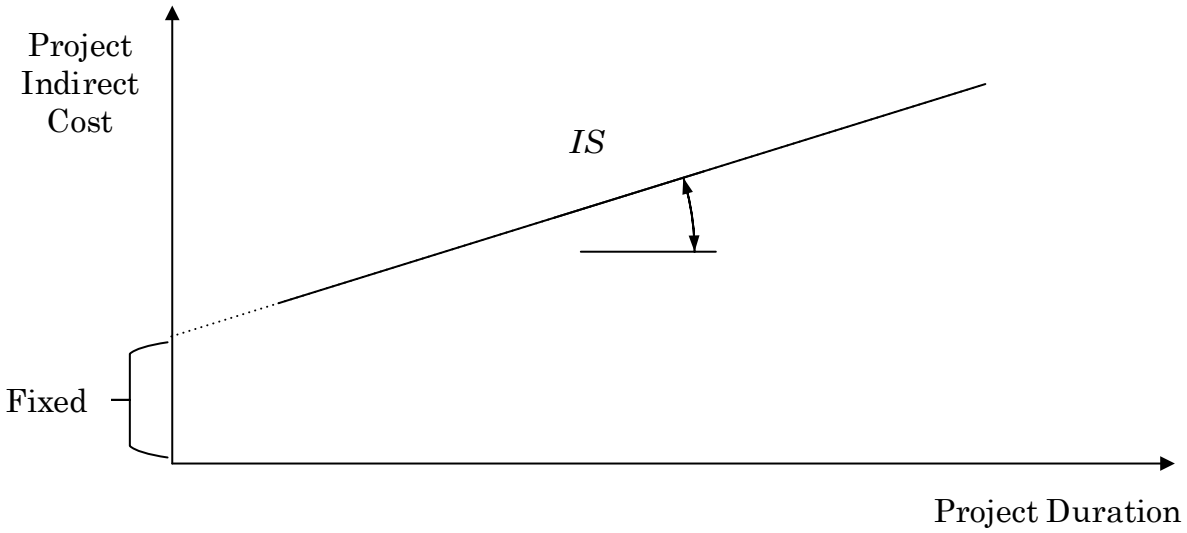



Figure 3.16 Single constant indirect cost slope

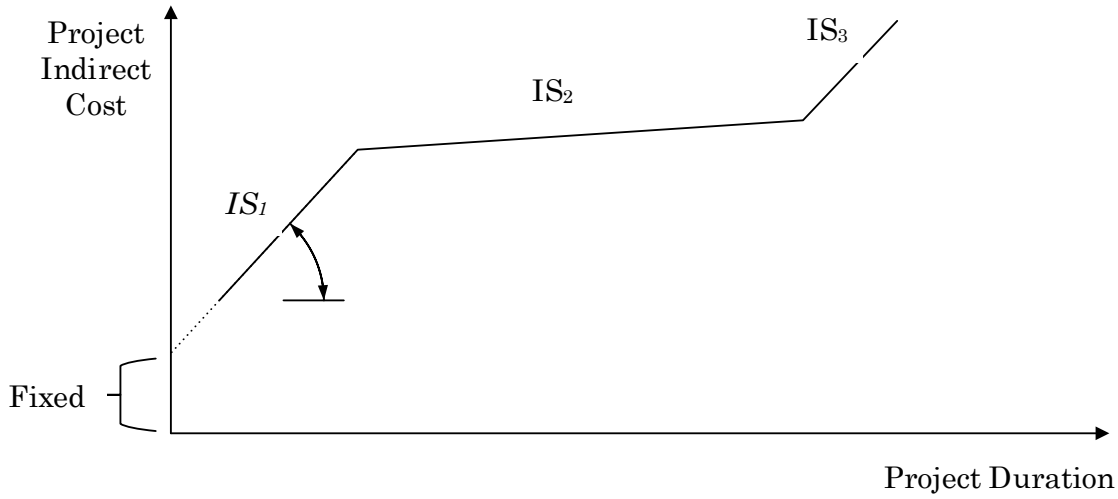



Figure 3.17 Three constant indirect cost slopes

As stated before, the value of objective function in each iteration represents the added direct cost if project duration is reduced by one time-unit. On the other hand, IS represents the reduced indirect cost if project duration is reduced by one time-unit. As a result, the addition of these two values determines the net total cost slope and can guide us in finding local or global optimums. If the conditions for increasing

objective function are met (linear or convex activity time-cost relationships), single and three indirect cost slopes must be examined separately:

- For single constant indirect cost slope, once the value of objective function (combined direct cost slope) surpasses the indirect cost slope in a cycle, total cost can only increase by any further crashing and the lowest total cost has already been recorded, as the increase on direct cost resulting from crashing would outstrip the decrease on indirect cost due to the shortening of project duration.
- For three indirect cost slopes, once the value of objective function (combined direct cost slope) surpasses the corresponding indirect cost slope in a cycle, we must check if the value of indirect cost slope does not increase (it would be equal or lower than current indirect cost slope) as a result of any further project duration crashing. If this is ensured, then total cost can only increase by any further crashing and the lowest total cost has already been recorded. Otherwise, the total cost may be further lowered until the condition is satisfied, that is: if the value of indirect cost slope does not increase (it would be equal or lower than current indirect cost slope) as a result of any further project duration crashing.

## 3.5 Explanatory Notes

Considering a project having 15 activities as an example, with each activity having three options for completion, the total number of possible combinations will be $3^{15}$ (14,348,907). This number is only for a small network. If the number of activities increases to represent a real network, the number of permutations increases exponentially. For a 100-activity network, with 50 of the activities having three crash options, the number would be $3^{50}$ (7.18e+23). Considering the structure of the

problem, the proposed method narrows the search space by breaking down the problem into smaller ones and focusing only on critical path(s) (highlighted activities in Figure 3.18). In addition, for each activity on the critical path(s) that has available crash time, only two modes (crashing or not crashing) is considered. As a result, computational efficiency is improved.

It must be noted that although the search space is limited, IP is still required as we can have multiple critical paths, each having several crashing options, which is probable for a large-sized network. As an example, consider the critical paths highlighted in Figure 3.18. Assuming all activities on the critical paths (9 activities) have crash options, the number of possible combinations to be examined is $2^9$ (512). By extending this analogy to a large-sized network, the necessity of automated IP becomes apparent. In general, manual TCT cannot be performed on a real project.

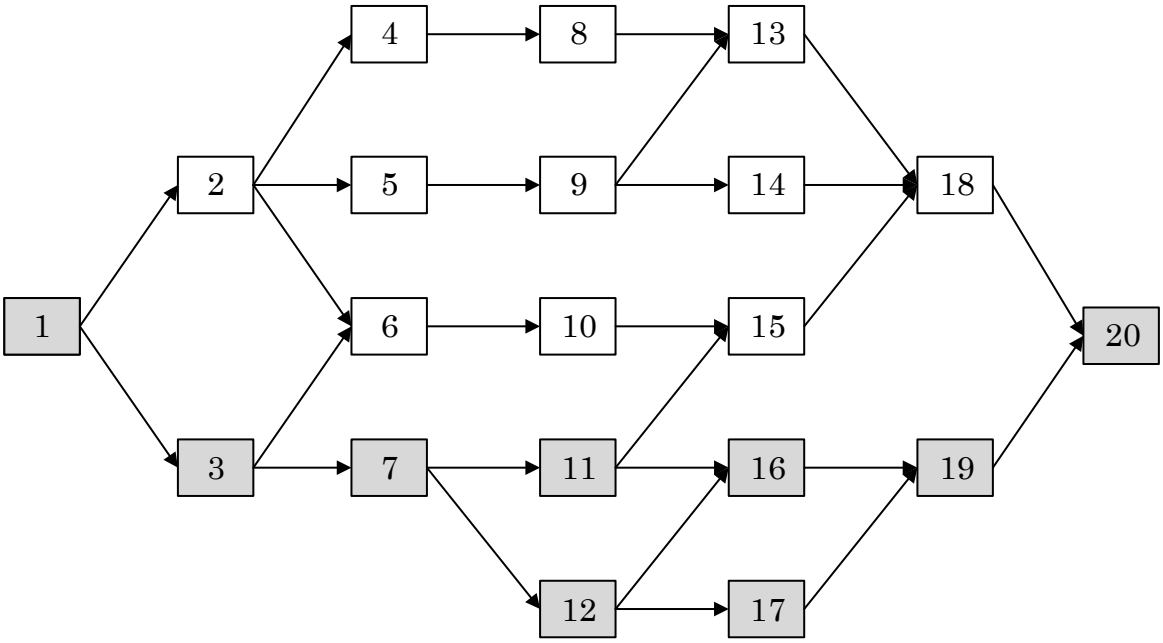



Figure 3.18 Graphical representation of search space

For comparison of formulation, TCT programming models proposed by Liu et al. (1995), Jiang and Zhu (2010), or Ahuja et al. (1994) (refer to Sections 4.2 and 4.3)

can be examined. Common constraints in existing methods include precedence relationships and activity start and finish times. Those constraints are replaced by one constraint for each critical path in the proposed algorithm, which significantly streamlines the modelling effort. Furthermore, only critical paths are required to be formulated, resulting in simpler formulation.

By using PFCPM's path-based method of scheduling, there is no need for the backward pass of classic CPM. In other words, once the paths are identified, updating the schedule in each iteration requires minimal effort.

The proposed method provides a range of feasible alternative solutions that reveals the trend of total cost against project time, and can be used for what-if analysis. Risk of delays can be reduced by choosing a near-optimal solution with a longer duration but reasonably close total cost. The longer duration may result in more total floats for certain activities.

The rules are simple and can be easily understood and applied by practitioners. Early completion bonus and late completion penalty can be easily modelled by extending the procedures in the proposed algorithm and there is no need for them to be modelled in the mathematical programming formulation.

The method can be automated without the need for any specialized software. Solver is a free add-in program for Microsoft Excel to handle mathematical optimization. This program is easy to use and does not require a knowledge of mathematical programming in order to use it. The entire method can be modelled in a spreadsheet. A guide on how to set up the Excel program is provided in Appendix C.

# Chapter 4. Case Studies

In this chapter, three case studies are solved by the proposed method. The first and second cases have linear and curvilinear time-cost relationships, respectively. The results are compared to the solution provided by each reference for cross-validation. The purpose of the third case is to implement the method for a real, relatively large-sized project network.

## 4.1 Case One – Linear Time-Cost Relationship

This case was presented in Section 8.5 of Hegazy (2002). The network consists of 11 activities and activity time-cost relationship is assumed to be continuous linear. Activities' normal and crash data and the project network are presented in Table 4.1 and Figure 4.1. Indirect cost of project is $500 per day of project duration.

