HYBRID PARTICLE SWARM OPTIMIZATION ALGORITHM FOR OBTAINING PARETO FRONT OF DISCRETE TIME-COST TRADE-OFF PROBLEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SAMAN AMINBAKHSH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CIVIL ENGINEERING

JANUARY 2013

Approval of the thesis:

HYBRID PARTICLE SWARM OPTIMIZATION ALGORITHM FOR OBTAINING PARETO FRONT OF DISCRETE TIME-COST TRADE-OFF PROBLEM

submitted by **SAMAN AMINBAKHSH** in partial fulfillment of the requirements for the degree of **Master of Science in Civil Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. Ahmet Cevdet Yalçıner
Head of Department, **Civil Engineering** ________________

Assoc. Prof. Dr. Rıfat Sönmez
Supervisor, **Civil Engineering Dept., METU** ________________

Examining Committee Members:

Asst. Prof. Dr. Metin S. Arıkan
Civil Engineering Dept., METU ________________

Assoc. Prof. Dr. Rıfat Sönmez
Civil Engineering Dept., METU ________________

Prof. Dr. M. Talat Birgönül
Civil Engineering Dept., METU ________________

Asst. Prof. Dr. Aslı Akçamete
Civil Engineering Dept., METU ________________

Ender Şenkaya, M.Sc.
Partner, Angora Beach Resort ________________

Date: 25.01.2013

ABSTRACT

HYBRID PARTICLE SWARM OPTIMIZATION ALGORITHM FOR OBTAINING PARETO FRONT OF DISCRETE TIME-COST TRADE-OFF PROBLEM

Aminbakhsh, Saman
M.S., Department of Civil Engineering
Supervisor: Assoc. Prof. Dr. Rıfat Sönmez

January 2013, 92 Pages

In pursuance of decreasing costs, both the client and the contractor would strive to speed up the construction project. However, accelerating the project schedule will impose additional cost and might be profitable up to a certain limit. Paramount for construction management, analyses of this trade-off between duration and cost is hailed as the time-cost trade-off (TCT) optimization. Inadequacies of existing commercial software packages for such analyses tied with eminence of discretization, motivated development of different paradigms of particle swarm optimizers (PSO) for three extensions of discrete TCT problems (DTCTPs). A sole-PSO algorithm for concomitant minimization of time and cost is proposed which involves minimal adjustments to shift focus to the completion deadline problem. A hybrid model is also developed to unravel the time-cost curve extension of DCTCPs. Engaging novel principles for evaluation of cost-slopes, and pbest/gbest positions, the hybrid SAM-PSO model combines complementary strengths of overhauled versions of the Siemens Approximation Method (SAM) and the PSO algorithm. Effectiveness and efficiency of the proposed algorithms are validated employing instances derived from the literature.

Throughout computational experiments, mixed integer programming technique is implemented to introduce the optimal non-dominated fronts of two specific benchmark problems for the very first time in the literature. Another chief contribution of this thesis can be depicted as potency of SAM-PSO model in locating the entire Pareto fronts of the practiced instances, within acceptable time-frames with reasonable deviations from the optima. Possible further improvements and applications of SAM-PSO model are suggested in the conclusion.

Keywords: Discrete Time-Cost Trade-Off Problem, Time-Cost Curve Problem, Pareto Front, Particle Swarm Optimization, Hybrid Algorithm

ÖZ

KESİKLİ ZAMAN-MALİYET ÖDÜNLEŞİM PROBLEMLERİNDE PARETO EĞRİSİNİN MELEZ KUŞ SÜRÜSÜ OPTİMİZASYON ALGORİTMASI İLE OLUŞTURULMASI

Aminbakhsh, Saman
Yüksek Lisans, İnşaat Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Rıfat Sönmez

Ocak 2013, 92 Sayfa

İnşaat projelerinin süresinin kısaltılması maliyetleri düşürebileceğinden hem işveren hem de müteahhit açısından önemlidir. Ancak, projelerin hızlandırılması ek maliyetlere neden olmakta ve sadece belirli bir sınıra kadar toplam maliyetleri düşürebilmektedir. İnşaat yönetiminde büyük önem taşımakta olan, süre ve maliyet arasındaki bu ödünleşimin analizi, zaman-maliyet ödünleşim (TCT) optimizasyonu olarak adlandırılmaktadır. Mevcut ticari yazılımlar ve literatürde önerilen yöntemler kesikli zaman-maliyet ödünleşim probleminin (DTCTP) çözümü için son derece sınırlı çözümler üretebilmektedir. Bu doğrultuda, bu tezde DTCTP'nin üç farklı türü için, değişik kuş sürüsü algoritmaları (PSO) geliştirilmiştir. Önerilen yalın-PSO algoritması zaman ve maliyetin birlikte minimize edilmesini mümkün kılmakta ve küçük değişiklerle, zaman sınırlı problem için de sonuç elde edilmesini sağlamaktadır. Bir diğer model ise, DTCTP'nin zaman-maliyet ödünleşim eğrisinin elde edilmesi için geliştirilmiştir. Bu model doğrultusunda oluşturulan melez algoritmada maliyet eğrileri ve pbest/gbest pozisyonlarının değerlendirilmesi için yeni yöntemler önerilmiş ve aynı zamanda Siemens Yaklaşım Metodu (SAM) ve PSO algoritmasının güçlü özellikleri entegre edilmiştir. Önerilen algoritmaların etkinliği ve performansı literatürden alınmış örneklerle gösterilmiştir.

Sayısal deneyler esnasında, karışık tamsayı programlama tekniği vasıtasıyla, iki denektaşı probleminin optimal tam Pareto eğrileri literatürde ilk kez belirlenmiştir. Bu çalışmanın bir diğer önemli katkısıysa, geliştirilen SAM-PSO algoritmasının örnek problemlerin tam Pareto eğrisini kısa bir süre içerisinde optimum eğrilerden makul bir sapma ile elde edilebilmesidir. SAM-PSO'nun kullanılabileceği diğer alternatif uygulamalar ve geliştirilebileceği potansiyel alanlar sonuç kısmında önerilmiştir.

Anahtar Kelimeler: Kesitli Zaman-Maliyet Ödünleşim Problemi, Zaman-Maliyet Eğrisi Problemi, Pareto Eğrisi, Kuş Sürüsü Algoritması, Melez Algoritması

This thesis is dedicated to my beloved family

ACKNOWLEDGEMENTS

The best and the worst moments of my Master's journey have been shared with many people. One of the joys of completion is to look over the journey past and remember all the friends and family who have helped and supported me along this long but fulfilling road. It has been a great privilege to spend some years in the Construction Engineering and Management division of Middle East Technical University; members of which will always remain dear to me. There are a number of people without whom this thesis might not have been written, and to whom I am greatly indebted.

First and foremost, I would like to extend my sincere thanks to my supervisor, Assoc. Prof. Dr. Rıfat Sönmez, who offered his continuous advice and encouragement throughout the course of this thesis. I attribute the level of my Master's degree to his encouragement and effort and without him this thesis, too, would not have been completed or written.

I take this opportunity to offer my sincerest gratitude with all of my heart to my beloved parents, Mohammad Aminbakhsh and Simin Nahaei. They raised me, supported me, taught me, and loved me; living thousand miles apart from them was not easy at all. I also wish to thank my lovely grandmother for her love and affection, not to mention her frequent morale booster phone calls.

I would like to show my greatest appreciation to my roommate, best friend, greatest thing that has ever happened in my life, sibling, or maybe it is just better to call him my brother, Sina Aminbakhsh. His love and continuous support, both spiritually and materially, have been great sources of motivation and encouragement at every stage of my life.

I am especially grateful to Marjan and Mozhgan Khoylou sisters for helping me get through the difficult times, for all the good memories and all the emotional support, camaraderie, entertainment, and caring they provided.

Last but not least, I would like to express my thankfulness to Özgür Dedekargınoğlu and Murat Ayhan, for their precious friendship, and for all the academic and non-academic supports they provided throughout this journey. They were the persons who made me feel at home in a foreign land.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
ACS	Ant Colony System
ANOVA	Analysis Of Variance
AoA	Activity on Arrow
AoN	Activity on Node
APD	Average Percent Deviation
Avg.	Average
CPM	Critical Path Method
CPU	Central Processing Unit
cs	Centiseconds
CS	Cost Slope
dc	Direct Cost
D-PSO	Discrete Binary Particle Swarm Optimization
DTCTP	Discrete Time-Cost Trade-Off Problem
Dur.	Duration
ES	Early Start
FMOPSO	Fuzzy Multi-Objective Particle Swarm Optimization
GA	Genetic Algorithm
gbest	Global Best Position
GHz	Gigahertz
GMIR	Graded Mean Integration Representation
HA	Hybrid Genetic Algorithm
ic	Indirect Cost
MA	Memetic Algorithm
MAWA	Modified Adaptive Weight Approach
MOW	Multi-Objective Weighting
M-PSO	Modified Particle Swarm Optimization
NA-ACO	Non-dominated Archiving Ant Colony Optimization
NP-Hard	Non-Polynomial Hard
pbest	Personal Best Position
PD	Percent Deviation
PSO	Particle Swarm Optimization
QSA	Quantum Simulated Annealing
RAM	Random Access Memory
SA	Simulated Annealing
SAM	Siemens Approximation Method
SFL	Shuffled Frog Leaping
TCT	Time-Cost Trade-Off
TCTP	Time-Cost Trade-Off Problem
USD	United States Dollar
WM-ACO	Weighted Method Ant Colony Optimization

CHAPTER 1

INTRODUCTION

Due to ever evolving competition among construction firms, besides, owing to the intrinsic challenges associated with the construction projects, the prerequisite for a company to survive is to perform profound appraisals in preparation of the project schedules. Opposed to other industries, transient nature of the construction projects impose heavy burden to decision makers regarding unequivocal optimal devotions of time, cost, and resources. Conflicting aspects of planning coupled with other impartible components of prosperous project plans further narrow the field for the project management teams; such aspects involve provisions of safety and productivity upkeeps in the interim of the planning phase. Accordingly, uniting multidisciplinary collaborations, construction companies seek to develop realistic schedules with systematic updating techniques. Evidently, any company would fail to meet the anticipated resolutions in the absence of a decent schedule.

Classical network analyses like critical path method (CPM), in essence, merely incorporate the time aspect. Such methods attempt to minimize the project duration regardless of the availability of resources (both money and physical resources). Even suchlike analyses necessitate a number of assumptions in coping with constriction endeavors. Logical relationships, lag times, working calendars, resource requirements, and contingency plans are some of the many essential concerns amidst preparation of a schedule. An inadequate schedule might induce misery for a company, grounding for financial losses, dissatisfied customers, disputes, bad reputation, and so forth.

Almost every construction project involves a completion deadline determined in the contract by the client. This date is generally obtained by means of the network analyses. For such limitations, resource overloads are usually provisioned by recruiting subcontractors or directing alternative resource supplies. In addition, it is common for construction projects to seek meeting shorter than prescribed deadlines, in pursuance of making more profits. Any reduction in project duration is facilitated by compression or acceleration of the schedule. Decision makers speed up the project forcing least additional costs by deploying the slack times of the networks along determining the best combination of alternatives for realization of the activities. This task is facilitated by providing the best balance between the direct and indirect costs of a project, since, exposed to schedule accelerations, they exhibit inverse oscillations.

This trade-off between time and cost of projects are dubbed as time-cost trade-off problem (TCTP), which is one of the most prominent aspects of the project management. For the first time, considerations of TCT problem have emerged almost half a century ago in 1960s. Ever since, it has caught attention of numerous researchers and as a result, several studies have been devoted to address this problem in the literature. The discrete version of this problem considers discrete sets of time-cost options for the activities. Such a concern is imperative to TCT practices as they are commonplace in real-life projects; besides, time-cost function of any type can be estimated by discrete functions.

A real-life construction project almost always involves hundreds or even thousands of multi-modal activities. Projects of such kind are classified as Non-Polynomial hard (NP-hard) problems that require concurrent searches over the solution space, owing to the reason that any escalation in the size of the project significantly augments the required time to determine the optimal combination of options. Due to this inherent complexity, discrete TCT problems are unraveled merely for small instances.

In the literature, the discrete TCT problem has been studied under three classes of deadline, budget, and time-cost curve problems. Deadline problem minimizes the total cost taking into account an upper boundary for the duration of the project. Whilst, the budget constraint minimizes the duration of the project without exceeding an upper boundary set for the budget. It is traced to find the solutions with minimum amounts of cost or duration while satisfying any of these constraints. The latter state of the discrete TCT problem maps optimal total costs to any feasible completion time to generate a set of so-called non-dominated solutions. It is extensively acknowledged in the literature (e.g. Zheng et al. 2005, Yang 2007a, Eshtehardian et al.2008) that the ultimate resolution of TCT analyses is to generate all the non-dominated solutions, hailed as the Pareto front, for all the practical realizations of project duration.

Although implementation of the scheduling principles appears to be straightforward, the commercial scheduling software packages virtually provide inadequate strategies for TCT analyses, signalizing presence of a gap between the theoretical attainments and the practical exertions. On the other hand, exhaustive enumerations fall short of delivering an efficient and convenient mean for discrete TCT analyses. Existence of such gap coupled with inefficiencies of the existing procedures, have initiated development of several algorithms and heuristics addressing the optimal or near-optimal solution for this problem. The researchers, benefitting from the ever evolving computer science technologies have established numerous optimization methods, mainly involving exact methods (such as linear programming, dynamic programming, branch-and-bound methods), heuristic, and meta-heuristic algorithms (such as genetic algorithms (GAs), particle swam optimization method (PSO), and so forth). Though, none of the proposed methods are without deficiencies. The proposed exact algorithms, while necessitating massive computational resources, are incapable of solving large problems. They are more difficult to implement and are prone to being stagnated in local optima in non-convex solution spaces (De et al. 1995, Feng et al. 1997, Eshtehardian et al. 2008, Afshar et al. 2009). The studies providing heuristic algorithms acknowledge that they are problem dependent and cannot handle large-scale problems efficiently (Siemens 1971). Most of the heuristics assume merely linear time-cost functions and they fail to solve time-cost curve problems (Feng et al. 1997, Zheng et al. 2005). Besides, main deficiency of the existing meta-heuristic algorithms is observed as their inability to escape from local optima (Zheng et al. 2005, Sonmez and Bettemir 2012).

In fact, exact procedures are the only methods guaranteeing optimality of the solutions; nevertheless, heuristics and meta-heuristics are incapable of securing the optimality of the solutions. Compared to exact procedures, heuristics and meta-heuristics demand insignificant computational efforts to determine optimal or near-optimal solutions, within acceptable time-frames. In recent years, in order to improve the convergence capabilities of the meta-heuristics, researchers have focused on hybrid algorithms to combine the complementary strengths of different procedures. Results of the Hybrid algorithms reveal that they are capable of dealing with real-world instances more efficiently than the sole algorithms.

In this thesis study, it is perceived that relatively scarce devotion is made toward identification of the complete Pareto front for discrete TCT problems. Owing to this inadequacy, it is of great importance to provide the decision makers with robust techniques to take on the TCT analyses. It is also observed that a relatively scant work has been carried out to adopt particle swarm optimization (PSO) method in discrete TCT problems. Consequently, due to efficiency of meta-heuristics along with the robustness of the PSO algorithm, this study aims to take on different paradigms of the PSO algorithm in analyzing three extensions of the TCT problems.

The main objective of this study is to present a state-of-the-art model with higher efficiency and improved accuracy, which is capable of exerting the time-cost curve problem, i.e., identifying the complete Pareto front for larger discrete TCT networks. To this end, heuristic method of Siemens (1971) along with PSO algorithm are recruited. Primarily, overhauled versions of the Siemens Approximation Method (SAM) and the PSO algorithm are introduced; thereby, a hybrid-PSO model is generated exploiting the merits of the modified-SAM method augmented with the global convergence capabilities of the proposed PSO algorithm. The hybrid algorithm contrasts with the previous studies both in terms of the approach taken to generate the first population, and in terms of the objective function used to evaluate pbest and gbest of the particles. The SAM-PSO model is envisioned to support decision makers in competent evaluations of the subsequent “what if” scenarios.

All the proposed algorithms have been implemented in C++ programming language using the Microsoft Visual Studio 2010. Well-known problems obtained from the literature are fed into the PSO optimizers, and experiments have been directed to validate their potencies accordingly. In addition, on the verge of performance assessments, all the instances are solved to optimality by dint of mixed integer programming using the AIMMS optimization software. The quality of solutions obtained from the PSO algorithms are measured using the optimal solutions; the average deviations are then evaluated for multiple experimental runs. Moreover, the time-frames required to unravel the test problems are also determined. The results reveal that the proposed algorithms are successful for providing optimal or near optimal solutions for the discrete TCT problems. The SAM-PSO was able to obtain adequate Pareto fronts for every discrete TCTP, within reasonable computational time.

The sequel of the thesis is organized as follows. Chapter 2 starts with a brief introduction on CPM and TCTP, followed by detailed review of existing optimization techniques dealing with TCT analyses. Chapter 3 presents the main body of this work, illustrating the particle swarm optimizers followed by description of the novel hybrid algorithm, and its implementation for solution of optimization problems. Chapter 4 presents TCT analyses of sample problem sets, followed by results of the computational experiments. Chapter 5 includes the conclusions and points out potential topics for future research.

CHAPTER 2

LITERATURE REVIEW

In this chapter, the principles of project scheduling are summarized. First, Critical path method (CPM) is described along with the types of the expenses imposed to the contractors. The time-cost trade-off problem (TCTP) is presented in addition to some of the related optimization techniques. Along with the other proposed methods in the literature, prospects of particle swarm optimization (PSO) in TCT problem are then presented.

2.1. CPM

Scheduling can be defined as the appraisal of timing and sequence of a project’s actions which facilitates determination of the overall completion date (Mubarak 2010). In the course of scheduling, network analysis – a generic term for various planning methods – is always exploited (Lock 2007).

Being one of the most widespread scheduling and network analysis techniques, the critical path method (CPM), involves determination of the longest path through the network. It facilitates defining a project’s duration in conjunction with the shortest amount of time required to complete the project (Kerzner 2009). Adoption of this technique necessitates identification of the duration and logical relationship among the activities. Based upon the initial information set, an illustration of the schedule is prepared either by using the activity on arrow (AoA) or the activity on node (AoN) notation systems. As the names imply, in the first system (AoA) the activities are represented by arrows intersecting nodes which resemble events; while, in the second system (AoN) activities are represented by nodes and the logical relationships are traced by arrows. Courtesy of the ensuing reasons, the activity on node system of notation is recruited in this thesis:

- Compared to activity on arrow diagrams, they are more easily understood due to their resemblance with the engineering flow diagrams.
- Starts and finishes of activities that do not directly correspond to their immediate predecessor and successor activities are clearly demonstrated.
- They assist designation of the activities capable of overlapping each other or, the contrary, identification of the activities that must be delayed imposing lag times.
- Widespread adoption of the activity on node systems by computer programs opposed to the activity on arrow system.

Accordingly, the activity on node system of notation is generally preferred and has emerged as the prominent method used for the scheduling (Lock 2007).

In a CPM schedule, all the remaining paths of a network are either equal in length to or shorter than the critical path. The time discrepancy between the latest possible completion times of each activity without affecting the completion of the overall project, and the planned completion date is known as the slack time. The slack times of the activities are determined by forward-pass and backward-pass calculations in finding activities' early start/finish and late start/finish times, respectively. If the admissible delay in start/finish time of an activity – slack time – is equal to zero, this indicates that the concerned activity must be commenced as soon as its predecessors are accomplished; these type of activities which constitute the critical path(s) of the network are called critical activities.

Virtually every construction project has an overall completion deadline determined by the client that is specified in the contract. Some of these deadlines are assigned strictly not allowing for any delays in the completion of the project. Thus, amidst scheduling phase of a construction project, the procedure must be time-limited if the completion date is considered as the chief objective. The prevailing resolution in this case is to warrant the project will be accomplished on the definite date. This date is usually the earliest possible completion date designated via network analysis (indicated by the CPM); however, the completion date might also be targeted on a later date. Any projected resource over-allocations must be accepted for a time-limited scheduling, possibly assuming that either resource overloads can be alleviated through hiring subcontract workforce or by making alternative short-term resource provisions.

2.2. TCTP

Obviously, it is an important resolution for both the contractor and the client to finish the project on or ahead of the schedule. Unambiguously, finishing on or under the specified budget is another favorable achievement. Accordingly, simultaneous realization of these two objectives is undeniably desirable for both the parties. Considering the slack time in conjunction with the possible crashing alternatives of the activities, a project manager can speed up the project to meet a predetermined deadline with imposition of least additional costs. For this purpose, respecting the crashing alternatives of the tasks, a manager evaluates the cost per unit time (Cost Slope) as well as a feasible budgets (cash-flow) region for the project. Hence, one of the dominant prospects of the network analysis can be concluded as finding a solution that not only satisfies the completion deadline, but also has the lowest feasible total cost that resides within the feasible budget boundaries.