Table 4.1 Activity data of case one

| | Normal | | Crash | |
|---|---|---|---|---|
| **Activity** | **Duration (d)** | **Direct Cost ($)** | **Duration (d)** | **Direct Cost ($)** |
| A | 4 | 2,000 | No crashing | |
| B | 6 | 10,000 | 3 | 16,600 |
| C | 2 | 4,000 | No crashing | |
| D | 8 | 18,000 | No crashing | |
| E | 4 | 20,000 | No crashing | |
| F | 10 | 15,000 | No crashing | |
| G | 16 | 12,000 | 12 | 12,800 |
| H | 8 | 16,000 | 4 | 17,000 |
| I | 6 | 10,000 | No crashing | |
| J | 6 | 10,000 | No crashing | |
| K | 10 | 8,000 | 9 | 9,000 |
| **Total Direct Cost ($)** | | 125,000 | | |

As this is a small simple network, paths can be easily identified by visual inspection which are ADEI, BFHI, BGK, and CJK. The results, obtained by the method streamlined for linear time-cost relationship (Section 3.3.1), are presented in Tables 4.2, 4.3, and 4.4. The minimum total cost is $140,300 with a corresponding project duration of 28 days. The shortest possible project duration is 24 days with a corresponding total of $145,150.

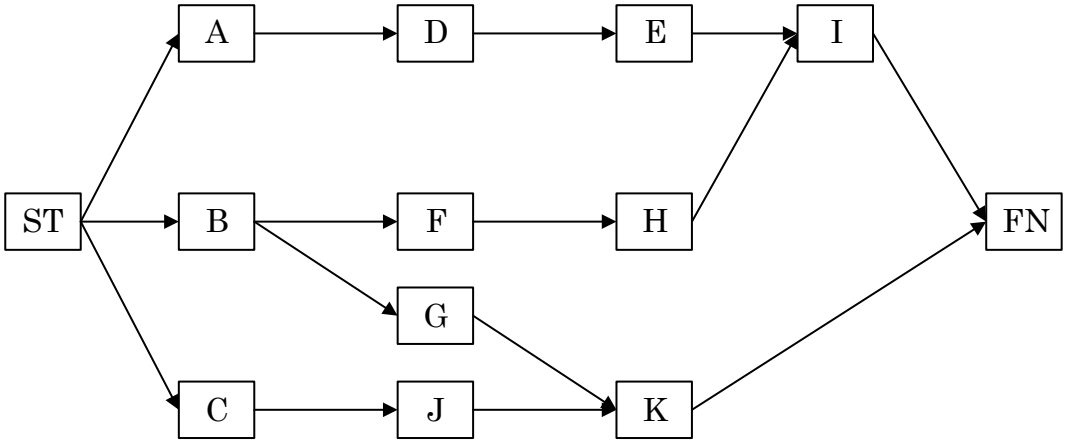



Figure 4.1 AON network of case one

Table 4.2 Cost slopes and available crash times of case one

| Activity | S ($/d) | AC (d) | | | | |
|---|---|---|---|---|---|---|
| | | C1 | C2 | C3 | C4 | C5 |
| A | - | 0 | 0 | 0 | 0 | 0 |
| B | 2,200 | 3 | 3 | 3 | 3 | 0 |
| C | - | 0 | 0 | 0 | 0 | 0 |
| D | - | 0 | 0 | 0 | 0 | 0 |
| E | - | 0 | 0 | 0 | 0 | 0 |
| F | - | 0 | 0 | 0 | 0 | 0 |
| G | 200 | 4 | 2 | 0 | 0 | 0 |
| H | 250 | 4 | 4 | 2 | 1 | 1 |
| I | - | 0 | 0 | 0 | 0 | 0 |
| J | - | 0 | 0 | 0 | 0 | 0 |
| K | 1,000 | 1 | 1 | 1 | 0 | 0 |

Table 4.3 Path lengths of case one

| | PL (d) | | | | |
|---|---|---|---|---|---|
| **Path** | **C1** | **C2** | **C3** | **C4** | **C5** |
| ADEI | 22 | 22 | 22 | 22 | 22 |
| BFHI | 30 | **30** | **28** | **27** | **24** |
| BGK | **32** | **30** | **28** | **27** | **24** |
| CJK | 18 | 18 | 18 | 17 | 17 |

Table 4.4 Results summary of case one

| Cycle | Project Duration (d) | Direct Cost ($) | Indirect Cost ($) | Total Cost ($) | Activity Crashed |
|---|---|---|---|---|---|
| C1 | 32 | 125,000 | 16,000 | 141,000 | |
| C2 | 30 | 125,400 | 15,000 | 140,400 | G-2d |
| C3 | 28 | 126,300 | 14,000 | **140,300** | H-2d & G-2d |
| C4 | 27 | 127,550 | 13,500 | 141,050 | H-1d & K-1d |
| C5 | **24** | 134,150 | 12,000 | 146,150 | B-3d |

For a better demonstration of the proposed method's formulation, a detailed solution for this case is provided. As this is a linear case, cost slopes will not change in each iteration. The following are sample calculations for activity B.

$$AC_B = 6 - 3 = 3 \text{ days}$$

$$S_B = \frac{\$16{,}600 - \$10{,}000}{3\ days} = \$2{,}200/d$$

In the first iteration, only path BGK (path length of 32 days) is critical and from this path, all activities have available crash time. Therefore, the IP formulation is as follows,

Minimize $2200 \times x_B + 200 \times x_G + 1000 \times x_K$

Subject to $x_B = 0{,}1;\ x_G = 0{,}1;\ x_K = 0{,}1$

$$x_B + x_G + x_K \geq 1$$

The result of the IP is as below,

$$x_B = 0;\ x_G = 1;\ x_K = 0$$

Therefore, activity G must be crashed in this cycle. The next-to-critical path is BFHI with a path length of 30 days and available crash time of G is 4 days. Hence, for the crash duration:

$$CD = min\begin{cases} PF_{ntc} = PF_{BFHI} = 32 - 30 = 2\ days \\ min\{AC_G\} = min\{4\} = 4\ days \end{cases} = 2\ days$$

G will be crashed by 2 days. For the second iteration, both paths of BGK and BFHI are critical (path length of 30 days) from which activities B, G, H, and K have available crash time. The IP formulation for second iteration is as follows,

Minimize $2200 \times x_B + 200 \times x_G + 250 \times x_H + 1000 \times x_K$

Subject to $x_B = 0{,}1;\ x_G = 0{,}1; x_H = 0{,}1;\ x_K = 0{,}1$

$$x_B + x_G + x_K \geq 1\ \ (path\ BGK)$$

$$x_B + x_H \geq 1\ (path\ BFHI)$$

The result of the IP is as below,

$$x_B = 0;\ x_G = 1;\ x_H = 1;\ x_K = 0$$

Therefore, activities G and H must be crashed in this cycle. The next-to-critical path is ADEI with a path length of 22 days and available crash times of G and H are 4 and 2 days, respectively. Hence, for the crash duration:

$$CD = min\begin{cases} PF_{ntc} = PF_{ADEI} = 30 - 22 = 8\ days \\ min\{4{,}2\} = 2\ days \end{cases} = 2\ days$$

G and H will each be crashed by 2 days. For the next iterations, a more summarized solution is provided. For the third iteration,

Minimize $2200 \times x_B + 250 \times x_H + 1000 \times x_K$

Subject to $$x_B = 0,1; x_H = 0,1;\ x_K = 0,1$$

$$x_B + x_K \geq 1\ \ (path\ BGK)$$

$$x_B + x_H \geq 1\ (path\ BFHI)$$

$$\Rightarrow x_B = 0;\ x_H = 1;\ x_K = 1$$

$$CD = min\begin{cases} PF_{ntc} = PF_{ADEI} = 28 - 22 = 6\ days \\ min\ \{2,1\} = 1\ day \end{cases} = 1\ day$$

H and K will each be crashed by 1 day. For the fourth iteration,

Minimize $$2200 \times x_B + 250 \times x_H$$

Subject to $$x_B = 0,1; x_H = 0,1;$$

$$x_B \geq 1\ \ (path\ BGK)$$

$$x_B + x_H \geq 1\ (path\ BFHI)$$

$$\Rightarrow x_B = 1;\ x_H = 0$$

$$CD = min\begin{cases} PF_{ntc} = PF_{ADEI} = 27 - 22 = 5\ days \\ min\ \{3\} = 3\ days \end{cases} = 3\ day$$

B will be crashed by 3 days. For the fifth iteration, none of activities on path BGK have available crash time. Therefore, path BGK cannot be crashed anymore and TCT analysis will be terminated. Hegazy (2002) has solved this problem by using the cost slope method with manual activity selection and classic CPM (with backward pass) and has obtained the same results as our analysis. In this case, manual activity selection is possible as the project network is small. Figure 4.2 represents the alternative feasible solutions and the trend of direct and total cost of the project. Table 4.5 is the comparison of objective function (combined direct cost slope) and indirect cost slope in each iteration.

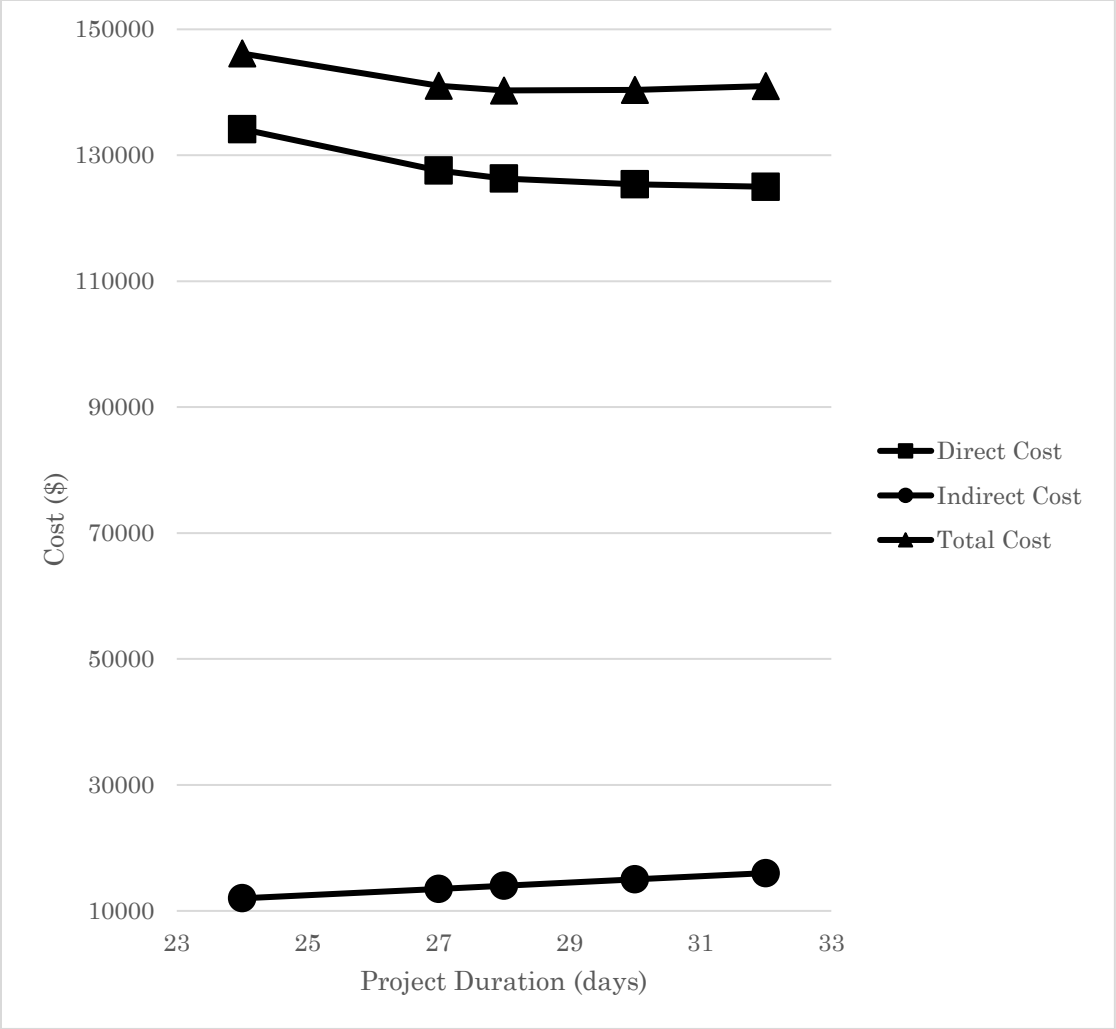



Figure 4.2 Cost trends of case one

Table 4.5 Objective function comparison with indirect cost slope

| Cycle | Value of Objective Function ($/day) | Indirect Cost Slope (4/day) |
|---|---|---|
| C2 | 200 | 500 |
| C3 | 450 | 500 |
| C4 | 1,250 | 500 |
| C5 | 2,200 | 500 |

## 4.2 Case Two – Curvilinear Time-Cost Relationship

This case was presented in Chapter 10 of Ahuja et al. (1994). The network consists of 5 activities and activity time-cost relationship is assumed to be continuous curvilinear. Activity time-cost relationships' one time-unit approximation is presented in Table 4.6 with cost slopes calculated. The project's AON network is presented in Figure 4.3. Indirect cost of project is $160 per day of project duration.

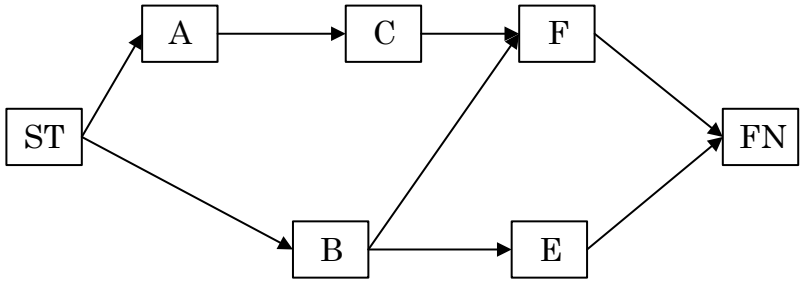



Figure 4.3 AON network of case two

Table 4.6 Crash data of case two

| | Normal | | First Crash | | | Second Crash | | |
|---|---|---|---|---|---|---|---|---|
| **Activity** | **Duration (d)** | **Direct Cost ($)** | **Duration (d)** | **Direct Cost ($)** | **S ($/d)** | **Duration (d)** | **Direct Cost ($)** | **S ($/d)** |
| A | 5 | 4,000 | 4 | 4,100 | 100 | 3 | 4,220 | 120 |
| B | 6 | 4,200 | 5 | 4,330 | 130 | 4 | 4,580 | 250 |
| C | 4 | 3,200 | 3 | 3,280 | 80 | 2 | 3,380 | 100 |
| E | 3 | 1,000 | 2 | 1,100 | 100 | | | |
| F | 2 | 2,600 | 1 | 2,740 | 140 | | | |
| **Total Direct Cost ($)** | | 15,000 | | | | | | |

Summary of the TCT analysis results is provided in Table 4.7 and Table 4.8. The lowest project cost is $16,620 with a corresponding duration of 9 days (only activity C requires to be crashed by 2 days). The shortest project duration is 6 days with a

corresponding cost of $16,980. Figure 4.4 represents the alternative feasible solutions and the trend of direct and total cost of the project.

Table 4.7. Path lengths of case two

| | PL (d) | | | | | |
|---|---|---|---|---|---|---|
| **Path** | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** |
| ACF | **11** | **10** | **9** | **8** | **7** | **6** |
| BF | 8 | 8 | 8 | **8** | **7** | 5 |
| BE | 9 | 9 | **9** | **8** | **7** | **6** |

Table 4.8 Results summary of case two

| Cycle | Project Duration (d) | Direct Cost ($) | Indirect Cost ($) | Total Cost ($) | Activity Crashed |
|---|---|---|---|---|---|
| C1 | 11 | 15,000 | 1,760 | 16,760 | |
| C2 | 10 | 15,080 | 1,600 | 16,680 | C-1d |
| C3 | 9 | 15,180 | 1,440 | **16,620** | C-1d |
| C4 | 8 | 15,380 | 1,280 | 16,660 | A-1d& E-1d |
| C5 | 7 | 15,630 | 1,120 | 16,750 | A-1d& B-1d |
| C6 | **6** | 16,020 | 960 | 16,980 | B-1d& F-1d |

Ahuja et al. (1994) solved this problem by using integer programming which produced one exact optimum solution that denoted the lowest total cost. Obtained optimum point was of 9 days and $16,620 which is the same as our solution. Linear programming formulation of Ahuja et al. (1994) consisted of 9 variables and 19 constraints and was based on an activity-on-arrow (AOA) network. Each variable represented durations of linear segments of activity time-cost relationships. Six of the constraints represented precedence relationships and the remaining thirteen constraints represented upper and lower limits of durations of each linear segment. Note that the total of 19 constraints did not include the integer non-negative

constraints imposed on variables. Proposed algorithm's formulation requires only 3 constraints, each representing a path. Furthermore, this cumbersome formulation for a 5 activity network only provides one solution (optimum) as opposed to five feasible alternatives obtained by the proposed algorithm.

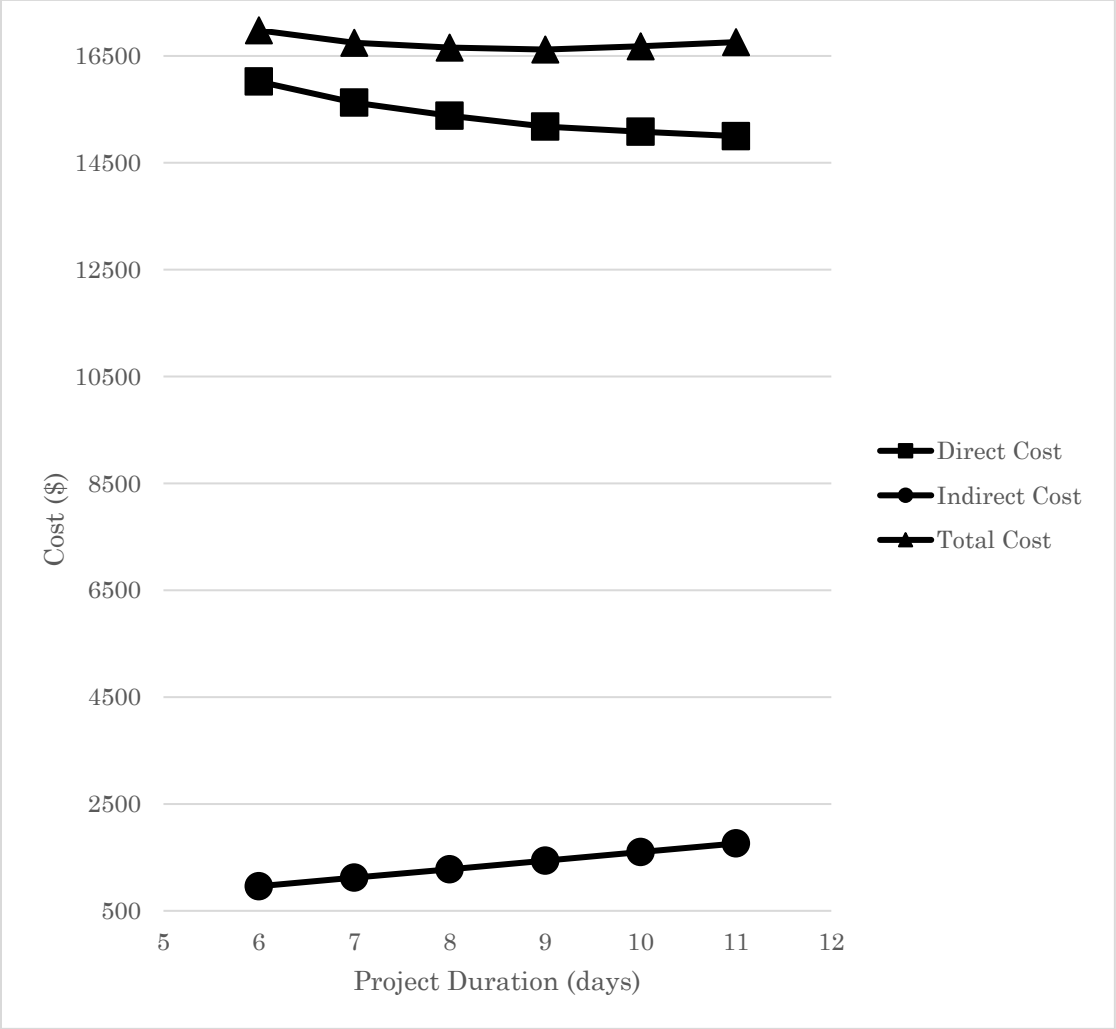



Figure 4.4 Cost trends of case two

## 4.3 Case Three – Practical Project

Proposed algorithm must be tested on a large network to ensure its practicality. However, there are no such examples found in connection with TCT analyses (regardless of heuristics or optimization method being applied) in literature. Therefore, a network based on a real two-story residential project obtained from industry has been selected to test the proposed algorithm on a practical, more complex scenario. This case can be used as a complete test bed for TCT methods in the future. The 81 activity AON network and project information has been summarized without eliminating any TCT related aspects for the purpose of simple presentation. Changes made include:

- Removal of activities without available crash time and transferring their duration to other activities (either crashable or non-crashable) without changing the logic of the network. Note that the direct cost of removed activities is still accounted for in total direct cost.
- The criticality theorem presented by Ahuja et al. (1994) has been used to remove redundant paths. Based on the criticality theorem, if a path has a normal length less than or equal to the crashed project duration (all activities executed under crash scenario), this path can never become critical and can be excluded from TCT optimization.