Rather than targeting to meet a prescribed completion deadline, there might be a couple of other reasons for a contractor to speed up a project. For instance, a contractor with good economic conditions might esteem to expedite a project in order to be able to start another earlier; to wit, to make more profit. Usually, the contractor knows the required date to mobilize to another project; and thus, this may involve compressing the current project in favor of freeing required resources so that they can be allocated to the new project. However, accelerating a project schedule might prove profitable only up to a certain level; since, it gets costly to diminish the project duration below a certain limit. The act of accelerating (compressing) or crashing a project schedule literally means reducing the duration of a project. Though, the distinction between these two approaches must first be clarified. Whilst both the techniques aim at advancing the completion date of the project, accelerating does not necessarily mean targeting to reach the least possible duration.

It must be also clarified that the possibility of schedule acceleration or availability of the crashing alternatives for the activities of the projects is highly affiliated with the typology of the projects. Those with strict logical relationships among the activities, such as high-rise buildings, usually offer less flexibility regarding the schedule acceleration compared to other types, such as pipeline projects.

According to the Construction Industry Institute (Force 1988), there are more than 90 techniques to compress a schedule. Reviewing these techniques, Mubarak (2010) has abstracted the ensuing 9 methods to accelerate a project schedule:

- Review or evaluate the schedule to discover any errors or imperfect logical relationship or constraints
- Fast-track achieving project objectives
- Carry out constructability studies and value engineering
- Assign over-time schedule either by increasing the hours per day and/or days per week
- Devote incentives for more productive workers or crews
- Allocate more human resources
- Undertake special construction method using specific materials and/or equipment to expedite the project
- Revamp project management and improve supervision
- Prevent communications breakdowns among parties

Throughout a project, the main expenses a contractor has to cope with are often classified into two categories; Direct costs and Indirect costs. The main principle for distinguishing the direct expenses from the indirect costs can be depicted as a direct cost item is directly associated with an explicit work item. Direct expenses may encompass labor, material, equipment, subcontractor, machinery, and other costs related to fees and permits; whereas, the indirect costs might bear project overhead, and general overhead expenditures.

Overhead expenses might incorporate salaries of the guard, cook, and office personnel as well as the energy costs. It is imperative for a contractor to regard that the amounts of direct and indirect costs are susceptible to schedule compression; they are usually impacted in an opposite manner as the acceleration takes place.

Despite the fact that staffing more crews or assigning them to work over-time decreases the productivity and, in turn, increases the ratio of unit cost per unit output, though, either of these approaches can be adopted should a schedule acceleration required. Therefore, in general, direct costs get increased in case of project acceleration; the more the acceleration is augmented, the more the daily cost of acceleration increases. As shown in Figure 2.1, this progression is usually nonlinear.

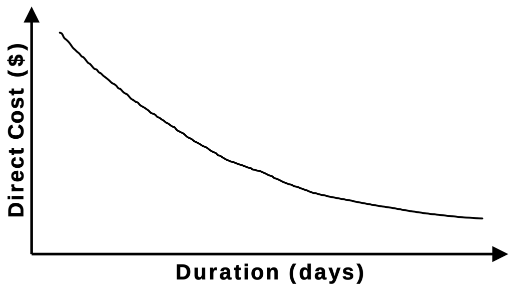

Figure 2.1 – Project acceleration results in a nonlinear escalation of direct costs.

As depicted previously, indirect costs encompass mainly the overhead expenditures which vary on daily basis in accord with the climatic conditions and the number of the staffed personnel. Nevertheless, for the sake of simplified cost computations, they are usually assumed to be linearly comparative to the duration of the project; besides, due to exact same reason, second order cost components, e.g. insurance and bond payments, are excluded from the daily indirect expenses. Hence, in case of schedule acceleration, the indirect costs decline at a constant rate (Figure 2.2).

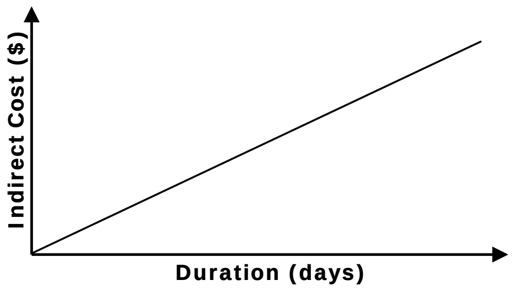

Figure 2.2 – Project acceleration cause linear decline in indirect costs.

As for the total cost curve, as shown in Figure 2.3, the total cost initially declines at a diminishing rate till it reaches a minimum amount. This point resembles the least total cost for the project. As the acceleration continues, total cost increases till the project's least possible duration is met.

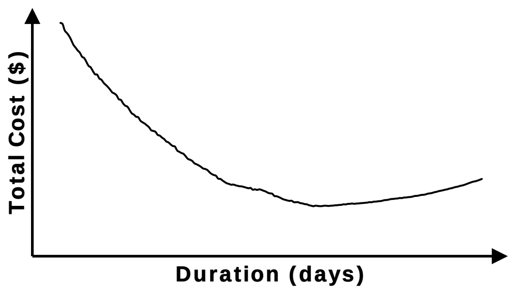

Figure 2.3 – Variation of total cost as a result of project acceleration.

According to the trend of project's total cost as a function of duration, it can be inferred that, in general, the less expensive the recruited resources are, the more time it takes to complete the tasks (Feng, Liu et al. 1997). This trade-off between time and cost of projects are termed as time-cost trade-off problem (TCTP) which is one the prominent aspects of the project management and there have been much considerations devoted to this problem in the literature.

On the verge of time–cost trade-off analysis, it is the lowest feasible total cost that gets assessed within the feasible budget boundaries, usually in accord to a specific deadline. Hence, a TCT problem in essence, is an optimization problem that attempts to reduce the project duration through accelerating the critical activities while relaxing non-critical ones to narrow the expenses (Siemens 1971). TCT aims at providing an optimal balance of project duration and cost by analyzing different combinations of decisions.

A real-life construction project almost always has hundreds or even thousands of activities each which might have several alternatives to opt from. Accordingly, it is a challenge for the project managers to find the optimal TCT decisions; in fact, it is a Non-Polynomial hard (NP-hard) problem and may require extremely time-consuming computations. Since any variation in selection of the alternative decisions alters the project schedule, therefore, it is required to re-assess the schedule, total cost, and total duration of the project using the critical path method (CPM). Exhaustive enumeration even with very fast computers is, therefore, not a convenient and economically feasible method in TCT analysis; specifically for a real-time project comprised of numerous activities. In pursuance of overcoming the pitfalls pertinent to the exhaustive enumeration, several algorithms and heuristics have been proposed in the literature addressing the optimal or near-optimal solution for the TCT problem.

For the first time, considerations of TCT problem emerged almost half a century ago, virtually concurrently with the introduction of project analysis techniques by Fulkerson (1961) and Kelley (1961). The TCT problem has been classified into several categories, ever since it was presented in the primary studies of the early sixties. There have been numerous studies conducted covering TCT problems with linear, non-linear, and discrete objective functions. Amidst analysis of the latter state of the TCT problem, i.e., a TCT problem with a discrete objective function, attempts are raised to make appropriate decisions among the limited number of time-cost alternatives, referred to as modes. Thus, one mode is assigned for any of the activities of the project with regard to the resolution of the TCT analysis. Such a concern is imperative to TCT practices as the discrete time-cost alternatives are commonplace in real-life projects; to boot, corresponding to the prospect of the TCT problem, three types of constraints might be incorporated in the TCT analysis. The three categories encompass the Deadline, Budget, and Time-cost curve problems. As it was discussed previously, generally it is essential for both the contractor and the client to finish the project on or ahead of the schedule, or, on or under a specified budget. The first assumption, being called the Deadline problem, minimizes the total cost taking into account an upper boundary for the duration of the project. Whilst, the second approach, termed as the Budget problem, minimizes the duration of the project without exceeding an upper boundary set for the budget. It is by exploitation of deadline or budget problem that all the non-dominated solutions generate with regard to the project duration and the total cost benchmarks. Thus, the latter state of the TCT analysis, referred to as the Time-cost curve problem, aims to generate all the efficient i.e. non-dominated set of solutions.

2.3. Exact, Heuristic, and Meta-heuristic Methods

Since, all the categories of the TCT problem are Non-Polynomial hard (NP-hard) problems and that finding the optimal decisions require enormous time-consuming computations, several different algorithms have been proposed in the literature to cope with this problem. The researchers, benefitting from the ever evolving computer science technologies, seek to implement various optimization methods in their TCTP studies. TCT problem mainly involve Exact, Heuristic, and Meta-heuristic optimization algorithms.

Exact procedures, as the name implies, attempt to explore the entire solution space in finding the exact optimal solution. Accordingly, they require massive amounts of computations which, in turn, necessitate higher-spec computers as well as intricate coding procedures. Respectively, they are often charged to just reinforce the competency of the Heuristic algorithms. Nevertheless, exact algorithms are indispensible means of the optimization problems since they are the only methods capable of guaranteeing optimality of the results. Some of the most popular variants of the exact algorithms are linear programming, mixed-integer programming, dynamic programming, and branch-and-bound method.

Opposed to the exact procedures, Heuristic algorithms involve much less computational efforts and are capable of producing solutions virtually in no time; usually they can be implemented even short of a computer’s assist. Derived from the Greek word “Heuriskein” meaning “to find”, heuristics involve simple rules to discover solutions to difficult optimization problems. Nonetheless, the optimality is not guaranteed in the heuristic methods and the obtained solutions are near-optimal, but rather satisfactory solutions. The constructive and the improvement heuristics are the most revered variants of the heuristic algorithms. The former uses a stepwise procedure to generate solutions, generating them one at a time until a feasible solution is met. Generally, a feasible solution is not obtained in the course of the construction heuristics unless the conclusion of the procedure is reached. The latter type of the heuristic algorithms i.e. the improvement heuristics, initiate with a feasible solution and successively improve it via a series of modifications. In the course of this procedure, usually a feasible solution is preserved regardless of the progression of the process.

In addition, the novel algorithms inspired by the stochastic occurrences of the nature, are called Meta-heuristic algorithms. “Meta”, meaning “beyond” is an indication of higher-level algorithms when compared to the heuristics, since, heuristics are problem dependent; whilst, meta-heuristics are independent of the problem’s nature. These random search techniques unravel an optimization problem by simulating the evolution and intelligent behaviors of the natural organisms. Iterative quality of the meta-heuristic algorithms, in contrast with the heuristics, may prevent getting stuck into the local optima. This latter trait generally stipulates thorough searches within the solution space, increasing the chances of discovering the desired global optimum solution. However, similar to the heuristics, meta-heuristics are incapable of securing the optimality of the solutions; rather, they obtain near-optimal solutions in an inconsiderable amount of time, with trivial computational efforts. This category of algorithms is well associated with the modern studies, such as evolutionary computation and swarm intelligence, promising for those who would prefer fast converged near optimal solutions in the face of slow but accurate ones. Some of the most prevalent meta-heuristics are genetic algorithms (GA), ant colony optimization (ACO), particle swarm optimization (PSO), shuffled frog leaping (SFL), simulated annealing (SA), and so forth.

2.4. Exact, Heuristic, and Meta-heuristic Methods for TCTP

Conventionally, the objective function of the TCT analysis was supposed to be linear (Fulkerson 1961) and the first notable attempt in construal of the TCT problem was regarded as the heuristic method proposed by Nicolai Siemens (1971). Subsequently, the assumption of linearity was relaxed allowing for consideration of other types of the objective function, namely, concave function (Falk and Horowitz 1972), convex function (Foldes and Soumis 1993), a hybrid of concave and convex functions (Moder, Phillips et al. 1983), quadratic function (Deckro, Hebert et al. 1995), and discrete function (Skutella 1998, Zheng, Ng et al. 2004). The last type of the TCT analysis, being more tangible in real-life, constitutes a large portion of the recent studies. As a result, several different approaches have been proposed for the various time-cost objective functions; viz., linear programming, integer programming, dynamic programming, and of course, the heuristics and meta-heuristics. In this part of the chapter, various methods adopted by the researchers in analyzing the TCT problem have been abstracted following a comprehensive literature survey.

2.4.1. Exact methods for TCTP

The study carried out by Meyer and Shaffer (1965), remains the pioneering attempt in application of mixed-integer programming for solving the TCT problem. Inspired by this endeavor, Moussourakis and Haksever (2004) present a flexible mixed-integer model considering the indirect cost in the analysis, whilst, making minimal assumptions regarding the type of the time-cost objective function. Assuming no specific notation system for the network, this model tackles objective functions that are linear, piecewise linear, or discrete. The only assumption required for this approach is stated as presumption of piecewise linearity for nonlinear continuous functions. This model is capable of minimizing the total cost subject to a completion deadline; besides, should a slight modification made to the algorithm, it optimizes the total cost respecting a budget constraint. Evidently, this model assists the decision makers in measuring several different “what if” scenarios.

De et al. (1995) discuss the scant and sparse considerations of the discrete TCT problem. Conducting an inclusive literature review, they survey some the previous solution approaches and thereby identify drawbacks pertinent to these methodologies. Introducing a new dynamic programming formulation, they present a centralized approach for the deadline problem with no parallel modules. Eventually, adopting the activity on arrow (AoA) representation, they take on modular decomposition conjointly with the incremental reduction approaches for the TCT problems with parallel modules.

Demeulemeester et al. (1998) propose an exact algorithm for discrete TCT problem assuming the activity on arrow (AoA) notation system for the network analysis. A variant of branch and bound optimization model is programmed by introducing a horizon-varying approach benefiting from the Visual C++ platform. This approach iterates minimizing the total cost of a time restricted scheduling problem in accord with the interval bounded by the crash modes and normal modes of the activities. They calculate the lower boundaries by setting out convex piecewise linear underestimations for the discrete TCT curves. They propose two distinct rules to assess qualities of underestimations, involving vertical distances computations. In the course of the branching process of this model, the activity with the largest vertical distance is identified; withal, in order to promote convex piecewise linear underestimations, they separate crashing modes of the activities into two subsets. The results are validated by means of a factorial experiment and are compared to the results of Demeulemeester et al.’s (1996) study which reveals that this model is capable of solving instances with up to 30 activities having 4 crashing modes.

Primarily introduced by Yang and Chen (2000), Vanhoucke (2005) elaborates on time/switch constrained discrete TCT problems. Time/switch constraints are established by imposing specific start time and inactive time-intervals to deal with day, night, and weekend shifts for the activities. Exploiting the lower bound calculation approach which was first introduced by Demeulemeester et al. (1998), the author suggests another branch and bound algorithm coded in the Visual C++. With the activity on arrow (AoA) considerations, this study incorporates a branching algorithm that identifies those activities whose lead times are greater than their durations under the lower bound calculations. The branching process creates three child nodes as it divides the start time of activities into three sections, which, is later relaxed by imposing a dominance rule. Referring to the results and further amplifying the real-life instances of the study carried out by Vanhoucke et al. (2002), Vanhoucke (2005) validates the solutions of this algorithm. The results show that on average, this model is four times faster than Vanhoucke et al.'s (2002) approach and that it is capable of solving instances of 30 activities with up to seven modes in less than 7 seconds on average.

2.4.2. Heuristic methods for TCTP

Siemens (1971) develops a logical systematic procedure based upon intuitive logic and analysis and verifies the algorithm using an empirical instance. The author proposes Siemens Approximation Method (SAM), a heuristic method for the TCT problem suited for both manual and computer aided calculations. It is capable of solving convex nonlinear TCT problems by making multiple curvilinear approximations. The procedure initiates with the construction of the project network, and involving a number of rules attempts to expedite selected activities that incur least additional costs. The results of this model are compared to a linear programming approach which affirms that the obtained results are either nearly the same or identical to the exact algorithms. However the author acknowledges that the heuristic algorithm might shorten the duration of a project beyond the intended amount, due to crashing activities merely with the minimum cost slope considerations, and regardless of the number of different paths the activities belong to.

Vanhoucke and Debels (2007) study three extensions of the discrete TCT problem; time/switch constraints (Yang and Chen 2000), work continuity constraints (El-Rayes and Moselhi 1998), and net present value maximization (Herroelen, VanDommelen et al. 1997). Assuming activity on arrow (AoA) representation, they provide a new meta-heuristic algorithm coded in the Visual C++. The heuristic portion of the proposed algorithm involves iteration of neighborhood search and diversification steps. The former step selects the best nearby solution whilst the latter randomly chooses a crash mode while setting taboo for the frequently evaluated mode-combinations. The second portion of their algorithm includes a truncated dynamic programming which increases the duration of the non-critical activities while meeting the desired completion deadline. Administering comparisons with results of an exact algorithm, they state the new approach is capable of producing promising results for time/switch constrained and net present value versions of the discrete TCT problem.

2.4.3. Meta-heuristic methods for TCTP

Feng et al. (1997) outline the favored results of TCTP analysis as providing optimal balance of time and cost, besides, delivering a TCT curve that shows the relationship between total duration and total cost of a project. The authors addressing the inefficacy of the existing methods in coping with large-scale

TCT problems, propose a more efficient model based upon the principle of Holland’s (1975) genetic algorithm (GA). The authors argue the drawbacks pertinent to multi-objective decision-making techniques such as multi-objective weighting (MOW) and there forth incorporate Pareto front approach by introducing the convex hull. This model sets two chromosomes, i.e. decision sequences, involving normal and crash modes of the activities; thence, determines the fitness values of the solutions in accord with their minimal distances to the convex hull. Iterative cross-overs and mutations are then performed to reproduce new solutions. This algorithm retains each string for the next generation in order to avoid the problem raised by Goldberg and Segrest (1987), called the genetic drift phenomenon. The authors validate their algorithm after developing a computer program (TCGA) with an interface designed in Microsoft Excel. Results indicate that this algorithm is capable of discovering more than 95% of the optimal solutions for a discrete TCT problem comprised of 18-activities.

The GA model proposed by Zheng et al. (2005) attempts to compromise the genetic drift phenomenon by reducing the chance of getting stuck into the local optima. Thus, the authors integrate a modified adaptive weight approach (MAWA) to adjust the priority of objectives respecting the quality of the preceding generation. MAWA flags and ranks all the non-dominated solutions that are identified amid each generation and clues higher ranked solutions are more likely to survive. As the generations evolve, MAWA administers a decreasing pattern for the mutation rate to prevent premature convergence. Pareto ranking and niche formation are also incorporated to this model with the former serving as selection criterion and the latter exerting as population diversifier besides supporting uniform sampling. The authors enhance the validation process of this model by means of a prototype system which operates in conjunction with the Microsoft Project. The 18-activity instance used by Feng et al. (1997) is fed into three representative modules and the results prove robustness of the proposed module, principally for the solutions archived beyond 300 generations.

In order for project managers to incorporate uncertainties in their TCT analysis, Eshtehardian et al.(2008), in their study, integrate GA with the fuzzy sets theory of Zadeh (1965). A model to tackle stochastic TCT problem is prepared for the real-life instances throughout which triangular fuzzy numbers are assumed for direct cost of the activities rather than their probability distribution, in that, they are merely partially known. Distinct Pareto fronts comprised of the non-dominated solutions are developed with consideration of the risk attitude of the experts, defined through α cut approach. Adopting Hamming-distance for binary comparison of fuzzy numbers and employing Euclidian-distance in fitness calculations, two separate techniques are prepared to rank the alternative set of options under different values of α cut. Implementing single crossover and uniform mutation, the 18-activity problem used by Feng et al. (1997) is fitted into the GA based prototype model. The second approach, not dealing with a defuzzifier, proves to perform slightly better than the other method.

Sonmez and Bettemir (2012) addressing the inherent inadequacies of sole meta-heuristic methods along with the merits pertinent to hybrid algorithms, introduce a hybrid GA (HA) for the discrete TCT analysis. This model combines the complementary potencies of genetic algorithm (GA), simulated annealing (SA), and quantum simulated annealing (QSA) in exploration of the solution space. SA introduces a better hill climbing capability for the HA in view of attaining the global optima; whilst, QSA revamps the quality of local search for this algorithm. HA is implemented to ten benchmark instances ranging from 18 to 630 activities, by means of the Visual C++ programming language. Test problems include 18-activity TCT problem described by Feng et al. (1997), highway upgrading project derived from Chassiakos and Sakellaropoulos (2005), and a hypothetical TCT problem with 63 activities, which is introduced in the course of this study. All the instances are analyzed regarding various assumptions and constraint impositions. Compared to optimal results of AIMMS optimization software (mixed integer programming), the average percent deviation (APD) of ten runs are measured. The authors apply paired t-test to assess the significance of performance difference between HA and sole GA, results of which, verifies robustness of the proposed algorithm.