Project information is provided in Table 4.9. Most activities have available crash times with either linear or curvilinear activity time-cost relationships.

Table 4.9 Project information for case three

| Activity ID | Predecessor | Normal Duration (d) | AC (d) | Type | Cost Slope ($/d) | | |
|---|---|---|---|---|---|---|---|
| A | ST | 4 | 0 | No crash | - | - | - |
| B | A | 3 | 0 | No crash | - | - | - |
| C | A | 6 | 0 | No crash | - | - | - |
| D | A | 2 | 0 | No crash | - | - | - |
| E | A | 5 | 2 | Linear | 3,000 | - | - |
| F | E | 2 | 1 | Linear | 1,000 | - | - |
| G | F | 12 | 4 | Linear | 700 | - | - |
| H | G | 5 | 1 | Linear | 2,800 | - | - |
| I | H | 15 | 3 | Linear | 5,700 | - | - |
| J | I | 33 | 4 | Linear | 2,300 | - | - |
| K | H | 5 | 1 | Linear | 2,000 | - | - |
| L | K | 14 | 1 | Linear | 400 | - | - |
| M | L | 4 | 1 | Linear | 1,700 | - | - |
| N | M, P | 12 | 3 | Curvilinear | 5,000 | 5,100 | 5,200 |
| O | N | 18 | 3 | Linear | 850 | - | - |
| P | L | 6 | 1 | Linear | 1,050 | - | - |
| Q | P | 12 | 1 | Linear | 2,300 | - | - |
| R | N, Q | 10 | 2 | Curvilinear | 3,700 | 5,000 | - |
| S | R | 6 | 1 | Linear | 820 | - | - |
| T | Q | 18 | 2 | Linear | 3,500 | - | - |

Total direct cost of the project is estimated to be $590,000 under normal conditions and normal project duration is 83 days. Indirect cost is modelled in two separate scenarios. The first scenario assumes a fixed indirect cost of $20,000 and a variable indirect cost of $2000 per day of project duration. For the second scenario, the variable indirect cost has been modified to include three indirect cost slopes (Table 4.10).

Table 4.10 Indirect cost information for second scenario (three indirect slopes)

| | Indirect Cost Slope ($/d) | Corresponding Project Duration (T) |
|---|---|---|
| Variable | 2,050 | T ≤71 |
| | 1,500 | 71< T ≤77 |
| | 1,890 | 77< T |
| Fixed | 20,000 | |

Identified non-redundant paths are presented in Table 4.11. Path lengths and available crash times of each iteration are presented in Table 4.12 and Table 4.13, respectively. Summary of results for scenario one and scenario two are provided in Tables 4.14 and 4.15, respectively. The shortest project duration is determined to be 70 days for both scenarios. The lowest project cost for first scenario is $767,250 with a corresponding project duration of either 75 or 76 days, as added direct cost by crashing activity K is equal to indirect cost slope in that iteration. The lowest project cost for second scenario is $768,300 with a corresponding project duration of 76 days. Figure 4.5 and Figure 4.6 represent the alternative feasible solutions and the trend of direct and total cost of the project for first (single slope) and second (three slopes) indirect cost scenarios, respectively.

Table 4.11 Identified paths of case three

| Path ID | Activities | Normal Path Length (d) |
|---|---|---|
| 1 | AB | 7 |
| 2 | AC | 10 |
| 3 | AD | 6 |
| 4 | AEFGHIJ | 76 |
| 5 | AEFGHKLMNO | 81 |
| 6 | AEFGHKLPNO | **83** |
| 7 | AEFGHKLMNRS | 79 |
| 8 | AEFGHKLPNRS | 81 |
| 9 | AEFGHKLPQRS | 81 |
| 10 | AEFGHKLPQT | **83** |

Table 4.12 Path length results

| Path ID | PL (d) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** | C7 | C8 | C9 | C10 | C11 | C12 |
| 1 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 2 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 3 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 4 | 76 | 76 | 74 | 72 | 71 | 71 | 71 | 70 | 69 | 69 | 69 | 69 |
| 5 | 81 | 80 | 78 | 76 | 75 | 75 | 74 | 73 | 72 | 71 | 70 | 69 |
| 6 | **83** | **82** | **80** | **78** | **77** | **76** | **75** | **74** | **73** | **72** | **71** | **70** |
| 7 | 79 | 78 | 76 | 74 | 73 | 73 | 72 | 71 | 70 | 70 | 70 | 69 |
| 8 | 81 | 80 | 78 | 76 | 75 | 74 | 73 | 72 | 71 | 71 | **71** | **70** |
| 9 | 81 | 80 | 78 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 70 | 69 |
| 10 | **83** | **82** | **80** | **78** | **77** | **76** | **75** | **74** | **73** | **72** | **71** | **70** |

Table 4.13 Available crash time of case three

| Activity | AC (d) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** | **C7** | **C8** | **C9** | **C10** | **C11** | **C12** |
| A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| D | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| E | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| F | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 4 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| H | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| I | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| J | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| K | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| L | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| N | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| O | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 1 | 0 |
| P | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Q | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| R | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| S | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| T | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 0 |

Table 4.14 Results of case three: single indirect cost slope

| Cycle | Project Duration (d) | Direct Cost ($) | Indirect Cost ($) | Total Cost ($) | Activity Crashed |
|---|---|---|---|---|---|
| C1 | 83 | $590,000 | $186,000 | $776,000 | |
| C2 | 82 | $590,400 | $184,000 | $774,400 | L-1d |
| C3 | 80 | $591,800 | $180,000 | $771,800 | G-2d |
| C4 | 78 | $593,200 | $176,000 | $769,200 | G-2d |
| C5 | 77 | $594,200 | $174,000 | $768,200 | F-1d |
| C6 | 76 | $595,250 | $172,000 | **$767,250** | P-1d |
| C7 | 75 | $597,250 | $170,000 | **$767,250** | K-1d |
| C8 | 74 | $600,050 | $168,000 | $768,050 | H-1d |
| C9 | 73 | $603,050 | $166,000 | $769,050 | E-1d |
| C10 | 72 | $606,200 | $164,000 | $770,200 | O-1d, Q-1d |
| C11 | 71 | $610,550 | $162,000 | $772,550 | O-1d, T-1d |
| C12 | 70 | $615,720 | $160,000 | $775,720 | O-1d, S-1d, T-1d |

Table 4.15 Results of case three: three indirect cost slopes

| Cycle | Project Duration (d) | Direct Cost ($) | Indirect Cost ($) | Total Cost ($) | Activity Crashed |
|---|---|---|---|---|---|
| C1 | 83 | $590,000 | $185,890 | $775,890 | |
| C2 | 82 | $590,400 | $184,000 | $774,400 | L-1d |
| C3 | 80 | $591,800 | $180,220 | $772,020 | G-2d |
| C4 | 78 | $593,200 | $176,440 | $769,640 | G-2d |
| C5 | 77 | $594,200 | $174,550 | $768,750 | F-1d |
| C6 | 76 | $595,250 | $173,050 | **$768,300** | P-1d |
| C7 | 75 | $597,250 | $171,550 | $768,800 | K-1d |
| C8 | 74 | $600,050 | $170,050 | $770,100 | H-1d |
| C9 | 73 | $603,050 | $168,550 | $771,600 | E-1d |
| C10 | 72 | $606,200 | $167,050 | $773,250 | O-1d, Q-1d |
| C11 | 71 | $610,550 | $165,550 | $776,100 | O-1d, T-1d |
| C12 | 70 | $615,720 | $163,500 | $779,220 | O-1d, S-1d, T-1d |

Note that as the conditions for increasing objective function (linear or convex activity time-cost relationships) are met, the comparison of value of objective function and indirect cost slope was performed for both scenarios. However, as we

are interested in not just the minimum total cost, but the minimum total duration as well, the analysis was continued.

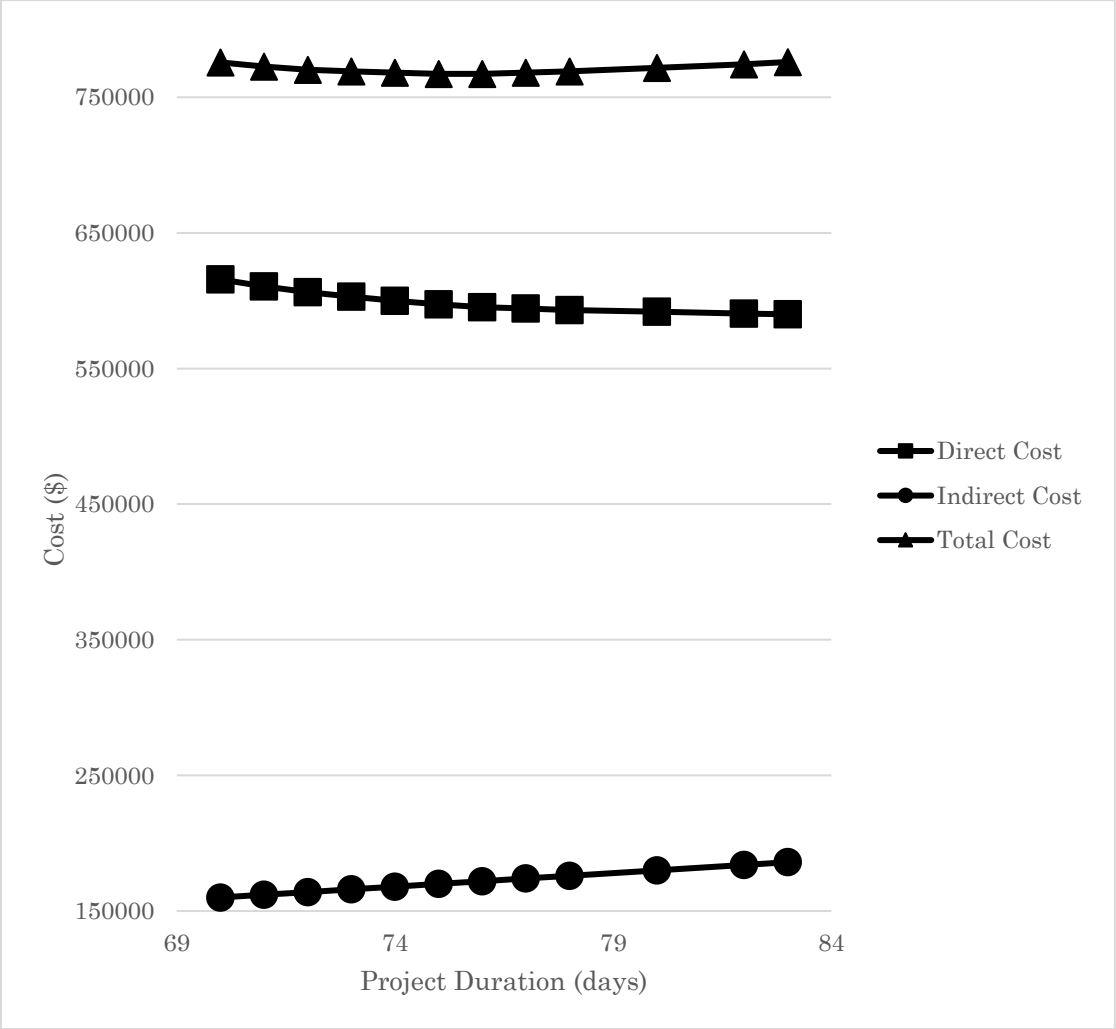



Figure 4.5 Cost trend of case three for first scenario (single slope)

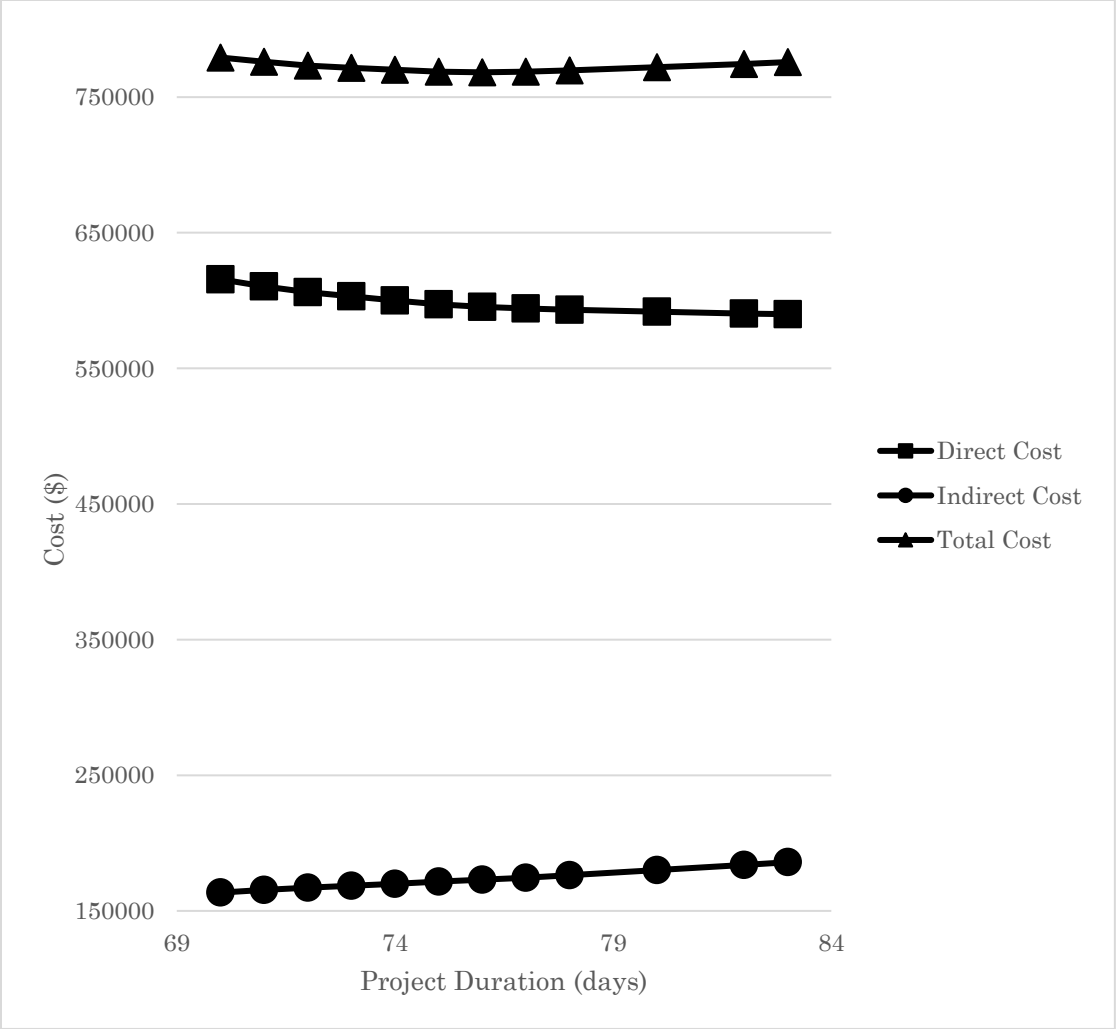



Figure 4.6 Cost trend of case three for second scenario (three slopes)

As can be seen, paths number 6 and 10 are critical in every cycle and path 8 becomes critical in the last two cycles. Therefore, the number of required constraints is 2 for the first ten cycles and 3 for the last two cycles. Note that constraints do not change from cycle to cycle as the critical paths remain the same and only one new constraint (for path 8) is added for the last two cycles.

To use the method proposed by Ahuja et al. (1994), the network is converted into AOA (Figure 4.7). For each activity $a_i$, let $t_i$ be the variable representing the duration of the activity. For each event (node) on the AOA, event time is designated

as $E_I$. The formulation based on Ahuja et al. (1994) for the first scenario (single indirect cost slope) is as follows:

Maximize $3000 \times t_E + 1000 \times t_F + 700 \times t_G + 2800 \times t_H + 5700 \times t_I + 2300 \times t_J + 2000 \times t_K + 400 \times t_L + 1700 \times t_M + 5000 \times t_{N1} + 5100 \times t_{N2} + 5200 \times t_{N3} + 850 \times t_O + 1050 \times t_P + 2300 \times t_Q + 3700 \times t_{R1} + 5000 \times t_{R2} + 820 \times t_S + 3500 \times t_T - 2000 \times E_{16}$

Subject to

$$t_A = 4$$
$$t_B = 3$$
$$t_C = 6$$
$$t_D = 2$$
$$t_E \leq 5$$
$$t_E \geq 3$$
$$t_F \leq 2$$
$$t_F \geq 1$$
$$t_G \leq 12$$
$$t_G \geq 8$$
$$t_H \leq 5$$
$$t_H \geq 4$$
$$t_I \leq 15$$
$$t_I \geq 12$$
$$t_J \leq 33$$
$$t_J \geq 29$$
$$t_K \leq 5$$
$$t_K \geq 1$$

$$t_L \leq 14$$

$$t_L \geq 13$$

$$t_M \leq 4$$

$$t_M \geq 3$$

$$t_{N1} \leq 12 - 11$$

$$t_{N2} \leq 11 - 10$$

$$t_{N3} \leq 10$$

$$t_{N3} \geq 9$$

$$t_O \leq 18$$

$$t_O \geq 16$$

$$t_P \leq 6$$

$$t_P \geq 5$$

$$t_Q \leq 12$$

$$t_Q \geq 11$$

$$t_{R1} \leq 10 - 9$$

$$t_{R2} \leq 9$$

$$t_{R2} \geq 8$$

$$t_S \leq 6$$

$$t_S \geq 5$$

$$t_T \leq 18$$

$$t_T \geq 16$$

$$E_1 + t_A - E_2 \leq 0$$

$$E_2 + t_B - E_{16} \leq 0$$

$$E_2 + t_C - E_{16} \leq 0$$

$$E_2 + t_D - E_{16} \leq 0$$

$$E_2 + t_E - E_3 \leq 0$$

$$E_3 + t_F - E_4 \leq 0$$

$$E_4 + t_G - E_5 \leq 0$$

$$E_5 + t_H - E_6 \leq 0$$

$$E_6 + t_I - E_7 \leq 0$$

$$E_7 + t_J - E_{16} \leq 0$$

$$E_6 + t_K - E_8 \leq 0$$

$$E_8 + t_L - E_9 \leq 0$$

$$E_9 + t_M - E_{10} \leq 0$$

$$E_{10} + t_{N1} + t_{N2} + t_{N3} - E_{12} \leq 0$$

$$E_{12} + t_O - E_{16} \leq 0$$

$$E_9 + t_P - E_{11} \leq 0$$

$$E_{11} + t_Q - E_{13} \leq 0$$

$$E_{14} + t_{R1} + t_{R2} - E_{15} \leq 0$$

$$E_{15} + t_S - E_{16} \leq 0$$

$$E_{13} + t_T - E_{16} \leq 0$$

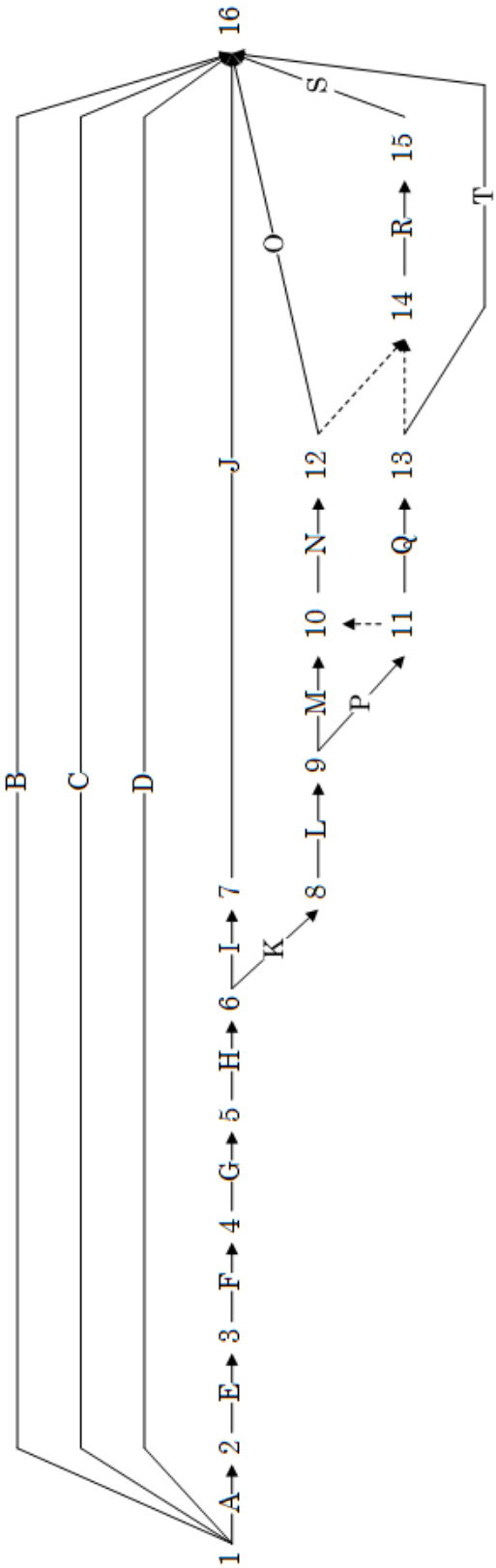



Figure 4.7 AOA network of case 3

Constraints of the formulation include:

- Activity duration upper and lower bounds,
- Precedence relationships (for the constraints that include event times).