Being first introduced by Colorni et al. (1992), Ng and Zhang (2008) use an evolutionary-based optimization algorithm known as the ant colony (ACO) to analyze the multi-objective TCT problem. They propose an ant colony system (ACS) adopting the modified adaptive weight approach (MAWA) (Zheng, Ng et al. 2004) for evaluating the fitness of their solutions. Exploiting the Visual Basic platform, an optimization program is developed, by means of which, the soundness of their algorithm is tested against other analytical methods that were studied by Elbeltagi et al. (2005) previously. They conclude that the results of the proposed ACS algorithm for the 18-activity instance depicted in Feng et al. (1997) are improved compared to the basic ACO and that it is capable of solving the TCT problem with much less requirements of computational resources.

Another attempt toward combining Zheng et al.'s (2004) modified adaptive weight approach (MAWA) with ant colony algorithm is made by Xiong and Kuang (2008). MAWA, applying a search stress toward the desired optimal point at each of the iterations, enhances the ACO in construal of the optimal solutions, and delivering the Pareto front as well. Thru this method, two selections are made to decide on possible alternatives. According to the membership of a random variable, the first selection is made regarding a maximization criterion, and the other involves a probability distribution function. The performance of this prototype model is assessed using the 7-activity problem acquired from Zheng et al. (2004) and the 18-activity instance solved by Feng et al. (1997). Modifying the parameters of the proposed ACO by a sequel of trial and error, this model manages to find the same results found thru the abovementioned studies by exploring rather smaller portion of the solution space; proving to be a more efficient measure.

The discrete TCT problem is represented as a graph in the study carried out by Afshar et al. (2009). In this study, a multi-colony non-dominated archiving ACO (NA-ACO) is introduced thru which separate ant colonies are assigned to each of TCTP objectives. Solutions found by agents of each colony are iteratively transferred to the next colony to be evaluated in accord with the competing objective. New solutions are then generated regarding the updated pheromone trail. Any iteration of this algorithm concludes with reserving the non-dominated solutions in a separate offline archive. Performance of this method is measured using the 18-activity problem derived from Feng et al. (1997), and the results are compared to WM-ACO (Weighted Method Ant Colony Optimization) and the GA model of Zheng et al. (2005). The test problem is solved engaging three alternate indirect costs of 0, 200, and 1500 units and for all the instances, it manages to substantially outperform the compared algorithms. This model surpasses the previous algorithms specifically for the problems with lower incurred indirect costs; however, it lacks the competency of exerting the full Pareto front for any of the considered cases.

Elbeltagi et al. (2007) recall shuffled frog leaping (SFL) as a robust algorithm in handling complex large-scale problems due to its competency in incorporating PSO-like (particle swarm optimization) local search along with information exchange of parallel local searches. They modify the original SFL algorithm by implementing a search-acceleration parameter (C) to avoid MSFL from being stagnated at local optimums. Time-variant augmentation of this parameter sustains the balance between local and global search; that is, gradually decreasing the amount of C from a larger initial value, contributes to a wider global search followed by a deeper local search. A parametric study is conducted on this parameter toward realization of better results. The authors validate their marks by feeding their model into F8 and F10 benchmark problems, variants of 18-activiy problem derived from Feng et al. (1997), and “rehabilitation of bridge-deck infrastructure” problem acquired from Hegazy et al. (2004). Test problems are set by implementing MSFL using Visual Basic, Microsoft project, and Microsoft Excel programs and are compared to original SFL and GA algorithms. The results of this study demonstrate competency of the MSFL, as it outperforms the matched approaches requiring significantly smaller time frames to produce set of solutions with higher success rates.

Anagnostopoulos and Kotsikas (2010) analyze five variants of a simulated annealing algorithm using activity on node (AoN) type of networks. They seek applying a search method analogous to annealing process of melted materials; besides, they utilize analysis of variance (ANOVA) and Duncan Multiple Range Test to measure quality and efficiency of the solutions exposed to several problem factors. They also estimate the confidence interval for the optimal solutions pertinent to discrete TCT instances. Sample problem sets are generated randomly using RanGen2 program for the SA algorithms coded in Visual Basic programming language. Exploiting formal statistical methods in conjunction with Microsoft Excel, SPSS, and Mathcad programs, they evade from drawing conclusions flawed by sampling errors. Eventually they rank the SA variants in accord with the results of the Duncan test and estimate confidence interval of optimum solution for the best and the worst algorithms.

Yang (2007a) bases its model upon the modified PSO algorithm of Shi and Eberhart (1998) to archive the Pareto front in a single run, for the feasible project durations. Selection of pbest is modified in this PSO as to update should a strongly dominating solution emerges. Withal, gbest membership only consents particles that dominate fewest members of the library. This algorithm, not relying on metrics, is capable of handling various types of objective functions regardless of the time-cost scaling. Non-dominated solutions are iteratively stored in a separate elite library, meanwhile, deleting the dominated particles. Members of this archive aid further explorations over the search space. Indirect costs are not provisioned throughout the optimization process; rather, they are implemented exogenously after setting up the final Pareto front. The performance of this algorithm is measured by externally imposing the constraints of the deadline and budget problems to the archived solutions of a 14-activity network. Results demonstrate the efficiency of the proposed algorithm, facilitating subsequent "what if" analysis by the decision makers.

Yang (2007b), in another study, uses the PSO algorithm to analyze crashing alternatives of the budget and deadline TCT problems. Capable of treating objective functions of any type, this method aims at creating the Pareto front, so as to assist decision makers in conducting further "what if" analysis. This model is coded in MATLAB programming environment and is implemented into a hypothetical test problem, as well as a real-life highway restoration project. Indirect costs are provisioned after setting up the final Pareto front for both the hypothetical example involving an 8-activty network, and the case-study incorporating 28 activities. Parametric studies for the algorithm operators of swarm size, inertia weight, and damping limit are performed, taking on the sensitivity analyses. Adopting appropriate parameters, performance of this algorithm is assessed through measurement of the average percent deviation (APD) per ten runs. Signifying rather small percentages of deviations, the efficiency and performance of the proposed PSO algorithm is validated.

In a recent venture by Zhang and Xing (2010), the authors attempt to introduce a Fuzzy-based PSO for solving time-cost-quality trade-off problems with nondeterministic input data. Fuzzy multi-attribute utility technique derived from Keeney and Raiffa (1976), is embedded to the constrained fuzzy arithmetic operations to enhance the PSO algorithm with exploration of solutions that secure maximum quality while requiring minimum time and cost. Fuzzy-multi-objective PSO (FMOPSO) is coded in Visual C++, treating time, cost, and quality of the alternatives as triangular fuzzy numbers. FMOPSO uses fuzzy attribute utility for generating composite fuzzy utility values for each mode combination. The proposed PSO algorithm incorporates the mean integration representation (GMIR), so as to discover the solution with the largest composite fuzzy utility. The algorithm is examined using a three modal 13-activity network, and the results are compared to a fuzzy-GA algorithm, illustrating the potency of the FMOPSO.

Another PSO-based algorithm for optimizing intrinsic problems of the construction industry is developed by Ashuri and Tavakolan (2012). This study integrates the Fuzzy set theory with the hybrid GA-PSO algorithm of Juang (2004), to tackle continuous time-cost-resource trade-off problems. Triangular fuzzy numbers are presumed for nondeterministic values of time, cost, and resources; further, GA and PSO are applied to lower and upper halves of the population, respectively. The proposed hybrid GA-PSO is implemented in Delphi programming platform to construct the Pareto fronts for the 7-activity instance acquired from Zheng and Ng (2005), as well as the 14-activity network discussed in Yang (2007a). Compared to earlier methods, this algorithm is capable of finding less costly solutions that not only require shorter durations, but also, prescribe fewer variations in allocated resource. Supremacy of the proposed algorithm is further proved by contrasting the demanded processing time with the previous approaches.

A renowned study covering various evolutionary meta-heuristic algorithms is carried out by Elbeltagi et al. (2005). In the course of this work, the authors conduct a comparison among five meta-heuristic algorithms. The performance and efficiency of GA, Memetic algorithm (MA), PSO, ACO, and SFL are compared with each other with regard to the required processing time and quality of the obtained solutions. Models are coded in Visual Basic platform and fitted with three benchmark test problems, including the F8 function, F10 function, and the 18-activity network explained in Feng et al. (1997). Best-suited parameters are identified for all the approaches subsequent to a large number of trials. The discrete TCT instance is analyzed with a deadline consideration of 110 days. Executing twenty experimental runs for each of the instances, mean processing time and quality of the results are observed. The results of this comparison demonstrate a rather poor performance by the GA; whereas, the PSO algorithm manages to outperform all the other approaches, inasmuch as, it produces solutions of higher quality within an acceptable time-frame, with a greater success rate.

Table 2.1 summarizes the exact, heuristic, and meta-heuristic procedures for TCT analyses that are detailed in this section. Records are arranged in a chronological order in this table, encompassing brief explanations for each study. Aside from the adopted methods and administered problems, the implemented programming languages are also indicated for due models. The table also includes the size of the network problems fitted into the models, as well as, the required processing times (seconds), and the associated average percent deviations (APD) per multiple runs. The last two columns of this table highlight remarkable points and clear-cut drawbacks pertinent to each study. Unreported materials are tabulated as ‘na’ in Table 2.1.

Table 2.1 – Exact, Heuristic, and Meta-heuristic algorithms for TCTP.

Year of Publication	Author(s)	Method	Problem	Platform	# of Activities	Seconds	APD (%)	Remarks	Shortcomings
1971	Siemens	Heuristic (SAM)	Time-Cost	na	5	na	na	Logical systematic approach involving a number of rules for expediting the activities that incur least additional costs.	Merely considers minimum cost slope for crashing activities that might shorten the duration beyond required amount.
1995	De, Dunne, Ghosh, and Wells	Dynamic Programming	Time-Cost	na	5, 10	na	na	A centralized approach for deadline problem with no parallel modules and a combination of modular decomposition with incremental reduction approaches for problems with parallel modules.	Only effective for networks with reasonably low values of certain parameters.
1997	Feng, Liu, and Burns	GA	Time-Cost	na	18	na	na	A genetic algorithm (GA) that calculates fitness values exploiting minimal distance to convex hull and retains each string for next generation to avoid genetic drift.	Only tackles Finish-to-Start relationships neglecting possible resource constraints.
1998	Demeulemeester, Reyck, Foubert, Herroelen, and Vanhoucke	Branch and Bound	Time-Cost	Visual C++	10, 20, 30, 40, and 50	0.34, 19.17, 58.00, 105.40, and 127.56	na	Horizon-varying approach is embedded into branch and bound method and qualities of lower boundary underestimations are assessed by vertical distance computations.	Effectiveness and efficiency decreases significantly for larger networks with multiple modes.
2004	Moussourakis and Haksever	Mixed-Integer Programming	Time-Cost	na	7	na	na	Requires no network notation system and makes minimal assumptions regarding the type of TCT functions availing subsequent "what if" analysis.	Requires substantial computational resources, thus, suits small to medium instances.

Table 2.1 – Exact, Heuristic, and Meta-heuristic algorithms for TCTP (*Continued*)

Year of Publication	Author(s)	Method	Problem	Platform	# of Activities	Seconds	APD (%)	Remarks	Shortcomings
2005	Vanhoucke	Branch and Bound	Time-Cost	Visual C++	10, 20, and 30	0.004, 1.507, and 11.506	0.000, 0.003, and 0.138	Branch and bound procedure incorporating a lower bound branching algorithm that involves activities whose lead times are greater than their durations.	na
2005	Zheng, Ng, and Kumaraswamy	GA	Time-Cost	na	18	na	na	A genetic algorithm (GA) that recruits MAWA, Pareto ranking, and Niche formation to avoid genetic drift, administer selection pattern, and exert diversifier, respectively.	na
2005	Elbeltagi, Hegazy, and Grierson	GA, MA, PSO, ACO, SFL	Time-Cost	Visual Basic	18	GA (16), MA (21), PSO (15), ACO (10), SFL (15)	GA (0.022), MA (0.007), PSO (0.004), ACO (0.033), SFL (0.029)	Five meta-heuristic algorithms (GA, MA, PSO, ACO, SFL) are compared, revealing poor performance of GA as well as robustness of PSO methods.	na
2007	Vanhoucke and Debels	Meta-heuristic (Exact+ Heuristic)	Time-Cost	Visual C++	10, 20, 30, 40, and 50	0.008, 0.096, 0.337, 0.811, and 1.605	0.037, 0.050, 0.044, 0.098, and 0.114	Heuristic portion involves neighborhood search and diversification steps. The second portion of algorithm uses truncated dynamic programming to relax non-critical activities.	na
2007	Elbeltagi, Hegazy, and Grierson	SFL	Time-Cost	Visual Basic	18	8	0	A modified Shuffle Frog Leaping (SFL) algorithm that incorporates a time-variant parameter to avoid falling into local optima.	na

Table 2.1 – Exact, Heuristic, and Meta-heuristic algorithms for TCTP (*Continued*)

Year of Publication	Author(s)	Method	Problem	Platform	# of Activities	Seconds	APD (%)	Remarks	Shortcomings
2007	Yang	PSO	Time-Cost	na	14	na	na	Selection process of Particle Swarm Optimization (PSO) method is modified in favor of strongly dominant solutions as pbest, and solutions of scant areas as gbest.	Indirect costs are not provisioned throughout the optimization process.
2007	Yang	PSO	Time-Cost	MATLAB	8 and 28	48 and 600	0.406 and na	A PSO algorithm capable of handling any function type which requires manual calculations for subsequent "what if" analysis.	Indirect costs are not provisioned throughout the optimization process.
2008	Ng and Zhang	ACO	Time-Cost	Visual Basic	18	na	na	Modified adaptive weight approach (MAWA) is integrated into Ant Colony System (ACS).	Probable premature convergence with higher iterations and too sensitive to selection of parameters.
2008	Xiong and Kuang	ACO	Time-Cost	na	7 and 18	na	na	MAWA is embeded into Ant colony System (ACS) and selection of the options are made according to membership of a random variable, the first selection involving a maximization criterion, and the other incorporating a probability distribution function.	na
2008	Eshtehardian, Afshar, and Abbasnia	GA	Time-Cost	na	18	na	0.73	Fuzzy set theory enables genetic algorithm (GA) to handle stochastic TCTPs.	na

Table 2.1 – Exact, Heuristic, and Meta-heuristic algorithms for TCTP (*Continued*)

Year of Publication	Author(s)	Method	Problem	Platform	# of Activities	Seconds	APD (%)	Remarks	Shortcomings
2009	Afshar, Ziaraty, Kaveh, and Sharifi	ACO	Time-Cost	na	18	na	na	Multi-colony non-dominated archiving ACO (NA-ACO) that assigns separate ant colonies to each objective and evaluates the found solutions respecting the competing objective within the next colony.	na
2010	Anagnostopoulos and Kotsikas	SA	Time-Cost	Visual Basic	na	na	na	Performance of five variants of simulated annealing (SA) algorithm are analyzed and compared to each other.	na
2010	Zhang and Xing	PSO	Time-Cost-Quality	Visual C++	13	na	na	A Fuzzy-based PSO with quality considerations that employs fuzzy attribute utility to generate composite values.	Generates only a single optimal solution rather than the Pareto front.
2012	Sonmez and Bettemir	Hybrid-GA	Time-Cost	Visual C++	18, 29, 63, 290, and 630	na	0.00, 0.00, 2.50, 0.43, and 2.41	A hybrid genetic algorithm combining potencies of simulated annealing (SA) along with quantum simulated annealing (QSA).	na
2012	Ashuri and Tavakolan	Hybrid GA-PSO	Time-Cost-Resource	Delphi	7 and 14	348 and 1140	na	A fuzzy-based hybrid GA-PSO with resource considerations that treats lower and upper halves of population using GA and PSO, respectively.	Only handles TCRO problems with continuous functions.

It is widely documented in the literature (e.g. Zheng et al. 2005, Yang 2007a, Eshtehardian et al.2008) that the ultimate resolution of TCT analyses is to find non-dominated set of solutions, dubbed as the Pareto front, for the feasible set of durations. Experts undertake TCT analyses to come up with appropriate selection of resources such as crews, equipment, machinery, etc., required for execution of project activities. Obtaining the Pareto front assists the decision makers in construal of minimum costs associated with any of the possible completion dates. Archiving the Pareto front in essence is a bi-criterion optimization problem that attempts to concurrently solve two classical derivate of TCT problems, viz., the budget and the deadline problems.

It is also argued in the literature that, literally, every construction project involves non-renewable resources dedicated for the discrete modes of the activities. Besides, discretization might come in handy amidst approximation of any time-cost relationship. Though, in accord to its intrinsic complexity, the discrete TCT problem is unraveled merely for small instances. Broadly embraced in the literature, is the network explained by Feng et al. (1997) comprising only 18-activities. Having surveyed a relatively vast domain of the management literature, there seems to be a scant attention devoted to practical procedures capable of scrutinizing large-scale projects. It is also perceived that sporadic devotion is made toward identification of the complete Pareto front for the problems.

As long as the commercial scheduling software packages, in general, do not bear robust strategies for TCT analyses, various methods have been developed by the researchers. Though, it is noticed that most of the proposed exact algorithms suffer from incapability to simultaneously exert more than one objective. Linear programming techniques are alleged that they fail to solve instances with discrete time-cost relationships. Besides, integer programming approaches demonstrate massive consumption of computational resources as the size and complexity of multi-modal problems increases. As depicted by De et al. (1995), any exact solution algorithm for the discrete TCT problem would almost always exhibit an exponential worst-case complexity; in that, the processing time would increase in an exponential manner as the size of the problem gets augmented. It has been concluded that exact algorithms are prone to being stagnated in local optima in non-convex solution spaces (De et al. 1995, Feng et al. 1997, Eshtehardian et al. 2008, Afshar et al. 2009). Moreover, the studies recruiting heuristic algorithms acknowledge that they, analogous to exact procedures, cannot handle large-scale problems efficiently (Siemens 1971). Most of the heuristics presume merely linear time-cost relationships and they fall short of delivering the set of possible solutions (Feng et al. 1997, Zheng et al. 2005). Main deficiency of the existing meta-heuristic algorithms is observed as the chance to get stuck into local optima (Zheng et al. 2005, Sonmez and Bettemir 2012). Such a condition causes premature convergence despite of iterated process of randomly manipulating the generated solutions. Accordingly, recognizing the limitations pertinent to various optimization techniques including the sole meta-heuristic algorithms, a recent trend toward combining various optimization methods has been emerged as hybrid algorithms. They are envisaged to combine the complementary strengths of different procedures to provide more efficient approaches in dealing with complex real-world problems.

Acquiring the drawbacks inherent to most of the existing techniques, it is of great importance to develop a state-of-the-art model, capable of identifying the complete Pareto front for larger discrete TCT networks. In addition, it is inferred from the literature on meta-heuristics that, contrasting with the genetic algorithms (GAs), a relatively scarce endeavor has been carried out to implement particle swarm optimization (PSO) technique in TCT problems. Accordingly, considering the efficiency of the meta-heuristic procedures in conjunction with the robustness of the PSO method illustrated by Elbeltagi et al. (2005), two particle swarm optimizers, as well as, a hybrid PSO algorithm are developed in this thesis study, so as to escalate the global convergence capabilities of the proposed method. The hybrid algorithm is intended to help decision makers to conduct subsequent “what if” analyses efficiently.

In the ensuing chapter, characteristics of the proposed particle swarm optimizers along with the hybrid PSO algorithm developed to solve different extensions of time-cost trade-off problems (TCTP) for construction projects are going to be presented.

CHAPTER 3

PARTICLE SWARM OPTIMIZATION ALGORITHMS

This chapter is devoted to particle swarm optimizers (PSO). Initially, theoretical properties of contemporary PSO algorithms in conjunction with principles of Siemens approximation method (SAM) are clarified. Developments of two particle swarm optimizers, as well as a hybrid meta-heuristic algorithm are presented for solution of time-cost trade-off problems, contributing specific emphasis on time-cost curve extension of these analyses. To this end, slight modifications as stated herein, are made to both of the SAM and PSO methods and a new hybrid SAM-PSO meta-heuristic algorithm with improved convergences capabilities is introduced. Flowcharts are given, illustrating the infrastructures of the proposed methods; besides, implementations of the proposed methods in C++ programming language are explained in the form of pseudo-codes.