A total of 59 constraints is required. This method can handle the three indirect cost slopes by adding a few constraints and is an established pure mathematical programming method to arrive at the optimum solution. As can be seen, the formulation effort required by the proposed method (3 constraints) is considerably less than the method of Ahuja et al. (1994). Quantitative comparison through controlled experiments is out of the scope of this thesis and will be left for immediate future follow-up work. In addition to formulation, solving a complex large-scale TCT problem with pure mathematical programming methods can be costly in terms of training, software, and hardware. Not to mention, the optimization engine might not converge to optimum at all due to inherent limitations of solution algorithms (Yang, 2008). Furthermore, TCT needs to be applied on dynamic construction projects, where resources, time, and expertise available for TCT analysis is limited; thus, frequent changes requires a light-weight solution for TCT like the one being proposed in order to easily update the analysis results and keep current of the changing situations in reality.

# Chapter 5. Conclusion

TCT is an important tool that can provide a major advantage in the competitive construction market. It obtains the most economical project duration and shortest project duration by reducing the duration of a selection of activities. However, in practice, its use is limited as current methods are complex and require tremendous effort. To address this problem, this study has proposed a new, more practical approach for TCT. The proposed iterative algorithm utilizes a path-based approach to analyze the project network; while IP is employed to handle the complexity of real-life cases. The method can be automated and is computationally efficient, easy to formulate, and ready to scale up to projects of practical size and complexity. The problem is broken down into smaller sub-problems for optimization in each iteration, focusing only on the critical paths instead of the entire project. Furthermore, the decision of duration of crashing is not part of the IP and the number of required constraints is comparatively less than other methods (refer to Sections 3.5, 4.2, and 4.3). Therefore, although no systematic way of comparison has been provided yet, it can be deduced that the execution time and modelling effort of the proposed method is less than most existing methods. The method is intended to be simple and can handle large-scale networks (e.g. 1000 activities). It also provides a range of feasible alternatives that reveals the cost trends, resulting in optimal or near optimal solutions. To summarize, the proposed method adds to the body of knowledge by extending PFCPM into an optimization scheme for TCT, while providing a more streamlined, less complex, and more practical approach to tackle the classic TCT problem in critical path scheduling.

Despite the advantages of the proposed methodology, certain limitations of this research should be noted and explained:

- Calendar days are not considered in the algorithm. For example, concrete curing can be carried out during weekends. The general solution is to add the calendar constraints onto the final schedule after TCT analysis, prior to implementation (similar to Microsoft Project and Primavera P6). In general, calendar days are deemed irrelevant in scheduling optimization.
- Uncertainties in costs and durations are not modelled. Activity cost and duration is assumed to be deterministic.
- The method assumes that the inputs, such as work breakdown structure, AON network, and activity time-cost relationships are sufficiently defined prior to TCT analysis. Any inaccuracy of inputs can undermine the reliability of TCT results.

For future work, the current research can be extended to several fields:

- The algorithm can adapt a stochastic approach to quantify the risk associated with TCT results. This approach requires sufficient time and cost data. In addition, existing data may not be detailed enough to be useful in TCT. However, artificial intelligence techniques may be used to extract usable information. As the proposed method is relatively light-weight and computationally efficient, it can be extended into a simulation scheme to enable the use of random sampling, Monte Carlo simulation and statistical analyses for addressing uncertainty or risks in connection with input data. To a certain extent, the proposed TCT method can be taken as a more sophisticated version of CPM for project time and cost planning; it would enable the development of a more sophisticated version of PERT simulation for investigating risks on project time and cost tradeoff.
- Resource use can be added to the proposed algorithm to combine TCT with resource allocation and perform time-cost-resource tradeoff simultaneously.

- Quantitative comparison with other methods through controlled experiments on case studies is part of the immediate future follow-up work.
- Planning and input preparation for TCT has been mostly untouched in literature. Several subjects such as activity time-cost relationship, implications of schedule compression on safety and its cost, TCT-related data collection and retrieval system, or project-type-specific TCT have great potential to improve the current method.

# References


Ahuja, H. (1984). Project management techniques in planning and controlling construction projects. Wiley, New York.

Ahuja, H. N., Dozzi, S. P., & AbouRizk, S. M. (1994). Project management: techniques in planning and controlling construction projects. John Wiley & Sons.

Al Haj, R. A., & El-Sayegh, S. M. (2015). Time–cost optimization model considering float-consumption impact. Journal of Construction Engineering and Management, 141(5), 04015001.

Ammar, M. A. (2010). Optimization of project time-cost trade-off problem with discounted cash flows. Journal of Construction Engineering and Management, 137(1), 65-71.

Ammar, M. A. (2018). Efficient modeling of time-cost trade-off problem by eliminating redundant paths. International Journal of Construction Management, 1-10.

Baker, B. M. (1997). Cost/time trade-off analysis for the critical path method: A derivation of the network flow approach. Journal of the Operational Research Society, 48(12), 1241-1244.

Bettemir, Ö. H., & Birgönül, M. T. (2017). Network analysis algorithm for the solution of discrete time-cost trade-off problem. KSCE Journal of Civil Engineering, 21(4), 1047-1058.

Danielson, G. (1968). On finding the simple paths and circuits in a graph. IEEE transactions on circuit theory, 15(3), 294-295.

Dantzig, G. B., Ford Jr, L. R., & Fulkerson, D. R. (1956). A primal--dual algorithm (No. P-778). Rand Corp Santa Monica CA.

De, P., Dunne, E. J., Ghosh, J. B., & Wells, C. E. (1995). The discrete time-cost tradeoff problem revisited. European journal of operational research, 81(2), 225-238.

De, P., Dunne, E. J., Ghosh, J. B., & Wells, C. E. (1997). Complexity of the discrete time-cost tradeoff problem for project networks. Operations research, 45(2), 302-306.

Deckro, R. F., Hebert, J. E., Verdini, W. A., Grimsrud, P. H., & Venkateshwar, S. (1995). Nonlinear time/cost tradeoff models in project management. Computers & Industrial Engineering, 28(2), 219-229.

Elbeltagi, E., Hegazy, T., & Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. Advanced engineering informatics, 19(1), 43-53.

El-Kholy, A. M. (2013). New aspects in time-cost tradeoff analysis. Journal of Management in Engineering, 31(4), 04014051.

Falk, J. E., & Horowitz, J. L. (1972). Critical path problems with concave cost-time curves. Management Science, 19(4-part-1), 446-455.

Feng, C. W., Liu, L., & Burns, S. A. (2000). Stochastic construction time-cost trade-off analysis. Journal of Computing in Civil Engineering, 14(2), 117-126.

Fratta, L., & Montanari, U. (1975). A vertex elimination algorithm for enumerating all simple paths in a graph. Networks, 5(2), 151-177.

Fulkerson, D. R. (1961). A network flow computation for project cost curves. Management science, 7(2), 167-178.

Goyal, S. K. (1975). Note—A Note on "A Simple CPM Time-Cost Tradeoff Algorithm". Management Science, 21(6), 718-722.

Goyal, S. K. (1996). A simple time-cost tradeoff algorithm. Production Planning & Control, 7(1), 104-106.

Hegazy, T. (2002). "Resource management: Part 2 - Time-Cost Tradeoff." Chapter 8 in Computer-Based Construction Project Management, T. Hegazy, Prentice Hall, Upper Saddle River, New Jersey, 211-236.

Hegazy, T., & Ayed, A. (1999). Simplified spreadsheet solutions: Models for critical path method and Time-Cost-Tradeoff analysis. Cost Engineering, 41(7), 26.

Hochbaum, D. S. (2016). A polynomial time repeated cuts algorithm for the time cost tradeoff problem: The linear and convex crashing cost deadline problem. Computers & Industrial Engineering, 95, 64-71.

Jiang, A., & Zhu, Y. (2010). A multi-stage approach to time-cost trade-off analysis using mathematical programming. International Journal of Construction Management, 10(3), 13-27.

Kelley Jr, J. E. (1961). Critical-path planning and scheduling: Mathematical basis. Operations research, 9(3), 296-320.

Kelley Jr, J. E., & Walker, M. R. (1959, December). Critical-path planning and scheduling. In Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference (pp. 160-173). ACM.

Kim, J., Kang, C., & Hwang, I. (2012). A practical approach to project scheduling: considering the potential quality loss cost in the time–cost tradeoff problem. International Journal of Project Management, 30(2), 264-272.

Kroft, D. (1967). All paths through a maze. Proceedings of the IEEE, 55(1), 88-90.

Liu, J., & Rahbar, F. (2004). Project time-cost trade-off optimization by maximal flow theory. Journal of construction engineering and management, 130(4), 607-609.

Liu, L., Burns, S. A., & Feng, C. W. (1995). Construction time-cost trade-off analysis using LP/IP hybrid method. Journal of construction engineering and management, 121(4), 446-454.

Lu, M., & Lam, H. C. (2009). Transform schemes applied on non-finish-to-start logical relationships in project network diagrams. Journal of construction engineering and management, 135(9), 863-873.

Lu, M., & Li, H. (2003). Resource-activity critical-path method for construction planning. Journal of construction engineering and management, 129(4), 412-420.

Lu, M., Liu, J., & Ji, W. (2017). Formalizing a path-float-based approach to determine and interpret total float in project scheduling analysis. International Journal of Construction Management, 17(4), 251-263.

Moussourakis, J., & Haksever, C. (2004). Flexible model for time/cost tradeoff problem. Journal of Construction Engineering and Management, 130(3), 307-314.

Moussourakis, J., & Haksever, C. (2009). Project compression with nonlinear cost functions. Journal of Construction Engineering and Management, 136(2), 251-259.

Olawale, Y. A., & Sun, M. (2010). Cost and time control of construction projects: inhibiting factors and mitigating measures in practice. Construction management and economics, 28(5), 509-526.

Pathak, B. K., & Srivastava, S. (2014). Integrated ANN-HMH approach for nonlinear time-cost tradeoff problem. International Journal of Computational Intelligence Systems, 7(3), 456-471.

Peurifoy, R. L., & Oberlender, G. D. (2002). Estimating construction costs. McGraw-Hill.

Phillips Jr, S., & Dessouky, M. I. (1977). Solving the project time/cost tradeoff problem using the minimal cut concept. Management Science, 24(4), 393-400.

Prager, W. (1963). A structural method of computing project cost polygons. Management Science, 9(3), 394-404.

Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-Aided Design, 43(3), 303-315.

Robinson, D. R. (1975). A dynamic programming solution to cost-time tradeoff for CPM. Management Science, 22(2), 158-166.

Rubin, F. (1978). Enumerating all simple paths in a graph. IEEE Transactions on Circuits and Systems, 25(8), 641-642.

Siemens, N. (1971). A simple CPM time-cost tradeoff algorithm. Management science, 17(6), B-354.

Siemens, N., & Gooding, C. (1975). Reducing project duration at minimum cost: A time-cost tradeoff algorithm. Omega, 3(5), 569-581.

Su, Z. X., Qi, J. X., & Wei, H. Y. (2015). A float-path theory and its application to the time-cost tradeoff problem. Journal of Applied Mathematics, 2015.

Toğan, V., & Eirgash, M. A. (2019). Time-Cost Trade-off Optimization of Construction Projects using Teaching Learning Based Optimization. KSCE Journal of Civil Engineering, 23(1), 10-20.

Xiong, Y., & Kuang, Y. (2008). Applying an ant colony optimization algorithm-based multiobjective approach for time–cost trade-off. Journal of Construction Engineering and Management, 134(2), 153-156.

Yang, I. T. (2007). Performing complex project crashing analysis with aid of particle swarm optimization algorithm. International Journal of Project Management, 25(6), 637-646.

Yang, X. (2008). Introduction to mathematical optimization. From Linear Programming to Metaheuristics.

Zheng, D. X., & Ng, S. T. (2005). Stochastic time–cost optimization model incorporating fuzzy sets theory and nonreplaceable front. Journal of Construction Engineering and Management, 131(2), 176-186.

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

Zou, X., Fang, S. C., Huang, Y. S., & Zhang, L. H. (2016). Mixed-integer linear programming approach for scheduling repetitive projects with time-cost trade-off consideration. Journal of computing in civil engineering, 31(3), 06016003.

# Appendix A Resource Levelling

Lu and Li (2003) presented a method called resource-activity critical path method (RACPM), which implemented resource limitations onto AON by adding resource links. To elaborate, consider the AON presented in Figure A.1. Activity durations and resource requirements based on original CPM results are presented in Table A.1. Assuming only five laborers are available, labor requirement on day 3 exceeds the limit.

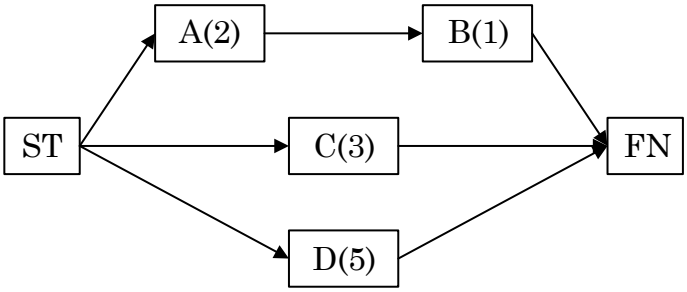



Figure A.1 Original AON for RACPM

Table A.1 Resource requirement for RACPM

| Activity | Duration | Labor Requirement | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 |
|---|---|---|---|---|---|---|---|
| A | 2 | 1 | 1 | 1 | | | |
| B | 1 | 2 | | | 2 | | |
| C | 3 | 3 | 3 | 3 | 3 | | |
| D | 5 | 1 | 1 | 1 | 1 | 1 | 1 |
| Labor Requirement | | | 5 | 5 | **6** | 1 | 1 |

Consider the new resource plan presented in Table A.2 which adheres to the limit of five laborers. According to RACPM, a new resource link of "B follows C" is identified as some laborers involved with activity C will later work on activity B. If we add this link to the original AON and remove the redundancies, the updated AON will

be built (Figure A.2). Results of CPM performed on updated AON are presented in Table A.3, which complies with the resource limit.

Table A.2 Levelled resource plan

| Labor ID | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 |
|---|---|---|---|---|---|
| LB1 | A | A | | | |
| LB2 | C | C | C | B | B |
| LB3 | C | C | C | B | B |
| LB4 | C | C | C | | |
| LB5 | D | D | D | D | D |

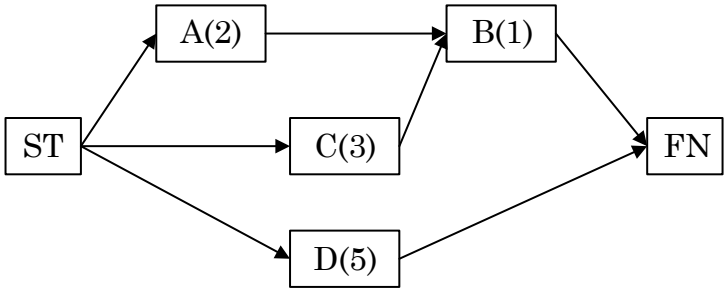



Figure A.2 Updated AON

Table A.3 Updated resource requirement

| Activity | Duration | Labor Requirement | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 |
|---|---|---|---|---|---|---|---|
| A | 2 | 1 | 1 | 1 | | | |
| B | 1 | 2 | | | | 2 | |
| C | 3 | 3 | 3 | 3 | 3 | | |
| D | 5 | 1 | 1 | 1 | 1 | 1 | 1 |
| Labor Requirement | | | 5 | 5 | 4 | 3 | 1 |

# Appendix B Network Transformation

Critical path analysis on a network with non-FS precedence relationships and lags is referred to as precedence diagram method (PDM). Lu and Lam (2009) presented transform schemes to convert PDM into a network with only FS relationships and without lags. Figure B.1 illustrates a selection of these schemes.

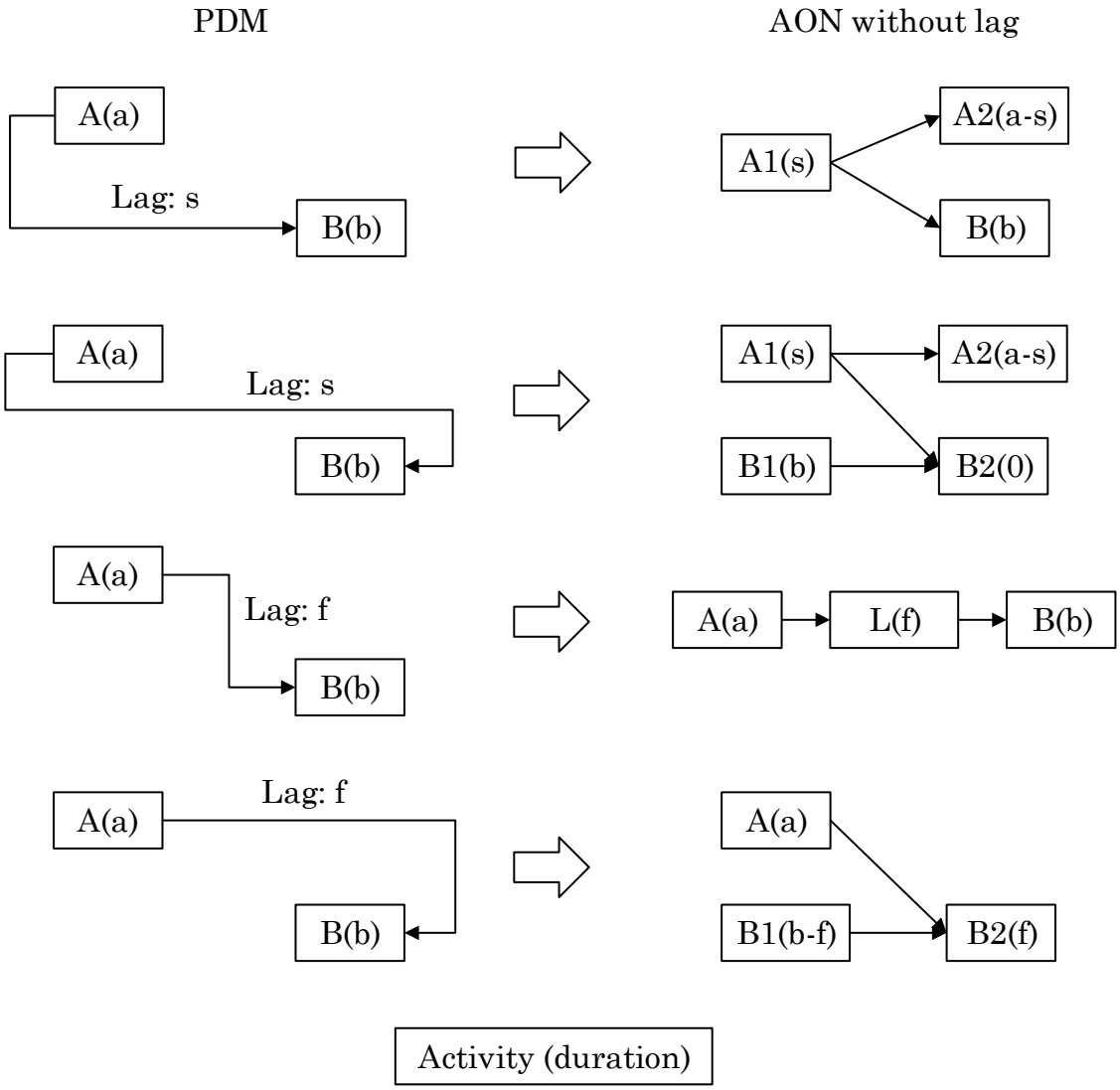



Figure B.1 Sample transformation schemes

# Appendix C Algorithm Setup in Excel

The proposed methodology can be easily set up in Excel. Solver (a free add-in for Excel) can solve the IP required in algorithm. In this section, the first iteration of the case presented in Section 4.2 (case two) is used for illustration, with IP formulation presented below. The only critical path is ACF, with all activities having available crash time.