3.1. Particle Swarm Optimization (PSO)

Exquisite studies on natural biological evolution and social behavior extant in systems such as animal herds, fish schools, and flock of birds where aggregated behaviors take place, triggered origination of swarm intelligence. Millonas (1994), toward developing models for artificial life, have documented five chief principles of the swarm intelligence, owing to the mutual properties of such natural systems:

1. Proximity: Ability to conduct space and time computations.

2. Quality: Ability to respond to environmental quality factors.

3. Diverse response: Flexibility of the responses with multitude spectrum of reactions.

4. Stability: Retain unaffected mode of behavior under slight environmental changes.

5. Adaptability: Change behavior mode under rewarding external stimuli.

This collective behavior of decentralized natural organisms, which was further studied and supported by numerous researchers, established the primitive initiatives for development of the PSO algorithm. The earliest precursors of PSO were computer simulations of migrating bird flocks for visualizing swarming behavior of the species in their search for food, carried out by Reynolds (1987) and Heppner and Grenander (1990). The basic rules governing simulation experiments of the migrating birds were to match nearest neighbor velocity and to accelerate by distance. Apprehending the potency of these simulation models in optimization and analysis of sophisticated problems, Kennedy and Eberhart in 1995 developed the first paradigm of the PSO, based upon the principles of the swarm intelligence (Eberhart and Kennedy 1995, Kennedy and Eberhart 1995).

PSO algorithm is a population-based algorithm that recruits potential solutions to concurrently search the domain, that is, promising to deliver enhanced convergence capabilities. It is rooted upon imitating the choreography of bird flocks that communicate together as they fly. This algorithm conceptually resembles evolutionary strategies and has ties to genetic algorithms; in that, it vastly relies on stochastic procedures, like the evolutionary algorithms. Withal, similar to genetic algorithms, PSO involves randomly generated populations and evolves individual solutions providing cooperation and competition processes that are theoretically analogous to the crossover operations. However, unlike evolutionary algorithms, PSO mimics the social behavior of the biological agents. Opposed to genetic algorithms, PSO incorporates memory; in addition, unlike GAs randomized velocities are assigned for individual solutions.

The system initializes with a population of random potential solutions. The population is hailed as “swarm”, while, the potential solutions are termed as “particles”. The particles are flown through a multidimensional search space. Particles iteratively fly over the search space in explicit directions, and are attracted to self-attained historical best position (personal best; pbest) and to the best position among the entire swarm (global best; gbest). Each particle memorizes the coordinates associated with the best location it has visited so far. At each time step, particles evaluate their own positions with respect to definite fitness criteria, then, comparing the fitness values, they communicate to identify the particle located in the best position. Thenceforth, aiming to imitate the best bird, each bird speeds towards the best position using a velocity that incorporates coordination of the personal best location. Accordingly, at any iteration, velocity of each particle is adjusted depending on random terms, with independent random numbers being generated for acceleration toward personal and global bests. Each particle, then, evaluates the domain from its new location, and the process reiterates until either the swarm reaches to a predefined target, or a computational limit.

Considering the numbers of variables, S, PSO randomly positions N particles in a S-dimensional solution space. Each particle is initialized with position and velocity vectors of S elements. The swarm comprising N particles is defined as a set:

$$A = \{x_1, x_2, \dots, x_N\}$$

where each x_i is represented by its position as a decision vector in the search space, denoted as a set:

$$x_i = \{x_{i1}, x_{i2}, \dots, x_{iS}\}$$

where each x_{is} resembles particle i's coordinates on Sth dimension. The particles are assumed to fly within the search space, iteratively. The position change is facilitated by means of each particle's velocity, represented as:

$$v_i = \{v_{i1}, v_{i2}, \dots, v_{iS}\}$$

velocities are modified with respect to the information acquired thru the earlier steps of the algorithm. Accordingly, in addition to the swarm set, A, containing current positions of the particles, PSO retains a memory set where each particle stores the best position visited thus far. This set is defined as:

$$P = \{p_1, p_2, \dots, p_N\}$$

which contains the personal best positions (pbest) reached in previous cycles, on Sth dimension, for each particle:

$$p_i = \{p_{i1}, p_{i2}, \dots, p_{iS}\}$$

The index of the best particle among the entire swarm in the population – the lowest function value in P at a given iteration – is represented by the symbol g, and the array index of that agent is assigned to a variable called gbest.

If t signifies the iteration counter, then the current position and velocity of the ith particle will be henceforth denoted as $x_i^{(t)}$ and $v_i^{(t)}$, respectively. Accordingly, each particle updates its velocity at each time step t, using current velocity, $v_i^{(t)}$, the distance to personal best experience, and the distance to best position of the swarm. Withal, the velocity term is modified involving some randomness in the direction of pbest and gbest, so as to approach toward the best particle g, using the velocity update equation (3.1).

$$v_{ij}^{(t+1)} = v_{ij}^{(t)} + c_1 r_1 \left(P_{ij}^{(t)} - x_{ij}^{(t)}\right) + c_2 r_2 \left(P_{gj}^{(t)} - x_{ij}^{(t)}\right) \tag{3.1}$$

$$i = 1, 2, \dots, N, \quad j = 1, 2, \dots, S$$

Likewise, using the velocity in proceeding time step, $v_{ij}^{(t+1)}$, the particle's updated position is measured exploiting the position update equation (3.2).

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)} \tag{3.2}$$

where, for both Eqs. (3.1) and (3.2), subscript j denotes the dimension of the search space; r_1 and r_2 are random S-dimensional vectors with their components uniformly distributed within the range $[0,1]$; and the constants c_1 and c_2 are hailed as the cognitive and social parameters, respectively. At each time step, t, succeeding the velocity and position modifications, the performance of particles are reevaluated respecting a predefined problem-specific fitness function. Thereon, the memorized personal bests (pbest) are updated accordingly. Ultimately, redetermination of index g for the updated "pbest"s, completes a cycle of the PSO algorithm. The computation of Eqs. (3.1) and (3.2) is demonstrated in Figure 3.1. Particle 1 is directed toward a new location regarding the positions of the pbest and the gbest, in this particular case the fifth particle.

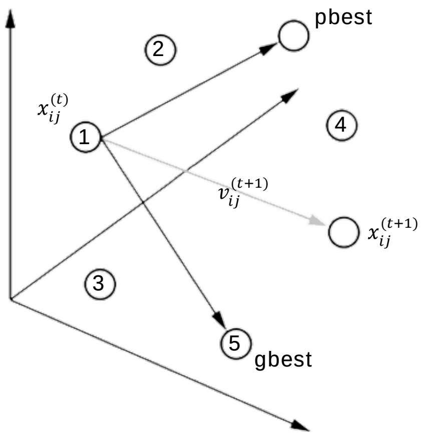

Figure 3.1 – Demonstration of PSO concepts (Yang 2007a).

On the verge of eliminating the explosion effect, i.e., the unrestrained escalation of the velocities that promotes swarm divergence, a mechanism to clamp individuals' velocities is applied. This damping factor is implemented using Eq. (3.3) prior to the position update.

$$v_{ij}^{(t+1)} \in [-v_{max} \, , v_{max}] \quad , \quad \begin{cases} v_{ij}^{(t+1)} < -v_{max} & \rightarrow \quad v_{ij}^{(t+1)} = -v_{max} \\ v_{ij}^{(t+1)} > v_{max} & \rightarrow \quad v_{ij}^{(t+1)} = v_{max} \end{cases} \tag{3.3}$$

where negative velocities are also considered, for, velocity vectors might reverse. In consensus with numerous studies, this parameter, v_{max}, is set as integer 2 for problems with continuous objective functions.

The values of c_1 and c_2 constants in Eq. (3.1), controlling the magnitude of search, can affect the convergence capabilities of a particle by biasing its movement either toward the pbest or the gbest positions, respectively. In that, in case of $c_1 > c_2$, migration would be biased toward the direction of pbest, whilst, the contrary, $c_1 < c_2$ case would favor migration toward the direction of gbest. In the literature, these two constants are commonly set to be integer 2, in favor of ascribing weighted average of 1 for the second and third terms of the equation. On the other hand, the randomness of r_1 and r_2 further adjust weightings which facilitate finding a better solution along the direction guided toward pbest and gbest.

The previous velocity term, $v_{ij}^{(t)}$, in the right-hand side of Eq. (3.1) provides the particle with an inertial movement, taking on its preceding velocity. This principle enhances the convergence capability of the swarm by avoiding biased migration towards the pbest and gbest positions. That is, this term functions as a perturbation for the global best particle, P_g. In absence of this term, the search space would shrink over generations and the global best particle might stagnate at the same location for several cycles, until identification of a better position by another particle. The ultimate generation deeply relies on the initial population (seeds), hence, most probably the swarm would perform local search short of the first term. The second term of Eq. (3.1) represents cognition, or the private thinking of the particle since compares current position to personally experienced positions. Whereas, the third term, characterizes the social cooperation among the particles, since compares current position of particles to the best position experienced by the swarm. Without these two terms, the particles will continue flying at a constant speed in the same direction. Therefore, the swarm would not be able to find any suitable solution unless meeting one on their flying trails.

3.1.1. Modified particle swarm optimization (M-PSO)

Shi and Eberhart (1998) introduced the so-called inertia weight (w) parameter for the original particle swarm optimization algorithm (PSO) of Kennedy and Eberhart (1995). Revisiting the roles for each part associated with the PSO velocity update equation, and hinting at the trade-off between local and global search capabilities of the particles, Shi and Eberhart (1998) modify the original velocity update equation by insertion of the inertia weight (3.4).

$$v_{ij}^{(t+1)} = w^{(t)} v_{ij}^{(t)} + c_1 r_1 \left(P_{ij}^{(t)} - x_{ij}^{(t)}\right) + c_2 r_2 \left(P_{gj}^{(t)} - x_{ij}^{(t)}\right) \tag{3.4}$$

$$i = 1, 2, \dots, N, \quad j = 1, 2, \dots, S$$

Coding the proposed modified PSO in Borland C++ compiler, the benchmark F6 function is solved under various w considerations. Thirty test runs are executed for each value of the w, and the number of failures (if any) is dedicated for each w. Visualizing the flying process along with the frequency of the failures, the authors conclude local search capabilities for smaller values of w, whereas, determine larger values of w treat PSO with more global search capabilities. Accordingly, the authors propose a time-variant (iteration) reduction for this parameter to enhance the algorithm with a better balance between the local and global searches, for, providing more exploration ability at the initial stages followed by more exploitation ability at the closing cycles. Hence, the modified PSO commences with a larger value of w, in order to promote the swarm with a better global search throughout the initial stages; thereafter, this parameter linearly decreases as a function of time (iteration) to avail a more in-depth local search by the particles (3.5).

$$w = w_{min} + (w_{max} - w_{min}) \left(\frac{t_{max} - t}{t_{max} - 1}\right) \tag{3.5}$$

where w_{max} and w_{min} denote upper and lower bounds of w, respectively; and t_{max} is the total number of iterations. Shi and Eberhart (1998) conclude the modified PSO with a range of $[0.9, 1.2]$ for the inertia weight outperforms the classical PSO algorithm discussed in section 3.1.

3.1.2. Discrete binary particle swarm optimization (D-PSO)

Acknowledging the importance of discrete optimization problems, the founders of classical PSO algorithm (Kennedy and Eberhart 1997), in 1997, introduced the binary version of this algorithm for problems with discrete objective functions which is slightly modified and adopted in the ensuing sections of this thesis to unravel discrete TCT problems. In the original paradigm of PSO, each particle's trajectory was defined as changes in position on some dimensions in a S-dimensional space; whereas, in the discrete version, trajectories and velocities are defined in a probabilistic space concerning state selection of the bits. The algorithm commences by randomly generating N particles with their corresponding velocity vectors. Then, based on a predefined fitness function, the swarm aims to uphold the probability of the binary variable that contributes to a better fit, through assigning due velocities. Kennedy and Eberhart (1997) define the migration of particles as flying in a state space with binary values of zero and one on any of the dimensions. Velocity on a single dimension is defined as the probability of change which is measured using the aforementioned Eq. (3.1), while satisfying the condition (3.3). Kennedy and Eberhart (1997), this time around, propose clamping the velocity for discrete PSO with $v_{max} = 6$, since, contrary to continuous version, smaller v_{max} allows for a larger range to be explored by the binary system. Moreover, the inertia weight parameter is not adopted for the velocity calculations, for, this version was developed prior to the modified paradigm of Shi and Eberhart (1998). Following the velocity measurements, they are margined to the range $[0,1]$ using the ensuing sigmoid function (3.6).

$$sig\left(v_{ij}^{(t)}\right) = \frac{1}{1 + \exp(-v_{ij}^{(t)})} \tag{3.6}$$

where each $v_{ij}^{(t)}$ represents the probability that the bit $x_{ij}^{(t)}$ would take the value 1, at time step t.

The component values of x_{iS}'s, including the pbest and gbest positions, are determined by selection of elements from the set $\{0,1\}$. However, x_{iS} does not hold a value unless it is evaluated based the probabilistic update equation. Accordingly, any bit with a certain velocity vector v_{iS} might possess diverse positions, x_{iS}, on a single dimension at every generation. The position update for ith particle on the jth dimension is measured subject to the following probabilistic condition (3.7).

$$x_{ij}^{(t+1)} = \begin{cases} 1 & if \quad sig\ (v_{ij}^{(t+1)}) > r_{ij} \\ 0 & \quad otherwise \end{cases} \tag{3.7}$$

where r_{ij} is a uniformly distributed random number in the range $[0, 1]$. Eq. (3.7) indicates that the value of x_{ij} will be kept 0, in case $sig\left(v_{ij}\right)$ equals to 0.

3.2. Siemens Approximation Method (SAM)

As it was discussed previously, in 1971, Siemens has developed a heuristic algorithm, hailed as the Siemens Approximation Method (SAM), for time-cost trade-off analyses (Siemens 1971). This method will be slightly modified and exploited in the ensuing sections of this thesis to provide a hybrid SAM-PSO algorithm for solving discrete time-cost curve problems. The original procedure is initiated with construction of the project network; thereafter, administering a couple of rules, activities that incur least additional costs are identified and crashed. The SAM algorithm comprises the stepwise procedure as follows:

I. Construction of the project network.

II. Identification of the paths in the network passing through the initial and final activities.

III. Evaluation of the cost slopes (cost per unit of time saved) for the activities in the network.

IV. Measurement of the completion time for the identified paths.

V. Determination of the longest path i.e. the critical path. If more than one choice exists, discrimination is made in favor of the path having smaller least cost slope.

VI. Detection of the activity with least cost slope within the selected critical path. If cost slope is common to more than one activity, discrimination is made in favor of the activity which is common to greater number of paths. If more than one choice still exists, discrimination is made in favor of activity that permits greater amount of expedition.

VII. Expedition of the detected activity by the available amount of duration.

VIII. Reiteration of steps III through VII until all the activities of the selected critical path is crashed.

The most prominent aspect of this method can be depicted as its assistance in crashing activities with lowest cost slopes, on progressively changing paths that have the longest duration, i.e., incessantly changing critical paths. However this heuristic algorithm might shorten the duration of a project beyond the intended amount, due to crashing activities merely with the minimum cost slope considerations, and regardless of the number of different paths the activities belong to. Besides, the cost slopes are evaluated merely with regard to the utmost tuples having shortest and longest durations. Thus, any crashing mode residing within the cheapest and the expensive modes will most probably be neglected in cost slope calculations. Acknowledging this drawback, a slight modification will be made prior to adoption of this model in the novel hybrid algorithm.

3.3. Initialization and Termination

It is obvious that any optimization algorithm involves fitness functions, suchlike, any algorithm requires a pattern to initialize the process, and another condition to terminate it. Initialization can be performed either by a deterministic or a random scheme; nonetheless, the latter condition is most commonly used with the population-based stochastic algorithms. Though, following types of initialization can be generally adopted for iterative algorithms (Parsopoulos and Vrahatis 2009):

- Deterministic initialization with constant seed: Suitable for the algorithm that repeatedly initialize from a certain point, determined by the users.

- Deterministic initialization with different seeds: Appropriate for the algorithm that initialize from a different point thru each cycle, selected by the user within the domain.

- Random initialization with constant seed: Applicable for the algorithm that repeatedly requires a certain randomly selected point to initialize.

- Random initialization with different seeds: Due for the algorithm that at each time step requires a different randomly generated point to initialize, selected from the domain.

The latter scheme is widely incorporated with the evolutionary strategies. Accordingly, in this thesis, in addition to a sole-PSO algorithm with random initial seeds, a hybrid PSO is also developed which embeds both the merits of the first scheme and the last pattern. Deterministic seeds developed by SAM portion of the algorithm are fed into the PSO, while, generating additional random populations.

Such as the initialization, there exist a couple of approaches toward termination of an algorithm. Termination is probably the most user-dependent part of the optimization process, and occurs when at least one of the user-defined conditions arise. The most typical termination conditions are as follows (Parsopoulos and Vrahatis 2009):

- Convergence in domain: The series of produced solutions converge to a minimizer.

- Convergence in function value: The function values of the solutions converge to a minimum.

- Computational budget limitations: The exhaustion of all the available computational resources.

- Search stagnation: The state of not being able to produce any new solutions. Such condition occurs when velocities in PSO incline to zero.

The latter condition is typically used as a termination criterion in the evolutionary algorithms. The progress rate of an algorithm reveals its efficiency as well as its potential for improving the attained results. In case of slow evolutions, search is said to be stagnated. Consequently, in this thesis, the algorithm halts if no improvement is observed for the best solution, within a certain number of successive generations (a fraction of the maximum number of iterations), or, if the maximum number of iterations is reached.

3.4. Particle Swarm Optimizer for Time-Cost Trade-Off Analyses

In this thesis, TCT problem has been provisioned under three different considerations. Initially, slight modifications are applied to the discrete version of the classical sole-PSO algorithm. Additionally, a time-constraint paradigm has been also developed. Afterwards, a novel hybrid algorithm has been introduced that embeds a slightly modified SAM method to an overhauled discrete PSO algorithm, extended for multi-objective optimizations. The hybrid SAM-PSO model aims at solving the time-cost curve problem in archiving the full Pareto front. In the ensuing sections, the developed algorithms, as wells as their flowcharts and pseudo-codes are elaborated.

3.4.1. Discrete TCTP

As it was discussed earlier, it is imperative for the TCT practices to consider discrete sets of time-cost options for the activities, whereof, discrete alternatives are commonplace in real-life projects and that any time-cost function can be estimated by means of discretization. Accordingly, an efficient procedure has been developed in this thesis to enhance the decision making process for the managers. The initial discrete sole-PSO algorithm is grounded on the version proposed by the founders (Kennedy and Eberhart 1997), which was demonstrated in section 3.1.2. However, slight modifications have been applied to the equations and an alternate position update equation has been adopted. The flowchart of the proposed discrete PSO algorithm is demonstrated in Figure 3.2, which is going to be detailed step by step in this section.

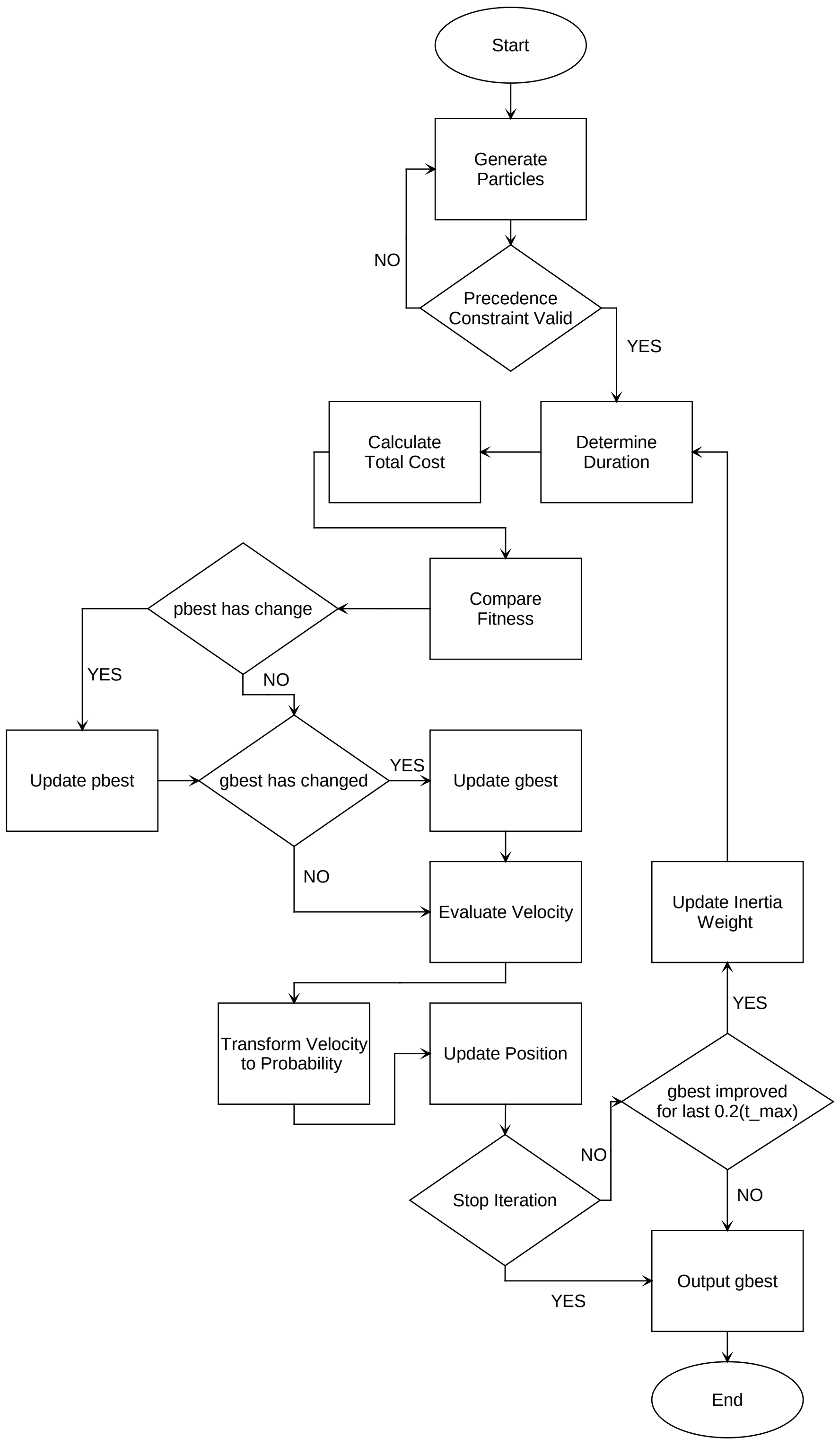

Figure 3.2 – Flowchart of the proposed discrete PSO algorithm.

Since the CPM calculations are executed within the framework of the proposed algorithm, the initial step requires definition of the project by the users, namely, direct cost (dc), duration (d), logical relationship, and realization alternatives of the activities, as well as the daily indirect cost (ic). Thereafter, the algorithm demands setting values for the operators like number of generations (t) and number of birds (N). Subsequent to determination of the preliminary information, the PSO algorithm commences as follows.

First of all, it must be noted that in this thesis, alternatives are presumed as the decision variables; yet, solutions are encoded in N number of $S \times 2m$ matrices, called position matrices, in which m is the maximum number of available modes (each mode comprising 2 columns, with odd columns dedicated for "duration" amounts, and even columns devoted for "direct costs" values) for S number of activities. The position of ith particle for the kth option of the jth activity, in the time step t is represented by $x_{ijk}^{(t)}$. Moreover, the algorithm involves binary discrete values, that is, each particle i, for its jth activity selects only a certain k that obtains integer 1 as its value, whereas, the rest of the x_{ijk}'s hold 0. In doing so, it is guaranteed that during each iteration, every particle chooses only a single option for any of the activities (3.8).

$$\sum_{k=1}^{m} x_{jk}^{(\mathrm{t})} = 1 \quad , \quad \forall\, j = \{1, \dots, S\} \tag{3.8}$$

Accordingly, incorporating the initialization techniques detailed in section 3.3, this model starts with positioning N number of randomly generated seeds, x_{ijk}'s, over the solution space, subject to the precedence constraints formulated as Eq. (3.9).

$$ES_j^{(t)} + d_j^{(t)} - ES_l^{(t)} \leq 0 \qquad , \qquad \forall\, l \in S_j \tag{3.9}$$

where d_j is activity j's duration; ES_j represents the early start time of jth activity; and S_j contains all the immediate successors of the jth activity. For the first activity of the network ($j = 1$), the early start time is assumed to be 1, i.e., $ES_1^{(t)} = 1$. Besides, non-negativity of the early start times and the durations are ensured satisfying the (3.10) condition.

$$ES_j^{(t)}, d_j^{(t)} \geq 0 \tag{3.10}$$

In addition to the random positions, each particle is treated with a random velocity vector through the first generation, $v_{ijk}^{(1)}$, which is clamped in accord with the predetermined v_{max}, as follows:

$$v_{ijk}^{(1)} = Random\ Number \quad , \quad v_{ijk}^{(1)} \in [-v_{max}, v_{max}] \tag{3.11}$$

Conception of the first generation finalizes with setting each particle's primary pbest, $P_i^{t_0}$, and gbest, $P_g^{t_0}$, positions as the current randomly generated location, using $S \times 2m$ matrices, as:

$$P_{ijk}^{t_0} = P_{gjk}^{t_0} = x_{ijk}^{(1)} \tag{3.12}$$

Concluding construction of the initial generation, each particle i's fitness is evaluated with respect to the objective function (3.13), which involves minimization of the total project cost in this case.

$$Minimize \sum_{\forall i} C_i \tag{3.13}$$

And the fitness evaluations are conducted by means of the fitness functions (3.14) and (3.15).

$$D_i = \max\left[\sum_{j=1}^{S}\sum_{k=1}^{m} d_{jk}^{(t)} x_{jk}^{(t)}\right] \tag{3.14}$$

$$C_i = \sum_{j=1}^{S}\sum_{k=1}^{m} dc_{jk}^{(t)} x_{jk}^{(t)} + D_i \times ic \tag{3.15}$$

$$\forall\, j = \{1, \dots, S\}, \quad \forall\, k = \{1, \dots, m\}$$

where D_i and C_i represent the total duration and the total cost of ith particle, respectively; d_{jk} represents duration of the kth options for the jth activity; dc_{jk} denotes direct cost for the kth alternative of the jth activity; and ic denotes the daily indirect cost.

Succeeding fitness evaluation of the particles, the optimality of the solutions are compared with each other regarding condition (3.16).

$$u > v \quad if \quad C_u \leq C_v \tag{3.16}$$

which determines discrimination is made in favor of decision vector u, in case the total cost of that particle is less than or equal to decision vector v; while, in case of equality, i.e., $C_u = C_v$, discrimination is made in favor of the particle having smaller duration (3.17).

$$u > v \quad if \quad \begin{cases} C_u = C_v \\ D_u < D_v \end{cases} \tag{3.17}$$

For the occasion that both particles u and v provide the same total costs and durations, discrimination is made randomly. Following identification of the better fitted individuals, P_i's and P_g's are updated accordingly. Meanwhile, for the first generation, P_{ijk} will remain identical to x_{ijk}.

Later, particles are flown to their new positions using the velocity vector formulated as Eq. (3.18) which are also encoded as $S \times 2m$ matrices. Albeit, notwithstanding the absence of the inertia weight (w) in the original velocity update equation of the discrete PSO (Kennedy and Eberhart 1997), this parameter is incorporated in this thesis. Hence, recruiting the aforementioned Eq. (3.5), a time-variant reduction for this parameter is directed to enhance the algorithm with a better balance between the local and global searches.

$$v_{ijk}^{(t+1)} = w^{(t)}\, v_{ijk}^{(t)} + c_1 r_1 \big(P_{ijk}^{(t)} - x_{ijk}^{(t)}\big) + c_2 r_2 \big(P_{gjk}^{(t)} - x_{ijk}^{(t)}\big) \tag{3.18}$$

Components of the Eq. (3.18) are identical to those discussed in sections 3.1 and 3.1.1; whilst, subscript k, denoting the numeral of the alternatives, has been supplemented to the operators. As mentioned in section 3.1, damping factor (3.3) is applied to the calculated velocities. Thence, measured velocities are transformed to probabilities and are margined to the range $[0,1]$, using a logistic transformation (3.19), as discussed earlier.

$$sig\big(v_{ijk}^{(t)}\big) = \frac{1}{1 + \exp(-v_{ijk}^{(t)})} \tag{3.19}$$

Each particle is then migrated to a new position subject to the probabilistic condition (3.20), adopted from Izakian et al. (2009, 2010).

$$x_{ijk}^{(t+1)} = \begin{cases} 1 & if \quad sig\ (v_{ijk}^{(t+1)}) = \max\Big\{sig\ (v_{ijk}^{(t+1)})\Big\} \\ 0 & \qquad otherwise \end{cases} \tag{3.20}$$

Eq. (3.20) differs from the position update equation proposed by Kennedy and Eberhart (1997), in that, for every activity, it involves determination of the alternative(s) associated with the maximum amount of probability; whereas, the original version takes on a uniformly distributed random number for evaluations. Eq. (3.20) indicates that in each row of position matrix, only a single alternative will obtain value 1, whose corresponding element in the velocity vector has the maximum probability. If $\max\Big\{sig\ (v_{ijk}^{(t+1)})\Big\}$ is common to more than one alternative, then, discrimination will be made randomly.

There forth, the process will reiterate by making comparison among particles using the fitness functions. This procedure will repeat until meeting the termination conditions, deliberated in section 3.3. Accordingly, the algorithm will halt if no improvement is monitored for the best particle, within $0.2 \times t_{max}$ successive generations, or, if the maximum number of iterations is reached. Ultimately, the algorithm will return the final gbest particle as the optimum or near-optimum solution for the discrete TCT problem. The pseudo-code of the proposed discrete PSO algorithm is illustrated in Figure 3.3.

```
Begin;
   For each $j = 1, \dots, S$;
      For each $k = 1, \dots m$;
         Retrieve first information;
         Create $S \times 2m$ matrices for $N$ particles;
      End;
   End;
   For each particle $i = 1, \dots, N$;
      Initialize an array with random positions and velocities on $S$ dimensions;
      If Eq. (3.9)=true;
         While $t \le t_{max}$ && gbest improved within last $0.2 \times t_{max}$;
            For each particle $i = 1, \dots, N$;
               Determine Duration using Eq. (3.14);
               Calculate Total Cost using Eq. (3.15);
               Initialize value of $w$;
               If $x_i > P_i$;
                  Set $x_{ijk}$ as pbest;
                     If $P_i > P_g$;
                        Set $P_i$ as gbest;
                     End;
               End;
               Calculate velocity using Eq. (3.18);
               Transform velocity to probability using Eq. (3.19);
               Update position using Eq. (3.20);
            End;
            Update value of $w$;
         End;
      End;
   End;
   Return gbest;
End;
```

Figure 3.3 – Pseudo-code of the proposed discrete PSO algorithm.

3.4.2. Time-constraint TCTP

In addition to the modified discrete PSO procedure, a slightly revised version of this algorithm is also developed for time-constraint TCT analyses, which is going to be presented in this section. The time-constraint TCT problems engage minimization of the total cost, taking into account an upper boundary for the completion time of projects. These types of TCT problems typically occupy contractual clauses concerning the daily liquidated damages for delays, and daily incentives for early completions. Accordingly, an efficient procedure has been developed in this thesis to assist decision makers with assessment of provisions that employ incentives and liquidated damages. On the verge of satisfying the resolution of time-constraint optimization, minor adjustments are applied to the algorithm detailed in section 3.4.1. These adjustments comprise implementation of three new parameters of Deadline, Penalty, and Bonus, as well as, introduction of a new fitness function to the system. The flowchart of the proposed PSO algorithm for time-constraint version of TCT problem is demonstrated in Figure 3.4, which is going to be explained methodically in this section.

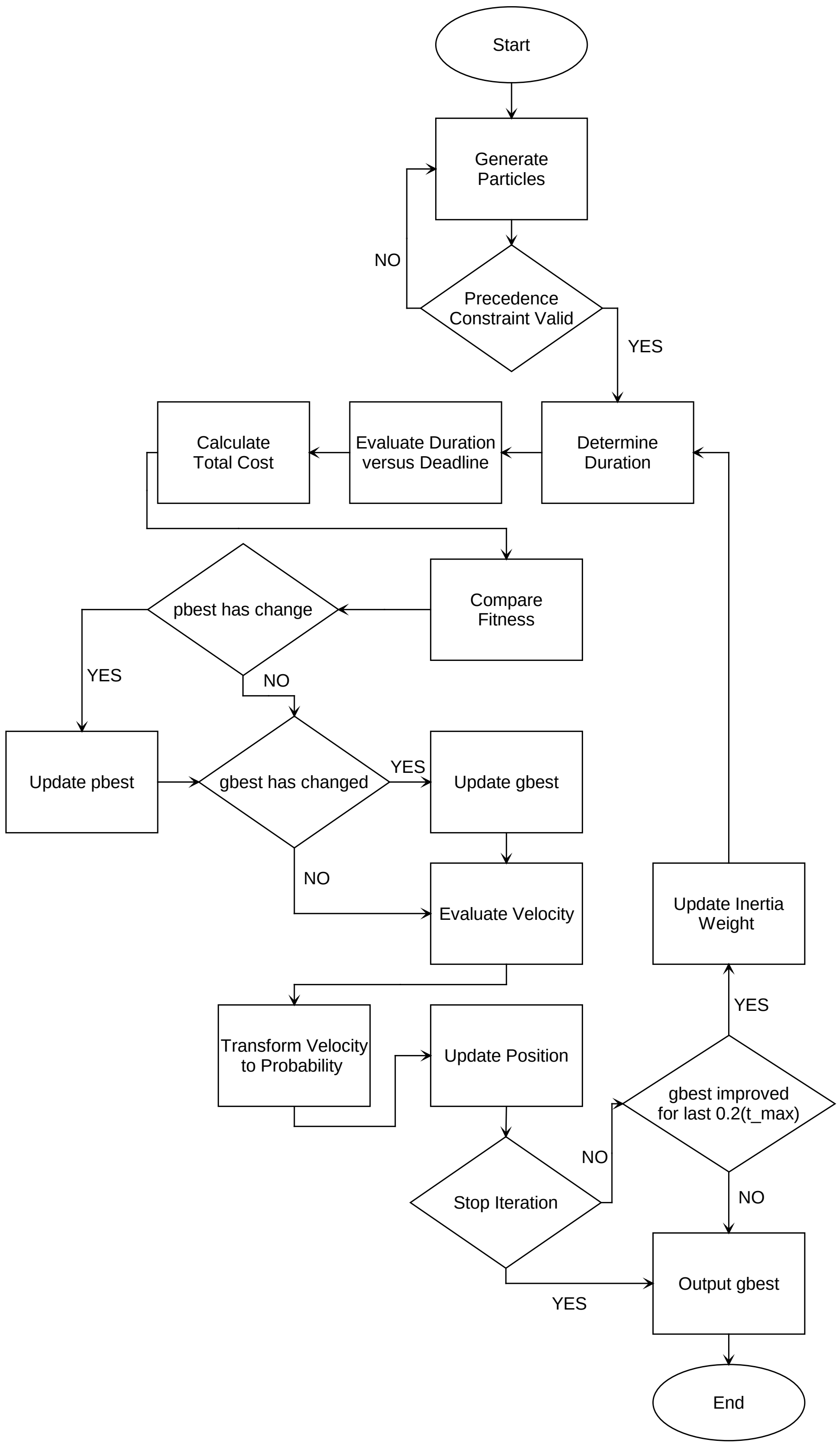

Figure 3.4 – Flowchart of the proposed PSO algorithm for time-constraint TCTP.

The proposed procedure remains identical to the version demonstrated in section 3.4.1, except for the retrieved initial information and the fitness function. In addition to the former definitions, this extension involves setting values for three new parameters by the users, namely, the amount of daily liquidated damage (pc), the daily incentive amount (bc), and the desired deadline (dd). Total cost representing the fitness of the particle is revised, such that, it ensures completion date of the optimal solution to be equal to or less than the specified deadline. Respectively, succeeding construction of the first generation, each particle i's fitness is evaluated with regard to the following conditions:

$$D_i = \max\left[\sum_{j=1}^{S}\sum_{k=1}^{m} d_{jk}^{(t)} x_{jk}^{(t)}\right] \tag{3.21}$$

$$T_i = D_i - dd \tag{3.22}$$

$$C_i = \begin{cases} \left(\sum_{j=1}^{S}\sum_{k=1}^{m} dc_{jk}^{(t)} x_{jk}^{(t)} + D_i \times ic\right) + (|T_i| \times pc) & if \quad T_i \geq 0 \\ \left(\sum_{j=1}^{S}\sum_{k=1}^{m} dc_{jk}^{(t)} x_{jk}^{(t)} + D_i \times ic\right) - (|T_i| \times bc) & otherwise \end{cases} \tag{3.23}$$

$$\forall\, j = \{1, \ldots, S\}, \quad \forall\, k = \{1, \ldots, m\}$$

Eq. (3.21) measures the critical path's duration; afterwards using Eq. (3.22), for each particle i, the time discrepancy between the desired deadline (dd) and the calculated duration is evaluated. This time discrepancy is then reflected on total cost calculations (3.23) as an extra term, provisioning penalties (pc) for delays and bonus payments (bc) for early completions. The sequel of this algorithm practices exactly the same procedure as, the system discussed in section 3.4.1. The pseudo-code of the proposed discrete PSO algorithm for time-constraint TCT problem is illustrated in Figure 3.5.

```
Begin;
    For each $j = 1, \dots, S$;
        For each $k = 1, \dots m$;
            Retrieve first information;
            Create $S \times 2m$ matrices for $N$ particles;
        End;
    End;
    For each particle $i = 1, \dots, N$;
        Initialize an array with random positions and velocities on $S$ dimensions;
        If Eq. (3.9)=true;
            While $t \leq t_{max}$ && gbest improved within last $0.2 \times t_{max}$;
                For each particle $i = 1, \dots, N$;
                    Determine Duration using Eq. (3.21);
                    Evaluate Duration vs. Deadline;
                    Calculate Total Cost using Eq. (3.23);
                    Initialize value of $w$;
                    If $x_i > P_i$;
                        Set $x_{ijk}$ as pbest;
                            If $P_i > P_g$;
                                Set $P_i$ as gbest;
                            End;
                    End;
                    Calculate velocity using Eq. (3.18);
                    Transform velocity to probability using Eq. (3.19);
                    Update position using Eq. (3.20);
                End;
                Update value of $w$;
            End;
        End;
    End;
    Return gbest;
End;
```

Figure 3.5 – Pseudo-code of the proposed PSO algorithm for the deadline problem.

3.4.3. Time-cost curve TCTP

It was declared earlier that the major concern with the TCT analyses is to unravel time-cost curve problem through obtaining a complete time-cost profile for the feasible project completion times. Originally conceded by Vilfredo Pareto, this profile is dubbed as the Pareto front or the efficient frontier, whose components are mutually non-dominated with respect to multiple criteria. As such, time-cost curve extension of the TCT problem is a multi-objective decision making problem, and any of its objectives might reach their optimal amounts at miscellaneous positions; thus, necessitating judgments of the experts for ultimate selection of the optimum solution along the efficient frontier. Obtaining the Pareto front for TCT problem, in essence, engages concurrent optimization of two classical TCT extensions, viz., the budget and the deadline problems. The results of these analyses are typically stored in a repository hailed as the external archive. Resultantly, for TCT problems, the Pareto front does not occupy vth solution if there is already another solution, u, in the archive, such that $D_v \geq D_u$ while $C_v \geq C_u$, and one of these inequalities holds strictly.

Owing to inherent complexity of the time-cost curve problem, it is imperative to develop a state-of-the-art model, capable of identifying the complete Pareto front for larger discrete TCT networks. To this end, two chief considerations are taken into account in this thesis. First, as stated in section 3.3, most of the evolutionary algorithms use a random scheme to initialize the first generation. Though, convergence capabilities of these algorithms extremely rely on the initial seed. Second, it is a challenge for the multi-objective optimization problems to compare the fitness of the archived solutions; for, solutions stored in the repository are mutually non-dominated and that there exist no general criterion for optimality. Thus, extending PSO for multi-objective problems urges practicing novel approaches toward evaluation of pbest and gbest positions of the particles.

Focusing on the two concerns mentioned, in this thesis, the complementary potencies of a heuristic method and a PSO are combined to develop a novel hybrid algorithm. Toward satisfying the first concern, minor modifications are applied to the original Siemens Approximation Method (SAM), and it is then embedded to a revamped PSO algorithm. The modified-SAM method accounts for a certain portion of the initial seed, with the remaining initial particles being generated randomly. Grounded upon the discrete PSO algorithm proposed in section 3.4.1, an overhauled system has been developed to address the second problem. Novel techniques for fitness evaluations are implemented into this paradigm of the PSO algorithm.

Correspondingly, this hybrid approach contrasts with the previous studies both in terms of the scheme used to generate the first population, and in terms of the objective function used to evaluate pbest and gbest of the particles. The escalated convergence competencies of this model, in mapping optimal costs to feasible durations, will be verified in the ensuing chapter. This section is dedicated to practice of the proposed hybrid SAM-PSO algorithm, whose flowchart is presented in Figure 3.6. Steps of this model are going to be illuminated in the sequel of this section.