Minimize $100 \times x_A + 80 \times x_C + 140 \times x_F$

Subject to $x_A = 0,1;\ x_C = 0,1;\ x_F = 0,1$

$x_A + x_C + x_F \geq 1$

At first, a range of cells is assigned to the binary decision variables representing activities that have available crash time. Subsequently, the path constraint imposed by Eq. 3.10 is defined using the decision variables (Figure C.1). Note that the constraint of binary variables will be defined later in the Solver itself.

| | A | B |
|---|---|---|
| 1 | Objective Function | 0 |
| 2 | | |
| 3 | Xa | |
| 4 | Xb | |
| 5 | Xc | |
| 6 | Xe | |
| 7 | Xf | |
| 8 | | |
| 9 | ACF | =B3+B5+B7 |

Figure C.1 Definition of path constraints

Next step is to define the objective function using decision variables (Figure C.2).

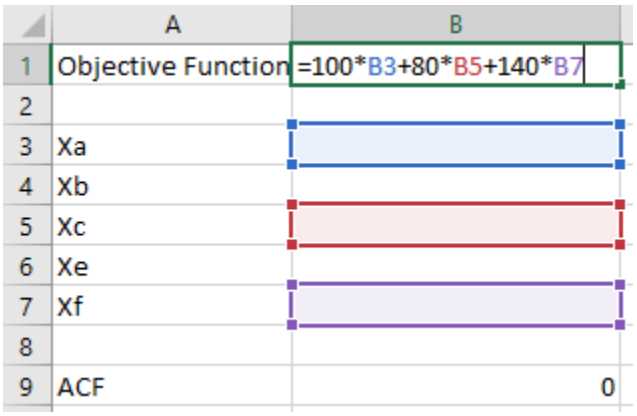



Figure C.2 Definition of objective function

Now, we must go to Data tab, analyze, and open solver. In this part, objective function, variable cells, and constraints should be specified (Figure C.3). As illustrated in Figure C.3, optimization goal should be set to minimize and simplex must be selected as the solving method. In order to adjust the solving method for binary programming, click on options in solver dialog box (Figure C.3) and set the integer optimality percentage to zero (Figure C.4). Process of specifying binary and path constraints is illustrated in Figure C.5. Finally, click on solve to obtain the results (presented in Figure C.6). In this case, activity C should be crashed with a combined cost slope of $80 per day. Solver can also provide a solution report on a separate sheet. In each iteration, cost slopes in objective function must be updated if necessary. If a new critical path is formed, corresponding path constraint must be added. Furthermore, activities on the new critical path that have available crash time must be added to objective function (if not included previously).

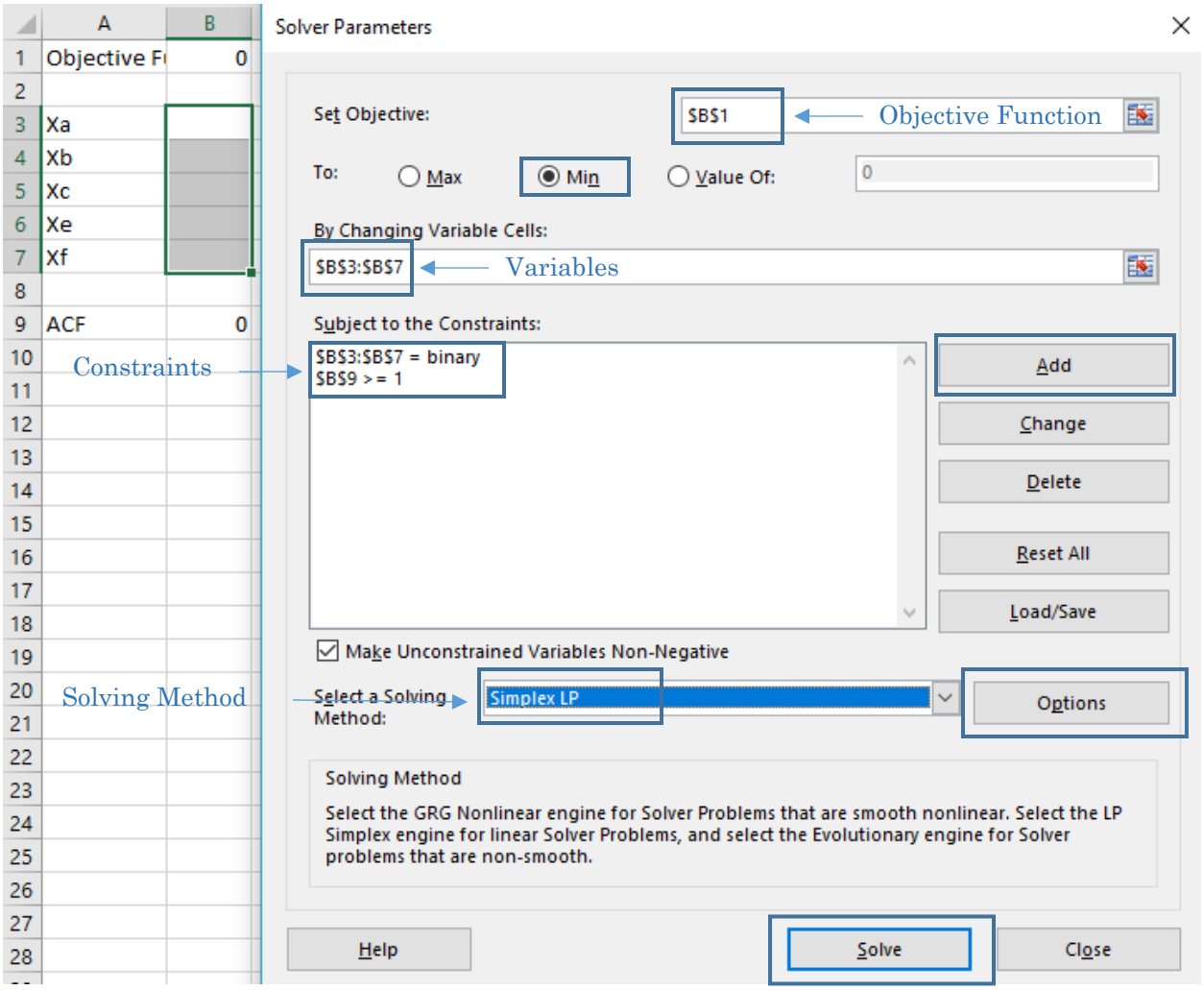



Figure C.3 Solver dialog box

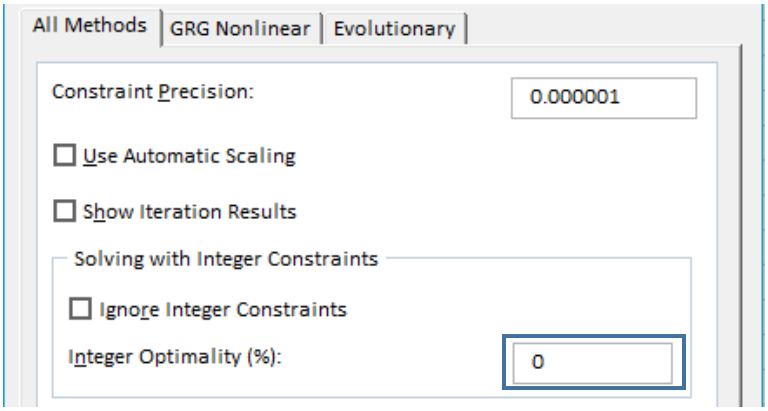
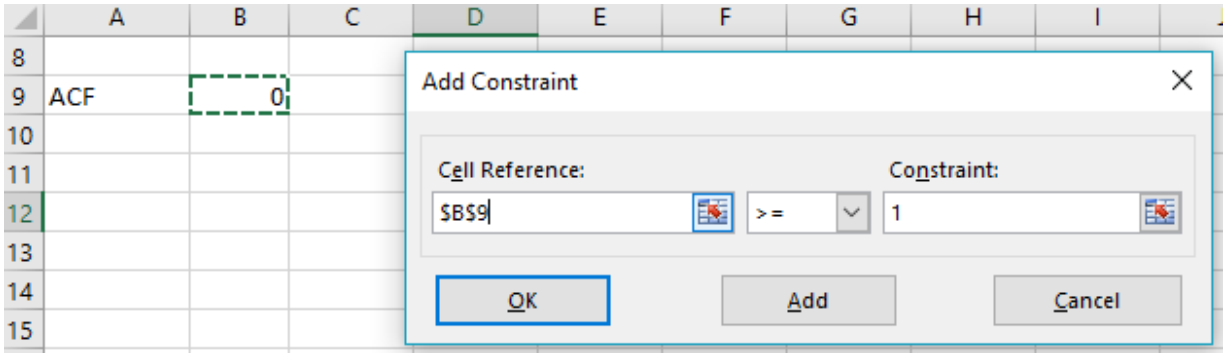
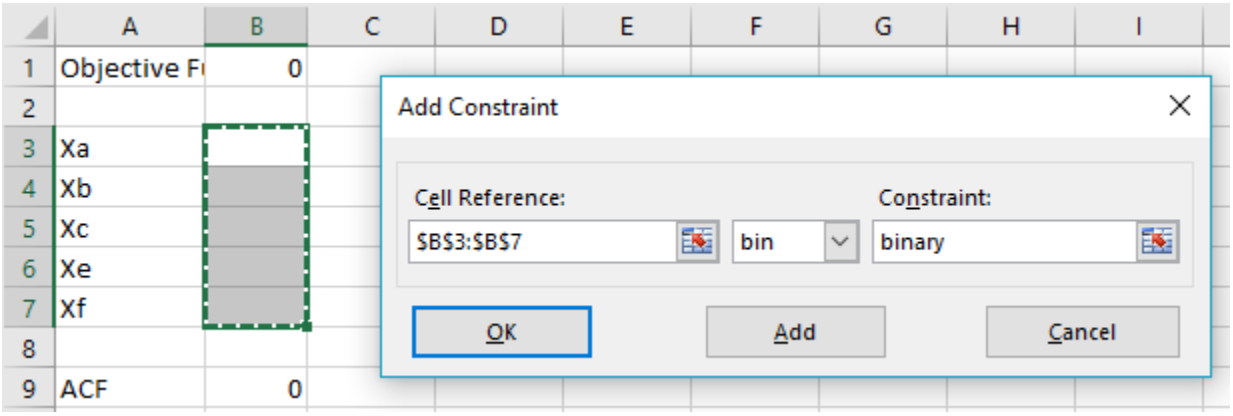



Figure C.4 Integer optimality percentage




(a) Specifying path constraints




(b) Specifying binary constraints

Figure C.5 Constraint specification

| | | |
|---|---|---|
| 1 | Objective F | 80 |
| 2 | | |
| 3 | Xa | 0 |
| 4 | Xb | 0 |
| 5 | Xc | 1 |
| 6 | Xe | 0 |
| 7 | Xf | 0 |
| 8 | | |
| 9 | ACF | 1 |

Figure C.6 Optimization results