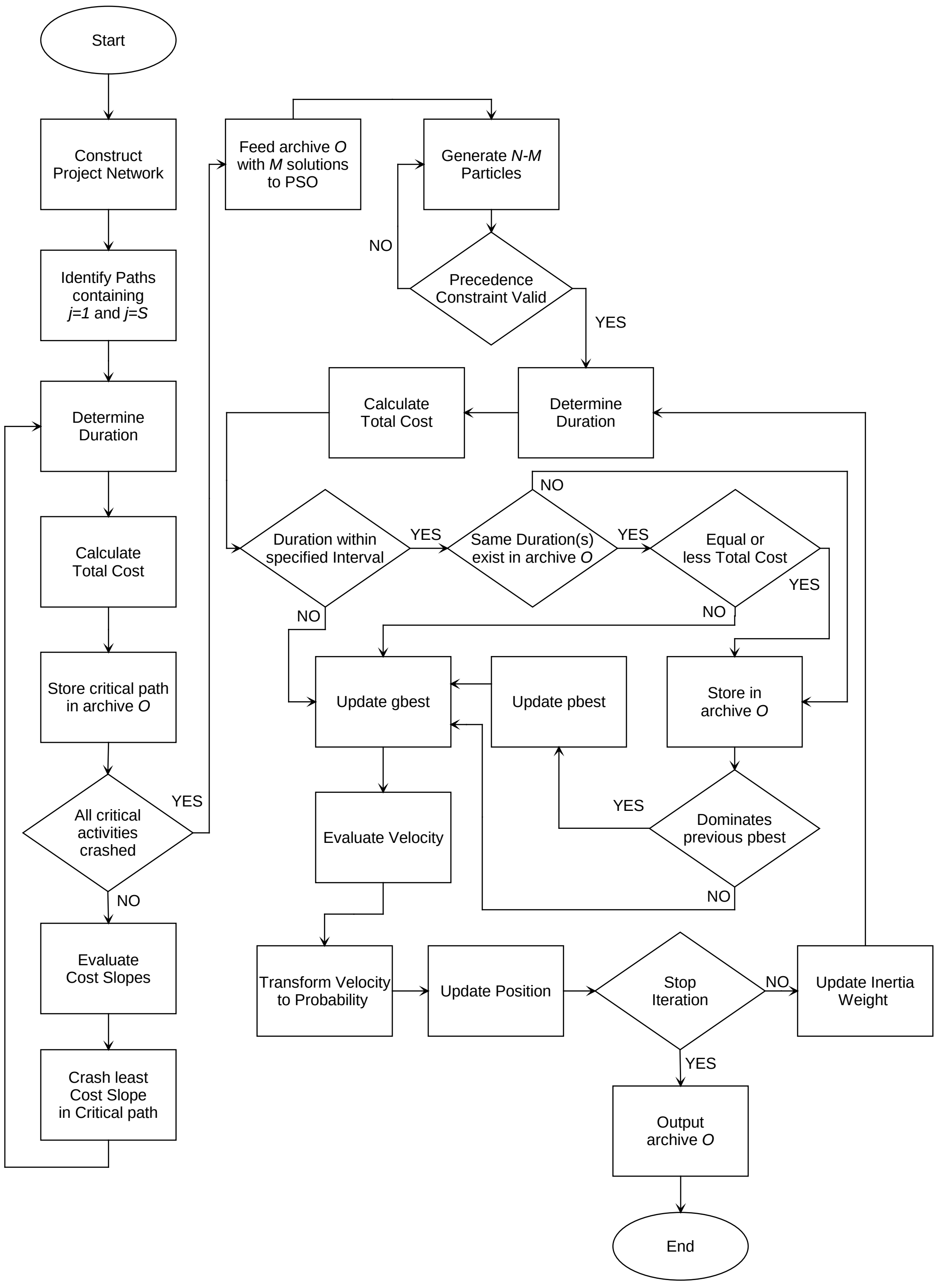
Start
Construct Project Network
Identify Paths containing *j=1* and *j=S*
Determine Duration
Calculate Total Cost
Store critical path in archive *O*
All critical activities crashed
YES
NO
Evaluate Cost Slopes
Crash least Cost Slope in Critical path
Feed archive *O* with *M* solutions to PSO
Generate *N-M* Particles
NO
Precedence Constraint Valid
YES
Calculate Total Cost
Determine Duration
Duration within specified Interval
YES
Same Duration(s) exist in archive *O*
NO
YES
Equal or less Total Cost
YES
NO
NO
Update gbest
Update pbest
Store in archive *O*
YES
Dominates previous pbest
NO
Evaluate Velocity
Transform Velocity to Probability
Update Position
Stop Iteration
NO
Update Inertia Weight
YES
Output archive *O*
End

Figure 3.6 – Flowchart of the proposed hybrid SAM-PSO algorithm.

Identical to the earlier proposed systems, the initial step of this model requires definition of the project by the users, namely, direct cost (dc), duration (d), logical relationship, and realization alternatives of the activities, as well as the daily indirect cost (ic). In addition to project information, the hybrid algorithm demands assigning a time interval (Z_{min} to Z_{max}) within which, the feasible realizations of durations will be explored. Afterwards, the algorithm invokes setting values for the operators like number of generations (t) and number of birds (N), with the latter being set meticulously (usually larger swarms) for the sake of deterministic seeds (discussed later in this section). Subsequent to determination of the preliminary information, the SAM-PSO algorithm embarks performing the following sequence of actions.

In lieu of the initialization technique adopted in the former models, SAM-PSO incorporates a semi-deterministic (semi-random) initialization scheme (readers are referred to section 3.3). To this end, a certain portion of the initial population is generated by dint of the modified-SAM method, with the remaining initial seeds being generated randomly. The modified-SAM method remains virtually unchanged compared to the original procedure (section 3.2), with a minor revision made to the cost slope evaluation pattern (step III). The reason behind this modification, as discussed in section 3.2 previously, is that the original SAM method evaluates cost slopes merely with regard to the utmost tuples having the shortest and the longest durations, which most probably causes neglecting any crashing mode residing within these alternatives. However, the modified-SAM assumes an incremental order for the tuples, from left to right, with respect to their costs; and unlike the original method, for each activity j, calculates the cost slopes (CS_j) involving the available utmost right and the penult crashing modes. During any iteration, this method uses Eq. (3.24) to evaluate the cost slopes.

$$CS_j = (C_{jk} - C_{j(k-1)})(D_{jk} - D_{j(k-1)})^{-1} \qquad (3.24)$$

$$\forall\, j = \{1, \dots, S\}, \quad \forall\, k = \{1, \dots, m\}$$

where the cost slopes of the first network are evaluated by setting $k = m$; afterwards, decreasing the numeral of option k, one at a time, as more alternatives of the jth activity gets crashed. The attained solutions from modified-SAM are represented by y_{ijk}, which implies solution i's position, for the kth option of the jth activity. An external repository, O, has been dedicated to the SAM-PSO model, so as to store all the non-dominated solution found by this algorithm. This external archive is designed to hold $\big((Z_{max} - Z_{min}) + 1\big)$ solutions, which are encoded in $S \times 2m$ matrices (m, allowable modes for S activities, comprising 2 column; odd and even columns devoted for “duration” and “direct costs”, respectively). Primarily, this hollow repository is exploited by the modified-SAM to archive all the time-cost realizations obtained thru each cycle. The number of solutions recorded in repository O, at the final cycle of this phase is denoted by M, allowing $\Big(\big((Z_{max} - Z_{min}) + 1\big) - M\Big)$ number of non-dominated particles to be added to the repository over the subsequent stages of SAM-PSO. Accordingly, as expressed previously, it is of great importance for this algorithm to set the number of particles, N, with a great obsession; in that, M number of particles will be occupied in this phase, leaving $N - M$ particles to be generated randomly amidst the PSO stage. Hence, the users are advised to take on larger swarms, such that:

$$N > M$$

The modified-SAM stage concludes with submission of the results stored in the archive O, to the particle swarm optimizer. In doing so, it is intended to feed the solutions archived in repository O, as initial seeds into the PSO model; as well, to exploit these solutions in pbest and gbest calculations of the PSO algorithm. It must be emphasized, however, that the external archive O is dedicated only to the non-dominated solutions and that the PSO process engages alternate matrices for the calculations. Therefore, succeeding the final cycle of the modified-SAM, solutions are seeded to the particle swarm optimizer; $N - M$ particles are then generated using a random scheme (identical to systems discussed in sections 3.4.1 and 3.4.2), initializing the PSO process. The aforementioned Eq. (3.9) is used to satisfy the precedence constraints of the generated particles. Thereafter, clamped to the feasible region $[-v_{max}, v_{max}]$, all the initial seeds, i.e., M deterministic and $N - M$ random particles are treated with random velocity vectors, $v_{ijk}^{(1)}$, through the first iteration.

Conception of the first generation concludes with determination of the primitive pbest, $P_i^{t_0}$, and gbest, $P_g^{t_0}$, positions. Encoded in $S \times 2m$ matrices, each particle acquires the "best" positions using Eq. (3.25) as follows:

$$\begin{cases} P_{ijk}^{t_0} = P_{gjk}^{t_0} = y_{ijk} & , \quad \forall i \in \{1, \dots, M\} \\ P_{ijk}^{t_0} = P_{gjk}^{t_0} = x_{ijk}^{(1)} & , \quad \forall i \in \{M+1, \dots, N\} \end{cases} \tag{3.25}$$

where each y_{ijk} is the position of the particle attained from the modified-SAM; and each $x_{ijk}^{(1)}$ represents position of the randomly generated particle.

Toward fitness evaluations, the rationale of the system discussed in section 3.4.1 has been totally revamped; in that, each particle i's fitness is evaluated with respect to the new objective function (3.26), which involves concurrent minimization of both the total duration and the total cost of the project.

$$Minimize\ y \equiv (D, C) \tag{3.26}$$

The fitness evaluations involve Eqs. (3.14) and (3.15) for total duration and total cost measurement. Herein, a controller is devised to carry out judgments regarding particles' qualification to enter the external archive O. For any decision vector x, this controller engages the following criteria with respect to the measured D_x and C_x:

$$Accept \quad if \quad \begin{cases} Z_{min} \leq D_x \leq Z_{max} \\ D_x \neq D_y \end{cases} \tag{3.27}$$

$$or \quad \begin{cases} Z_{min} \leq D_x \leq Z_{max} \\ D_x = D_y \\ C_x \leq C_y \end{cases} \tag{3.28}$$

$$Reject \qquad otherwise \tag{3.29}$$

where D_y and C_y respectively represent duration and cost of particle y, a solution in the archive O. Eq. (3.27) indicates any particle x, with a duration within the predetermined allowable interval, will be accepted if there is no decision vector y inside the archive with the same duration amount. Eq. (3.28) dictates any allowable solution x, with a duration amount identical to particle(s) y placed inside the repository, will be accepted if total cost of this solution is less than or equal to the archived particle(s). In this equation, a non-strict inequality is incorporated for cost comparison, so as to promote exploration. If particle x satisfies (3.28) condition, the individual y gets removed from the archive automatically. Moreover, Eq. (3.29) specifies no decision vector x will be accepted if none of the abovementioned conditions are met. At the end of the iterations, the solutions stored in the external repository form the non-dominated front have hitherto been solved.

Succeeding consecution of the archive members, SAM-PSO takes on a novel approach to measure pbets, P_i, and gbest, P_g, positions of the particles to calculate the velocity vectors. Selection of pbest is modified as to update should a strongly dominating solution emerges; that is, pbest will be updated only when the new position is non-dominated and it dominates all the preceding pbests. Former pbest, u, is said to dominate new postion, v, regarding condition (3.30).

$$u > v \quad if \quad \begin{cases} D_u < D_v \quad and \quad C_u \leq C_v \\ D_u \leq D_v \quad and \quad C_u < C_v \end{cases} \tag{3.30}$$

which determines discrimination is made in favor of decision vector u, in case its duration and cost are less than or equal to v, while one of these inequalities holds strictly. Meanwhile, for the first generation, P_i's will remain intact. Furthermore, the selection of gbest is revised to randomly select a non-dominated particle from the external repository, O, throughout each iteration. Such a concern is paramount for the model, since, all the archived solutions are non-dominated and are equally good.

Ultimately, particles are flown to their new positions following exactly the same procedures discussed in section 3.4.1. This process will reiterate until meeting the termination condition deliberated in section 3.3. Accordingly, the algorithm will halt if the maximum number of iterations is reached. Eventually, the algorithm will return the non-dominated solutions stored in the external archive, O, as the Pareto front of the time-cost curve problem. The pseudo-code of the proposed hybrid SAM-PSO algorithm is illustrated in Figure 3.7.

```
Begin;
    For each $j = 1, \dots, S$;
        For each $k = 1, \dots m$;
            Retrieve first information;
            Create $S \times 2m$ matrices for $(Z_{max} - Z_{min}) + 1$ particles;
            Create $S \times 2m$ matrices for $N$ particles;
        End;
    End;
    For each particle $i = 1, \dots, M$;
        Construct network;
        If path contains $j = 1$ && $j = S$;
            While all critical activities crashed != true;
                Determine Duration;
                Calculate Total Cost;
                Store critical path in archive $O$;
                Evaluate cost slopes using Eq. (3.24);
                Crash critical activity with least cost slope;
            End;
        End;
        Set $y_{ijk}$ as pbest and gbest using Eq. (3.25);
    End;
    For each particle $i = M + 1, \dots, N$;
        Initialize an array with random positions and velocities on $S$ dimensions;
        If Eq. (3.9)=true;
            Set $x_{ijk}$ as pbest and gbest using Eq. (3.25);
            While $t \leq t_{max}$;
                For each particle $i = 1, \dots, N$;
                    Determine Duration using Eq. (3.14);
                    Calculate Total Cost using Eq. (3.15);
                    Initialize value of $w$;
                    If Eq. (3.27) || Eq. (3.28) =true;
                        Add to archive $O$;
                        If $x_i$ is non-dominated && Eq. (3.30) =true;
                            Set $x_{ijk}$ as pbest;
                        End;
                        Select gbest randomly from archive $O$;
                    End;
                    Calculate velocity using Eq. (3.18);
                    Transform velocity to probability using Eq. (3.19);
                    Update position using Eq. (3.20);
                End;
                Update value of $w$;
            End;
        End;
    End;
    Return archive $O$;
End;
```

Figure 3.7 – Pseudo-code of the proposed hybrid SAM-PSO algorithm.

CHAPTER 4

VALIDATION AND EMPIRICAL ANALYSES

This chapter is devoted to validation and performance measurement of the developed algorithms. The proposed optimizers for discrete TCTP, as well as the time-constraint TCTP are validated using instances widely adopted in the literature. Empirical analyses are then conducted to measure performance and efficiency of the proposed model for solution of the time-cost curve problem. Throughout the computational experiments, the complete non-dominated fronts of test problems are introduced for the very first time in the literature. Experiments are also exerted to compare the obtained results using mixed integer programming technique.

4.1. Validating the Algorithms

In the course of development of the proposed algorithms, several test problems were employed to experiment their convergence capabilities. One of the chief prospects of these experimentations was to scrutinize the effect of selected parameters on the performance of the algorithms. As a result, the selected parameters were fine-tuned via a series of trial and error tests, respecting the convergence speed and quality of the solutions. The final operators set for each method, namely, iterations (t), particles (i), c_1, c_2, inertial weight (w), and v_{max} are given in the sequel of this chapter. The practiced test instances are some of the best known TCT problems analyzed in the construction management literature. Three extensions of discrete TCT problems are fed into the PSO optimizers, and experiments have been directed to validate their potencies accordingly.

Yet, as stated earlier, the optimality of the results cannot be confirmed unless an exact procedure is recruited. Respectively, it is not possible to accurately assess quality of the solutions obtained from heuristic or meta-heuristic algorithms short of identified optimal solutions. Accordingly, on the verge of performance evaluations, an exact procedure is also adopted within the context of this thesis. All the instances are solved to optimality by dint of mixed integer programming using the AIMMS 3.11 optimization software. Therewith, the obtained results are compared to solutions provided by the PSO models. The average percent deviations are then evaluated for multiple experimental runs. Moreover, the processing times required to unravel the test problems are also determined.

The first test problem involves the 18-activity network derived from Feng et al. (1997) incorporating the time-cost alternatives defined in Hegazy (1999). This instance is widely adopted by numerous researchers (Elbeltagi et al. 2005, Zheng et al. 2005, Elbeltagi et al. 2007, Ng and Zhang 2008, Xiong and

Kuang 2008, Afshar et al. 2009, Sonmez and Bettemir 2012) as a test-bed for performance evaluations. Four sample tests with different indirect cost provisions have been implemented to the discrete PSO algorithm introduced in section 3.4.1. This algorithm is also experimented for solution of a more complex problem to optimize time and cost concomitantly. Thus, the hypothetical 63-activity project derived from Sonmez and Bettemir (2012) is fitted into the model. Details of these instances along with the results of computational experiments are going to be presented in section 4.2.1. The achieved results ensure the robustness of the proposed algorithm compared to the solutions of well-developed algorithms, as well as the exact procedure.

For the time-constraint TCT analysis, a third test problem based upon the former 18-acitivty network is practiced. Described in Hegazy (1999), this instance incorporates liquidated damages and incentive payments with regard to a predetermined completion deadline. This instance has been implemented to the method discussed in section 3.4.2. The attained results further confirm efficiency of the proposed algorithm, providing sound solutions within a very small processing time. This test problem including the empirical analyses is going to be elaborated in section 4.2.2.

The hybrid SAM-PSO model is initially tested against the model developed by Afshar et al. (2009). To this end, the 18-activity TCT problem is adopted to unravel time-cost curve problem, assuming different values for the indirect cost. The results obtained from the modified-SAM method, and the final Pareto front are then compared against the optimal efficient frontier achieved by means of the exact procedure. Compared well against Afshar et al.'s (2009) model, application of SAM-PSO in solution of 63-activity problem derived from Sonmez and Bettemir (2012) is experimented. SAM-PSO's convergence capabilities in locating the Pareto front are demonstrated alongside the results of mixed integer programming. The results prove successful operation of the proposed algorithm by searching merely a small fraction of the search space, within an acceptable processing time. Details of the experimentations are given in section 4.2.3.

Throughout the validation process, ten successive test runs are executed for analysis of any of the instances. The average percent deviations from the optima, acquired by means of the exact procedure, are evaluated accordingly. The required processing times are also determined with regard to the CPU times taken to implement the instances. Throughout these implementations, no inflation, interest, or any second order cost component is reflected to the cost calculations. Besides, a 7-day workweek calendar is assumed to be available for the projects. Details of all the implemented TCT problems, selected parameter values, and the results of the empirical analyses are going to be presented in the ensuing section.

4.2. Empirical Analyses

All the algorithms proposed in sections 3.4.1, 3.4.2, and 3.4.3 have been coded in C++ programming language. Microsoft Visual Studio 2010 Ultimate Edition has been exploited to compile and debug the implemented algorithms.

All the experimentations have been carried out by a laptop computer running Windows 7 Ultimate Edition (64-bit) operating system, with Intel Core 2 Duo 3.06 GHz CPU, and 6 Gigabytes of Physical memory (RAM). The prepared programs engage CPM calculations for logical relationships of type finish-to-start, considering no lags in between. Processing times, presented in precision to centiseconds (cs), are measured concerning main blocks of code, regardless of the interval required to insert the inputs and the period taken to return the outputs. All the durations are reflected in "days", and the costs are measured in "USD ($)".

In the following sections, TCT problems and the experimentations are abstracted.

4.2.1. Discrete TCTP analyses

The validation and performance assessment of the algorithm introduced in section 3.4.1 is carried out implementing various cases of two test problems derived from the literature. The first examined TCT problem involves the 18-activity project derived from Feng et al. (1997) and Hegazy (1999). The logical relationships among the activities of this problem, together with the available time-cost alternative are given in Table 4.1. In this example, there is one activity with single mode, ten activities with three modes, two activities with four modes, and five activities with five modes; accounting for a total of 5.9×10^9 possible schedules. The activity on node (AoN) representation of this problem is illustrated in Figure 4.1. This problem has been examined under four different conditions regarding the amount of the indirect costs. The assumed indirect costs are as $200\$/day$, $500\$/day$, and $1{,}500\$/day$; while, in one of the situations the example is solved with no daily indirect cost considerations, i.e., $0\$/day$.

Table 4.1 – Data for the 18-activity TCT problem.

Act. No.	Pred.	Mode 1		Mode 2		Mode 3		Mode 4		Mode 5	
		Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $
1	–	14	2,400	15	2,150	16	1,900	21	1,500	24	1,200
2	–	15	3,000	18	2,400	20	1,800	23	1,500	25	1,000
3	–	15	4,500	22	4,000	33	3,200	–	–	–	–
4	–	12	45,000	16	35,000	20	30,000	–	–	–	–
5	1	22	20,000	24	17,500	28	15,000	30	10,000	–	–
6	1	14	40,000	18	32,000	24	18,000	–	–	–	–
7	5	9	30,000	15	24,000	18	22,000	–	–	–	–
8	6	14	220	15	215	16	200	21	208	24	120
9	6	15	300	18	240	20	180	23	150	25	100
10	2, 6	15	450	22	400	33	320	–	–	–	–
11	7, 8	12	450	16	350	20	300	–	–	–	–

Table 4.1 – Data for the 18-activity TCT problem. (*Continued*)

Act. No.	Pred.	Mode 1		Mode 2		Mode 3		Mode 4		Mode 5	
		Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $
12	5, 9, 10	22	2,000	24	1,750	28	1,500	30	1,000	–	–
13	3	14	4,000	18	3,200	24	1,800	–	–	–	–
14	4, 10	9	3,000	15	2,400	18	2,200	–	–	–	–
15	12	12	4,500	16	3,500	–	–	–	–	–	–
16	13, 14	20	3,000	22	2,000	24	1,750	28	1,500	30	1,000
17	11, 14, 15	14	4,000	18	3,200	24	1,800	–	–	–	–
18	16, 17	9	3,000	15	2,400	18	2,200	–	–	–	–

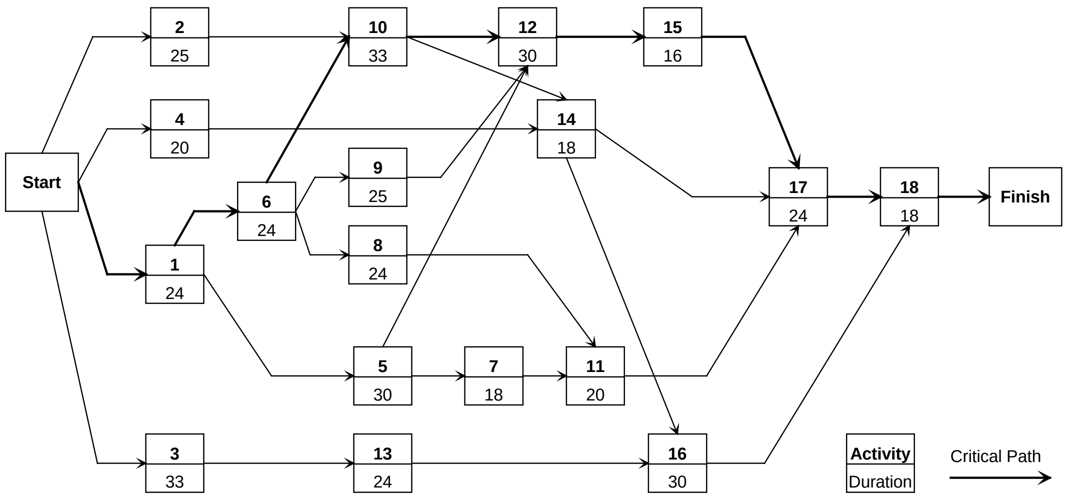

Figure 4.1 – Activity on node (AoN) representation of the 18-activity network.

The algorithm detailed in section 3.4.1 is adopted to exert TCT analyses for this problem. Since the convergence capabilities of the proposed algorithm are sensitive to the selected parameters, values are set for the operators through a series of trial experiments which are given in Table 4.2.

Table 4.2 – Parameters selected for discrete PSO algorithm.

Parameter	Value
t	40
i	70
c_1	2
c_2	2
w_{max}	1.2
w_{min}	0.4
v_{max}	6

As shown in Table 4.2, 40 generations with a swarm size of 70 are sufficient for the proposed algorithm to tackle the discrete TCT problem efficiently. Acquired from the literature, the cognition and social coefficients are set as integer 2, ascribing weighted average of 1 for the second and the third terms of Eq. (3.18). The inertia weight is set to linearly decrease through the execution of the algorithm, from the maximum value of 1.2 to the minimum value of 0.4. As depicted in section 3.1.2, the maximum allowed velocity is set as integer 6, contributing calculations of probabilities of range 0.0025 to 0.9975 via Eq. (3.19).

The results of this experiment are shown in Table 4.3, first column of which shows the amount of daily indirect cost; the second and the third columns demonstrate the best solution found by this algorithm. These results are compared to both the solutions provided by numerous researchers (Feng, Liu et al. 1997, Hegazy 1999, Elbeltagi, Hegazy et al. 2005, Zheng, Ng et al. 2005, Elbeltagi, Hegazy et al. 2007, Ng and Zhang 2008, Xiong and Kuang 2008, Sonmez and Bettemir 2012) and the optimal solutions. The optimal solutions are obtained using mixed integer model of Sonmez and Bettemir (2012) along with the AIMMS optimization software. The average percent deviations (APD) from the optima are then evaluated for ten consecutive experimental runs. Inasmuch as the algorithm located the global optima in any of the attempts, single solution for any values of the indirect cost are presented. Accordingly, APD's of zero amounts have been measured for the obtained solutions. In addition, the average CPU times taken to implement instances are also given in the last column of Table 4.3. The results prove that the proposed algorithm is capable of handling this instance effectively and efficiently, in that, finds global optimum solutions by searching merely a small fraction of the search space. In fact, only 2800 possible different schedules are explored thru each experiment. Searching only a small portion of the search space (4.74×10^{-5} %) allows this procedure to perform within an inconsiderable processing time of 0.08 seconds. As a result, the proposed algorithm outperforms all the earlier optimizers with regard to both the convergence speed and the quality of the solutions.

Table 4.3 – Results of experimental analyses for the proposed discrete PSO algorithm.

Number of Analyses	Indirect Cost	Duration	Cost	APD (%)	Average CPU time (s)
10	0	169	99,740	0.00	0.08
10	200	126	127,770	0.00	0.08
10	500	110	161,270	0.00	0.08
10	1,500	110	271,270	0.00	0.08

A second experiment employing a more complex problem is also conducted to measure the performance of the discrete PSO algorithm described in section 3.4.1. To accomplish this goal, the hypothetical 63-activity project described in Sonmez and Bettemir (2012) is fed into the model. Details of this instance, comprising the logical relationships of the activities, along with the available time-cost alternative are tabulated in Table 4.4. This instance contains two activities with three modes, fifteen activities with four modes, and forty-six activities with five modes; inducing a total of 1.37×10^{42} different possible project realizations. The activity on node (AoN) diagram of this problem is illuminated in Figure 4.2. This problem has been experimented under two different assumptions regarding the amount of the daily indirect costs. The presumed indirect costs are as $2{,}300\$/day$, and $3{,}500\$/day$.

Table 4.4 – Data for the 63-activity TCT problem.

Act. No.	Pred.	Mode 1		Mode 2		Mode 3		Mode 4		Mode 5	
		Dur. (days)	**Cost $**	**Dur. (days)**	**Cost $**	**Dur. (days)**	**Cost $**	**Dur. (days)**	**Cost $**	**Dur. (days)**	**Cost $**
1	-	14	3,750	12	4,250	10	5,400	9	6,250	-	-
2	-	21	11,250	18	14,800	17	16,200	15	19,650	-	-
3	-	24	22,450	22	24,900	19	27,950	17	31,650	-	-
4	-	19	17,800	17	19,400	15	21,600	-	-	-	-
5	-	28	31,180	26	34,200	23	38,250	21	41,400	-	-
6	1	44	54,260	42	58,450	38	63,225	35	68,150	-	-
7	1	39	47,600	36	50,750	33	54,800	30	59,750	-	-
8	2	52	62,140	47	69,700	44	72,600	39	81,750	-	-
9	3	63	72,750	59	79,450	55	86,250	51	91,500	49	99,500
10	4	57	66,500	53	70,250	50	75,800	46	80,750	41	86,450
11	5	63	83,100	59	89,450	55	97,800	50	104,250	45	112,400
12	6	68	75,500	62	82,000	58	87,500	53	91,800	49	96,550
13	7	40	34,250	37	38,500	33	43,950	31	48,750	-	-
14	8	33	52,750	30	58,450	27	63,400	25	66,250	-	-
15	9	47	38,140	40	41,500	35	47,650	32	54,100	-	-
16	9, 10	75	94,600	70	101,250	66	112,750	61	124,500	57	132,850
17	10	60	78,450	55	84,500	49	91,250	47	94,640	-	-
18	10, 11	81	127,150	73	143,250	66	154,600	61	161,900	-	-
19	11	36	82,500	34	94,800	30	101,700	-	-	-	-
20	12	41	48,350	37	53,250	34	59,450	32	66,800	-	-
21	13	64	85,250	60	92,600	57	99,800	53	107,500	49	113,750
22	14	58	74,250	53	79,100	50	86,700	47	91,500	42	97,400
23	15	43	66,450	41	69,800	37	75,800	33	81,400	30	88,450
24	16	66	72,500	62	78,500	58	83,700	53	89,350	49	96,400
25	17	54	66,650	50	70,100	47	74,800	43	79,500	40	86,800
26	18	84	93,500	79	102,500	73	111,250	68	119,750	62	128,500
27	20	67	78,500	60	86,450	57	89,100	56	91,500	53	94,750
28	21	66	85,000	63	89,750	60	92,500	58	96,800	54	100,500
29	22	76	92,700	71	98,500	67	104,600	64	109,900	60	115,600
30	23	34	27,500	32	29,800	29	31,750	27	33,800	26	36,200

Table 4.4 – Data for the 63-activity TCT problem. (*Continued*)

Act. No.	Pred.	Mode 1		Mode 2		Mode 3		Mode 4		Mode 5	
		Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $
31	19, 25	96	145,000	89	154,800	83	168,650	77	179,500	72	189,100
32	26	43	43,150	40	48,300	37	51,450	35	54,600	33	61,450
33	26	52	61,250	49	64,350	44	68,750	41	74,500	38	79,500
34	28, 30	74	89,250	71	93,800	66	99,750	62	105,100	57	114,250
35	24, 27, 29	138	183,000	126	201,500	115	238,000	103	283,750	98	297,500
36	24	54	47,500	49	50,750	42	56,800	38	62,750	33	68,250
37	31	34	22,500	32	24,100	29	26,750	27	29,800	24	31,600
38	32	51	61,250	47	65,800	44	71,250	41	76,500	38	80,400
39	33	67	81,150	61	87,600	57	92,100	52	97,450	49	102,800
40	34	41	45,250	39	48,400	36	51,200	33	54,700	31	58,200
41	35	37	17,500	31	21,200	27	26,850	23	32,300	-	-
42	36	44	36,400	41	39,750	38	42,800	32	48,300	30	50,250
43	36	75	66,800	69	71,200	63	76,400	59	81,300	54	86,200
44	37	82	102,750	76	109,500	70	127,000	66	136,800	63	146,000
45	39	59	84,750	55	91,400	51	101,300	47	126,500	43	142,750
46	39	66	94,250	63	99,500	59	108,250	55	118,500	50	136,000
47	40	54	73,500	51	78,500	47	83,600	44	88,700	41	93,400
48	42	41	36,750	39	39,800	37	43,800	34	48,500	31	53,950
49	38, 41, 44	173	267,500	159	289,700	147	312,000	138	352,500	121	397,750
50	45	101	47,800	74	61,300	63	76,800	49	91,500	-	-
51	46	83	84,600	77	93,650	72	98,500	65	104,600	61	113,200
52	47	31	23,150	28	27,600	26	29,800	24	32,750	21	35,200
53	43, 48	39	31,500	36	34,250	33	37,800	29	41,250	26	44,600
54	49	23	16,500	22	17,800	21	19,750	20	21,200	18	24,300
55	52, 53	29	23,400	27	25,250	26	26,900	24	29,400	22	32,500
56	50, 53	38	41,250	35	44,650	33	47,800	31	51,400	29	55,450
57	51, 54	41	37,800	38	41,250	35	45,600	32	49,750	30	53,400
58	52	24	12,500	22	13,600	20	15,250	18	16,800	16	19,450
59	55	27	34,600	24	37,500	22	41,250	19	46,750	17	50,750
60	56	31	28,500	29	30,500	27	33,250	25	38,000	21	43,800
61	56, 57	29	22,500	27	24,750	25	27,250	22	29,800	20	33,500
62	60	25	38,750	23	41,200	21	44,750	19	49,800	17	51,100
63	61	27	9,500	26	9,700	25	10,100	24	10,800	22	12,700

It has been widely documented in the literature that both the effectiveness and the efficiency of any meta-heuristic algorithm might be affected by the size of the problem; since, the solution space extents dramatically exposed to large-sized instances, urging excessive number of iterations. For such complex instances, setting larger swarm sizes with greater iterations provide greater chances in locating the global optimum; however, near optimum solutions demanding insignificant computational efforts are preferred for larger test problems. Accordingly, concluding several trial and error processes involving the 63-activity problem, as shown in Table 4.5, the discrete PSO algorithm is accommodated with tuned values for the operators.

Table 4.5 – Parameters selected for discrete PSO algorithm.

Parameter	Value
t	500
i	300
c_1	2
c_2	2
w_{max}	1.9
w_{min}	0.7
v_{max}	6

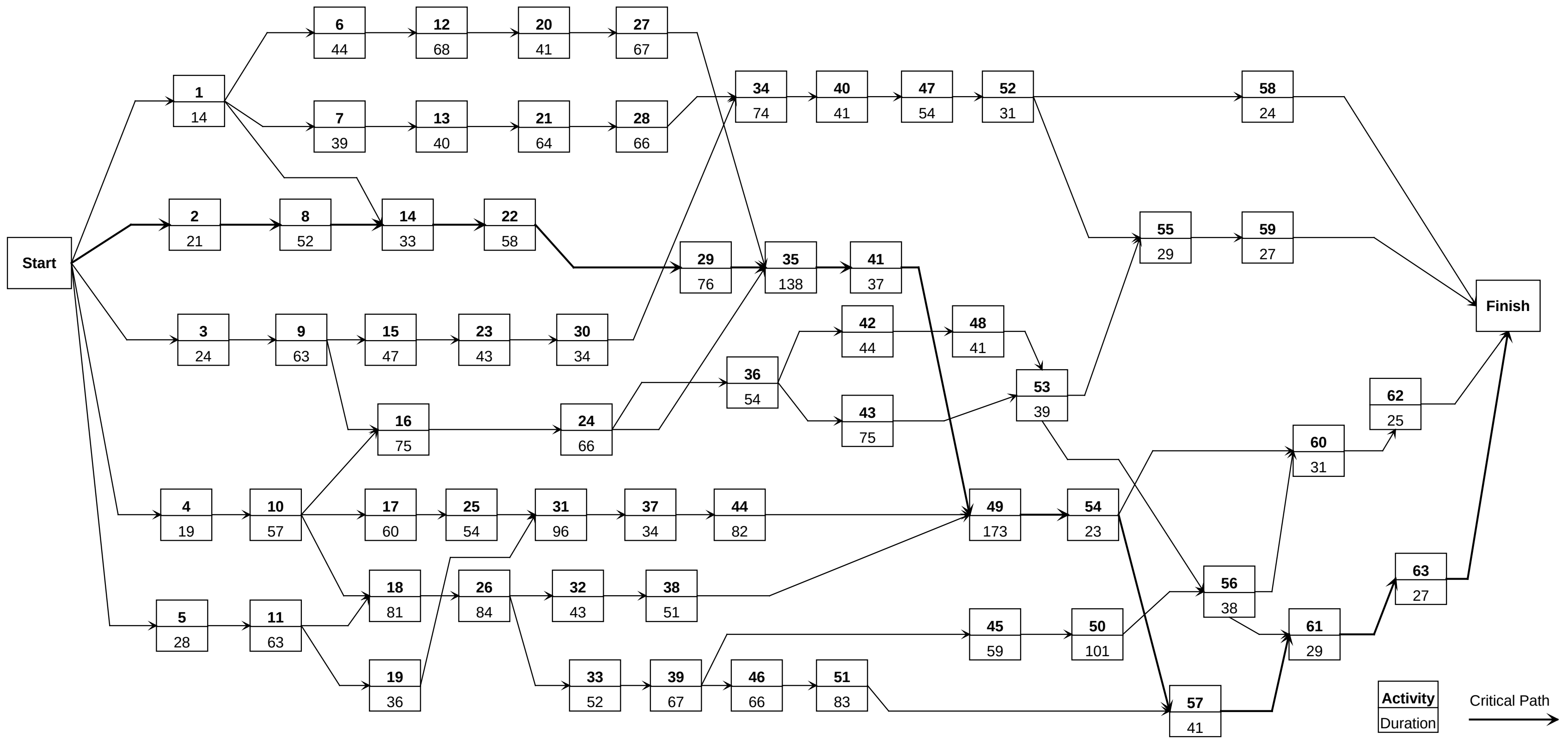

Figure 4.2 – Activity on node (AoN) representation of the 63-activity network.

500 iterations and a swarm size of 300 are experienced to provide sufficient convergence speed and quality for the proposed algorithm in solving the 63-activity TCT problem. The cognition and social coefficients remain unchanged with values set as integer 2, attributing weighted average of 1 for the second and the third terms of Eq. (1.18). For this instance, the inertia weight is set to linearly decrease through the iterations, from a larger maximum value of 1.9 to the minimum value of 0.7, for the sake of enhancing the process with greater explorations at the initial stages. However, the damping factor applied to the velocity calculations remains as integer 6, contributing calculations of probabilities of range 0.0025 to 0.9975 via Eq. (3.19).

The results of the experiments for the 63-activity project are abstracted in Table 4.6 and Table 4.7. The second and the third columns demonstrate durations and total costs of the best solutions found by this algorithm, respectively. These results are compared to both the solutions provided by Sonmez and Bettemir (2012), and the optimal solutions acquired from the mixed integer programming. Results of ten consecutive experimental runs are illustrated in Table 4.6 and Table 4.7, with corresponding percent deviations from the optima (PD) specified in the last columns of the aforementioned tables. The average percent deviations (APD) from the optima are then evaluated for each presumed values of daily indirect cost. In addition, last rows of these tables demonstrate the average processing times required to implement the instances. The results further validate robustness of the proposed algorithm, for, in any of the attempts it finds optimum or near-optimum solutions by searching simply a small portion of the solution space. Literally, mere 1.5×10^5 possible different combinations of the time-cost alternatives are explored throughout each experiment; that is, searching only a small fraction of the search space (1.09×10^{-35} %). This latter achievement allows the procedure to perform within an acceptable CPU time of 45 seconds.

Table 4.6 – Results of analyses for the 63-activity problem with daily indirect cost of 2,300$.

Analysis No.	Duration	Cost	PD (%)
1	630	5,421,120	0.00
2	630	5,422,420	0.02
3	630	5,421,120	0.00
4	630	5,421,120	0.00
5	633	5,421,320	0.00
6	636	5,422,970	0.03
7	631	5,424,420	0.06
8	633	5,421,320	0.00
9	633	5,421,320	0.00
10	629	5,423,270	0.04
		APD (%)	0.02
		Avg. CPU time (s)	45.00

Results summarized in Table 4.6 indicate that the proposed algorithm, concerning the values assigned for the generation and the swarm size (given in Table 4.5) was able to locate the global optimum solution during three attempts out of ten successive experiments; whereas, results for a higher daily indirect cost of 3,500$ reveals that the algorithm was able to find the global optimum only once over ten experiments. Nonetheless, setting larger iterations and/or swarms are discarded considering the closeness of the rest of the solutions to the global optimum; with the largest percent deviation being 0.03%. Ultimately, the proposed procedure proves to outperform all the earlier genetic algorithms discussed in Sonmez and Bettemir (2012), providing solutions with much less deviations from the optimum amounts.

Table 4.7 – Results of analyses for the 63-activity problem with daily indirect cost of 3,500$.

Analysis No.	Duration	Cost	PD (%)
1	616	6,177,820	0.03
2	626	6,177,370	0.02
3	621	6,176,220	0.00
4	621	6,178,020	0.03
5	629	6,177,270	0.02
6	621	6,177,120	0.02
7	621	6,176,170	0.00
8	618	6,177,570	0.02
9	618	6,177,670	0.02
10	618	6,177,570	0.02
		APD (%)	0.02
		Avg. CPU time (s)	45.00

4.2.2. Time-constraint TCTP analyses

Performance of the PSO model elucidated in section 3.4.2 is experimented implementing a time-constraint TCT problem derived from Hegazy (1999). The test problem exploited for the analyses is identical to the 18-activiy example described in section 4.2.1; though, it imposes an additional constraint by assuming a completion deadline for the project. Liquidated damages and incentive payments are incorporated into this problem. This instance has been examined under two different circumstances concerning the completion deadline. Supposed desired level of deadline, or, the maximum allowable duration for two analyses are set as 110 days and less than 110 days, respectively. Both the experiments assume an indirect cost of $200\$/day$, liquidated damages of $20{,}000\$/day$, and incentive payments of $1{,}000\$/day$.

For this specific example, appropriate values for the operators of the proposed PSO algorithm are identified following a sequence of trial and error. As presented in Table 4.8, the selected values for the parameters are analogous to the amounts identified for the experiments described in section 4.2.1.

Table 4.8 – Parameters of the discrete PSO algorithm for the deadline problem.

Parameter	Value
t	40
i	70
c_1	2
c_2	2
w_{max}	1.2
w_{min}	0.4
v_{max}	6

The solutions obtained from this experiment are demonstrated in Table 4.9. The third and the fourth columns of this table show durations and total costs of the global optima found by this algorithm, respectively. The acquired results are evaluated with regard to both the solutions provided Hegazy (1999) and the optimal solutions of AIMMS optimization software. The average percent deviations (APD) from the optima are measured for ten successive solutions provided by PSO optimizer for the deadline problem. Insomuch the algorithm located the global optima in any of the experiments, single solution for each of the assumed completion deadlines are tabulated. As a result, APD's of zero amounts have been measured for the obtained solutions. Moreover, the average CPU times taken to process the instances are illustrated in the last column of Table 4.9. These experiments verify the strengths of the proposed algorithm in dealing with such instances engaging exogenous constraints. Notwithstanding the extra calculations imposed to the algorithm, this procedure takes an inconsiderable processing time of 0.08 seconds, equal to the time required by the experimentations elaborated in section 4.2.1. As a result, the proposed algorithm outdoes the earlier optimizer with regard to the convergence speed; in addition, it bests nonsense solution of Hegazy (1999) for $Deadline < 110$ situation, where it provides a solution with duration of 107 days and total cost of 138,170$. The sound solutions provided by the PSO algorithm further affirm efficiency of the proposed optimizer.

Table 4.9 – Results of experimental analyses for time-constraint TCT problem.

Number of Analyses	Deadline	Duration	Cost	APD (%)	Average CPU time (s)
10	110	110	128,270	0.00	0.08
10	<110	104	136,120	0.00	0.08

4.2.3. Time-cost curve TCTP analyses

The validation and computational experiments of the hybrid SAM-PSO algorithm proposed in section 3.4.3 are carried out implementing various cases of the 18-activity and 63-activity problems, given in section 4.2.1

previously. For the primary analyses the 18-activity instance is occupied which was also used by Afshar et al. (2009) in a similar attempt for obtaining the Pareto front. However, the results of Afshar et al. (2009) fall short of delivering the full profile of the efficient frontier that comprises seventy solutions; rather, they report four, eighteen, and forty-four solutions located over the frontier for three different values of daily indirect cost. Furthermore, they do not report on the performance of their algorithm concerning multiple experiments; instead, results of only single experiments are documented. Therefrom, more comprehensive experiments involving multiple runs are directed within the context of this thesis.

Experimentations for the proposed hybrid algorithm have been initialed by evaluation of the durations associated with any feasible realization of the project. Thereafter, having identified the set of feasible durations, total costs will be calculated accordingly. Determination of the feasible durations is carried out by solving the instance to optimality, recruiting mixed integer programming technique. Accordingly, feasibility of the project for any amount of duration is tested using the AIMMS 3.11 optimization software, whereon, the optimal costs are evaluated. Tabulated below, the complete optimal Pareto fronts acquired by dint of this procedure for assumed indirect costs of $0\$/day$, $200\$/day$, and $1{,}500\$/day$ are being illustrated in Table 4.10, Table 4.11, and Table 4.12, respectively.

Table 4.10 – Optimal solutions for 18-activity problem over feasible set of durations (No indirect cost).

Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $
100	133,320	114	105,270	128	102,320	142	100,870	156	99,950
101	128,320	115	105,020	129	102,320	143	100,770	157	99,950
102	128,070	116	104,770	130	102,320	144	100,770	158	99,900
103	127,820	117	104,770	131	102,170	145	100,570	159	99,870
104	120,320	118	104,470	132	101,970	146	100,570	160	99,870
105	120,070	119	104,220	133	101,820	147	100,570	161	99,820
106	119,820	120	103,970	134	101,570	148	100,270	162	99,820
107	119,770	121	103,820	135	101,570	149	100,270	163	99,820
108	119,270	122	103,570	136	101,570	150	100,270	164	99,820
109	119,020	123	103,570	137	101,510	151	100,070	165	99,820
110	106,270	124	103,070	138	101,470	152	100,070	166	99,820
111	106,020	125	102,820	139	101,170	153	100,070	167	99,820
112	105,770	126	102,570	140	100,970	154	100,010	168	99,820
113	105,770	127	102,570	141	100,970	155	100,010	169	99,740

Table 4.11 – Optimal solutions for 18-activity problem over feasible set of durations (daily indirect cost of 200$).

Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $
100	153,320	114	128,070	128	127,920	142	129,270	156	131,150
101	148,520	115	128,020	129	128,120	143	129,370	157	131,350
102	148,470	116	127,970	130	128,320	144	129,570	158	131,500
103	148,420	117	128,170	131	128,370	145	129,570	159	131,670
104	141,120	118	128,070	132	128,370	146	129,770	160	131,870
105	141,070	119	128,020	133	128,420	147	129,970	161	132,020
106	141,020	120	127,970	134	128,370	148	129,870	162	132,220
107	141,170	121	128,020	135	128,570	149	130,070	163	132,420
108	140,870	122	127,970	136	128,770	150	130,270	164	132,620
109	140,820	123	128,170	137	128,910	151	130,270	165	132,820
110	128,270	124	127,870	138	129,070	152	130,470	166	133,020
111	128,220	125	127,820	139	128,970	153	130,670	167	133,220
112	128,170	126	127,770	140	128,970	154	130,810	168	133,420
113	128,370	127	127,970	141	129,170	155	131,010	169	133,540

Table 4.12 – Optimal solutions for 18-activity problem over feasible set of durations (daily indirect cost of 1,500$).

Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $
100	283,320	114	276,270	128	294,320	142	313,870	156	333,950
101	279,820	115	277,520	129	295,820	143	315,270	157	335,450
102	281,070	116	278,770	130	297,320	144	316,770	158	336,900
103	282,320	117	280,270	131	298,670	145	318,070	159	338,370
104	276,320	118	281,470	132	299,970	146	319,570	160	339,870
105	277,570	119	282,720	133	301,320	147	321,070	161	341,320
106	278,820	120	283,970	134	302,570	148	322,270	162	342,820
107	280,270	121	285,320	135	304,070	149	323,770	163	344,320
108	281,270	122	286,570	136	305,570	150	325,270	164	345,820
109	282,520	123	288,070	137	307,010	151	326,570	165	347,320
110	271,270	124	289,070	138	308,470	152	328,070	166	348,820
111	272,520	125	290,320	139	309,670	153	329,570	167	350,320
112	273,770	126	291,570	140	310,970	154	331,010	168	351,820
113	275,270	127	293,070	141	312,470	155	332,510	169	353,240

Table 4.10, Table 4.11, and Table 4.12 reveal 70 possible completion times for the 18-activity project. Therefore, having discussed in section 3.4.3, the external repository O, must be able to store 70 solutions as $18 \times 2(5)$ matrices

so as to be able to position one particle for any possible duration. Accordingly, for the sake of increasing the chances of obtaining a complete Pareto front thru any iteration, at least 70 particles must be generated. Respecting Table 4.13, Table 4.14, and Table 4.15, throughout the first phase of the hybrid algorithm, modified-SAM locates 24 particles over the solution space allowing for the rest of the population to be generated randomly.

Table 4.13 – Results provided by modified-SAM for 18-activity problem through the first phase of hybrid algorithm (No indirect cost).

Duration (days)	Cost $	Duration (days)	Cost $	Duration (days)	Cost $	Duration (days)	Cost $
100	133,420	114	105,270	128	102,970	154	100,010
101	128,320	116	105,020	134	101,570	156	99,950
101	128,420	120	104,770	140	100,970	158	99,900
104	120,320	122	104,270	145	100,570	159	99,870
105	120,270	123	104,020	148	100,270	161	99,820
110	106,270	124	103,770	151	100,070	169	99,740

Table 4.14 – Results provided by modified-SAM for 18-activity problem through the first phase of hybrid algorithm (daily indirect cost of 200$).

Duration (days)	Cost $	Duration (days)	Cost $	Duration (days)	Cost $	Duration (days)	Cost $
100	153,420	114	128,070	128	128,570	154	130,810
101	148,520	116	128,220	134	128,370	156	131,150
101	148,620	120	128,770	140	128,970	158	131,500
104	141,120	122	128,670	145	129,570	159	131,670
105	141,270	123	128,620	148	129,870	161	132,020
110	128,270	124	128,570	151	130,270	169	133,540

Table 4.15 – Results provided by modified-SAM for 18-activity problem through the first phase of hybrid algorithm (daily indirect cost of 1,500$).

Duration (days)	Cost $	Duration (days)	Cost $	Duration (days)	Cost $	Duration (days)	Cost $
100	283,420	114	276,270	128	294,970	154	331,010
101	279,820	116	279,020	134	302,570	156	333,950
101	279,920	120	284,770	140	310,970	158	336,900
104	276,320	122	287,270	145	318,070	159	338,370
105	277,770	123	288,520	148	322,270	161	341,320
110	271,270	124	289,770	151	326,570	169	353,240

Considering the total durations of the crashed (Z_{min}) and the all normal (Z_{max}) schedules, time intervals within which the feasible realizations of durations will be explored, and thereby the minimum population sizes have been identified. Consequently, performing numerous experiments with minimum population sizes of 70, the adequate values for the parameters have been dedicated as shown in Table 4.16.

Table 4.16 – Parameters of the SAM-PSO model for the 18-activity problem.

Parameter	Value
t	100
i	80
c_1	2
c_2	2
w_{max}	2.2
w_{min}	1.0
v_{max}	2

100 generations with a swarm size of 80 are experienced to suffice the convergence capabilities of the hybrid algorithm for the 18-activity problem. The cognition and social coefficients remain unchanged with values set as integer 2, attributing weighted average of 1 for the second and the third terms of Eq. (1.18). For this instance, the inertia weight is defined as a function of time to linearly decrease from a maximum value of 2.2 to a minimum amount of 1.0, through the execution of the PSO phase. Assignment of larger maximum and minimum values for the inertia weight facilitates broader explorations by the individuals through the entire execution of PSO phase. Moreover, the maximum allowed velocity is set as integer 2, contributing calculations of probabilities of range 0.12 to 0.88 via Eq. (3.19). In doing so, not only the unrestrained escalation of velocities that promote swarm divergence is eliminated, but also, the algorithm is accommodated with more exploration capabilities.

For any amount of daily indirect cost, ten successive experiments have been directed for solution of the time-cost curve problem. Henceforth, average percent deviations (APD) from the optima have been evaluated for any solution located along the time-cost profile. The ultimate Pareto fronts obtained by the hybrid algorithm, for three cases of the 18-activity problem are demonstrated in Table 4.17, Table 4.18, and Table 4.19. Associated to any identified duration, these tables encompass the least total cost observed over ten incessant trials. In any of these tables, in addition to APD's pertinent to each solution, the overall APD's are also evaluated, implying the performance of the proposed algorithm. It has been observed that the overall APD's of this model are slightly greater for the cases that assume smaller values for the indirect costs. Aside from that, the last rows of Table 4.17, Table 4.18, and Table 4.19 demonstrate the average processing times required to implement the instances; hereof, it takes an acceptable CPU time of approximately 8 seconds for the algorithm to unravel the 18-activity instance. These results validate effectiveness and efficiency of the hybrid model in locating the complete Pareto front for the 18-activity problem, thru any of the experimented cases. This model outperforming Afshar et al.'s

(2009) procedure, searches a mere 1.35×10^{-4} fraction of the solution space to provide the whole non-dominated front with inconsiderable deviations from the optimal solutions.

Table 4.17 – Complete Pareto front of 18-activity problem obtained by SAM-PSO model (No indirect cost).

Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %
100	133,320	0.00	114	105,270	0.00	128	102,320	0.00	142	100,870	0.00	156	99,950	0.00
101	128,320	0.00	115	105,020	0.00	129	102,510	0.19	143	100,770	0.00	157	100,370	0.42
102	128,070	0.00	116	104,770	0.00	130	102,470	0.15	144	101,050	0.28	158	99,900	0.00
103	127,820	0.00	117	104,770	0.00	131	102,170	0.00	145	100,570	0.00	159	99,870	0.00
104	120,320	0.00	118	104,470	0.00	132	101,970	0.00	146	100,710	0.14	160	100,540	0.67
105	120,070	0.00	119	104,220	0.00	133	101,820	0.00	147	100,750	0.18	161	99,820	0.00
106	119,820	0.00	120	103,970	0.00	134	101,570	0.00	148	100,270	0.00	162	100,940	1.12
107	119,770	0.00	121	103,820	0.00	135	101,770	0.20	149	100,570	0.30	163	100,240	0.42
108	119,270	0.00	122	103,570	0.00	136	101,620	0.05	150	100,450	0.18	164	100,440	0.62
109	119,020	0.00	123	103,570	0.00	137	101,510	0.00	151	100,070	0.00	165	100,740	0.92
110	106,270	0.00	124	103,070	0.00	138	101,470	0.00	152	100,400	0.33	166	99,940	0.12
111	106,020	0.00	125	102,820	0.00	139	101,170	0.00	153	100,150	0.08	167	100,240	0.42
112	105,770	0.00	126	102,570	0.00	140	100,970	0.00	154	100,010	0.00	168	na	na
113	105,770	0.00	127	102,570	0.00	141	101,270	0.30	155	100,100	0.09	169	99,740	0.00
												Overall APD (%)		0.10
												Avg. CPU time (s)		8.18

Table 4.18 – Complete Pareto front of 18-activity problem obtained by SAM-PSO model (daily indirect cost of 200$).

Dur.	Cost	APD	Dur.	Cost	APD	Dur.	Cost	APD	Dur.	Cost	APD	Dur.	Cost	APD
(days)	$	%	(days)	$	%	(days)	$	%	(days)	$	%	(days)	$	%
100	153,320	0.00	114	128,070	0.00	128	127,920	0.00	142	129,270	0.00	156	131,150	0.00
101	148,520	0.00	115	128,020	0.00	129	128,310	0.15	143	129,370	0.00	157	131,770	0.32
102	148,470	0.00	116	127,970	0.00	130	128,470	0.12	144	129,850	0.22	158	131,500	0.00
103	148,420	0.00	117	128,170	0.00	131	128,370	0.00	145	129,570	0.00	159	131,670	0.00
104	141,120	0.00	118	128,070	0.00	132	128,370	0.00	146	129,910	0.11	160	132,540	0.51
105	141,070	0.00	119	128,020	0.00	133	128,420	0.00	147	130,150	0.14	161	132,020	0.00
106	141,020	0.00	120	127,970	0.00	134	128,370	0.00	148	129,870	0.00	162	133,340	0.85
107	141,170	0.00	121	128,020	0.00	135	128,770	0.16	149	130,370	0.23	163	132,840	0.32
108	140,870	0.00	122	127,970	0.00	136	128,820	0.04	150	130,450	0.14	164	133,240	0.47
109	140,820	0.00	123	128,170	0.00	137	128,910	0.00	151	130,270	0.00	165	133,740	0.69
110	128,270	0.00	124	127,870	0.00	138	129,070	0.00	152	130,800	0.25	166	133,140	0.09
111	128,220	0.00	125	127,820	0.00	139	128,970	0.00	153	130,750	0.06	167	133,640	0.32
112	128,170	0.00	126	127,770	0.00	140	128,970	0.00	154	130,810	0.00	168	na	na
113	128,370	0.00	127	127,970	0.00	141	129,470	0.23	155	131,100	0.07	169	133,540	0.00
												Overall APD (%)		0.08
												Avg. CPU time (s)		7.98

Table 4.19 – Complete Pareto front of 18-activity problem obtained by SAM-PSO model (daily indirect cost of 1,500$).

Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %
100	283,320	0.00	114	276,270	0.00	128	294,320	0.00	142	313,870	0.00	156	333,950	0.00
101	279,820	0.00	115	277,520	0.00	129	296,010	0.06	143	315,270	0.00	157	335,870	0.13
102	281,070	0.00	116	278,770	0.00	130	297,470	0.05	144	317,050	0.09	158	336,900	0.00
103	282,320	0.00	117	280,270	0.00	131	298,670	0.00	145	318,070	0.00	159	338,370	0.00
104	276,320	0.00	118	281,470	0.00	132	299,970	0.00	146	319,710	0.04	160	340,540	0.20
105	277,570	0.00	119	282,720	0.00	133	301,320	0.00	147	321,250	0.06	161	341,320	0.00
106	278,820	0.00	120	283,970	0.00	134	302,570	0.00	148	322,270	0.00	162	343,940	0.33
107	280,270	0.00	121	285,320	0.00	135	304,270	0.07	149	324,070	0.09	163	344,740	0.12
108	281,270	0.00	122	286,570	0.00	136	305,620	0.02	150	325,450	0.06	164	346,440	0.18
109	282,520	0.00	123	288,070	0.00	137	307,010	0.00	151	326,570	0.00	165	348,240	0.26
110	271,270	0.00	124	289,070	0.00	138	308,470	0.00	152	328,400	0.10	166	348,940	0.03
111	272,520	0.00	125	290,320	0.00	139	309,670	0.00	153	329,650	0.02	167	350,740	0.12
112	273,770	0.00	126	291,570	0.00	140	310,970	0.00	154	331,010	0.00	168	na	na
113	275,270	0.00	127	293,070	0.00	141	312,770	0.10	155	332,600	0.03	169	353,240	0.00
												Overall APD (%)		0.03
												Avg. CPU time (s)		7.97

The solutions obtained from mixed integer programming, modified-SAM, and SAM-PSO procedures are plotted against each other in the ensuing figures. Figure 4.3, Figure 4.4, and Figure 4.5 illuminate the solutions obtained via SAM-PSO model for the 18-activity problem with indirect costs of $0\$/day$, $200\$/day$, and $1500\$/day$, respectively. For any of the cases, it can be easily observed that the 24 initial solutions seeded by modified-SAM are virtually lying over the final efficient frontier. Solutions located through this phase provide the second phase with seeds of higher quality. To some extent, these solutions give head-start to the PSO stage of the hybrid algorithm. Eventually, the best non-dominated solutions found over ten incessant experiments constitute the final time-cost profile, which practically lies over the optimal frontier solved by the AIMMS software. Contrary to the mixed integer programming technique, the proposed hybrid algorithm generates solutions only with regard to different combinations of time-cost alternatives and without incorporating lag times between finish-to-start relationships of the activities. Owing to this approach, the hybrid algorithm maps minimum total costs to 69 feasible realizations of project duration, whereas, the mixed integer programming technique benefitting from lag times, locates 70 solutions. In other words, for the 18-activity problem, there exist no certain combination of time-cost alternatives that results in 168 days of duration, that is, it is only feasible by adding lag times between the activities.

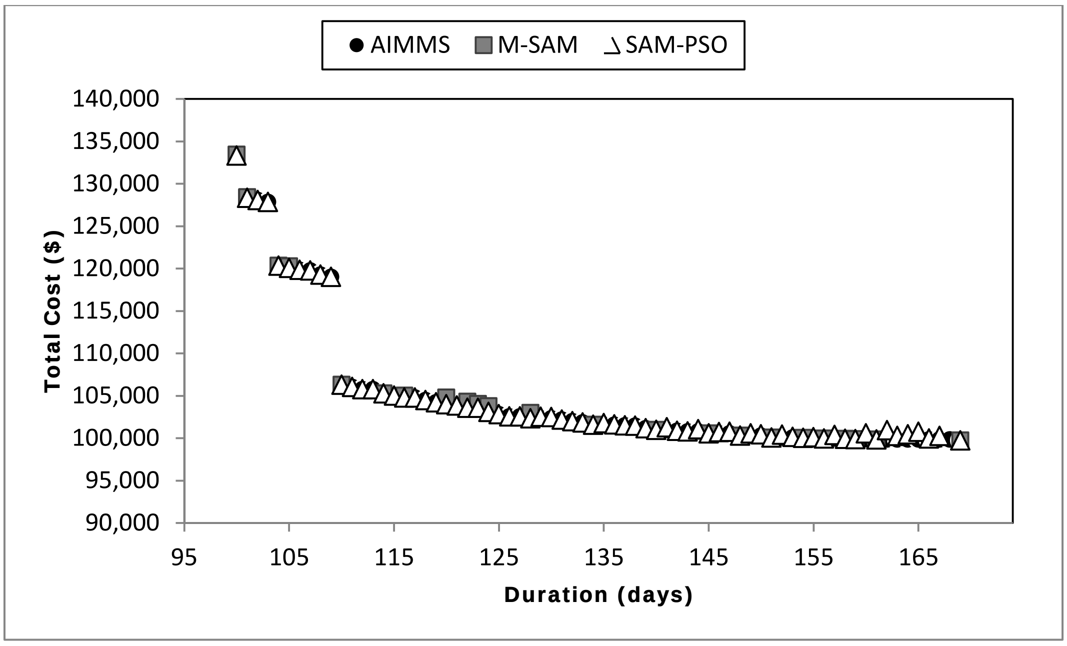

Figure 4.3 – Comparison of obtained Pareto fronts for 18-activity problem (No indirect cost).

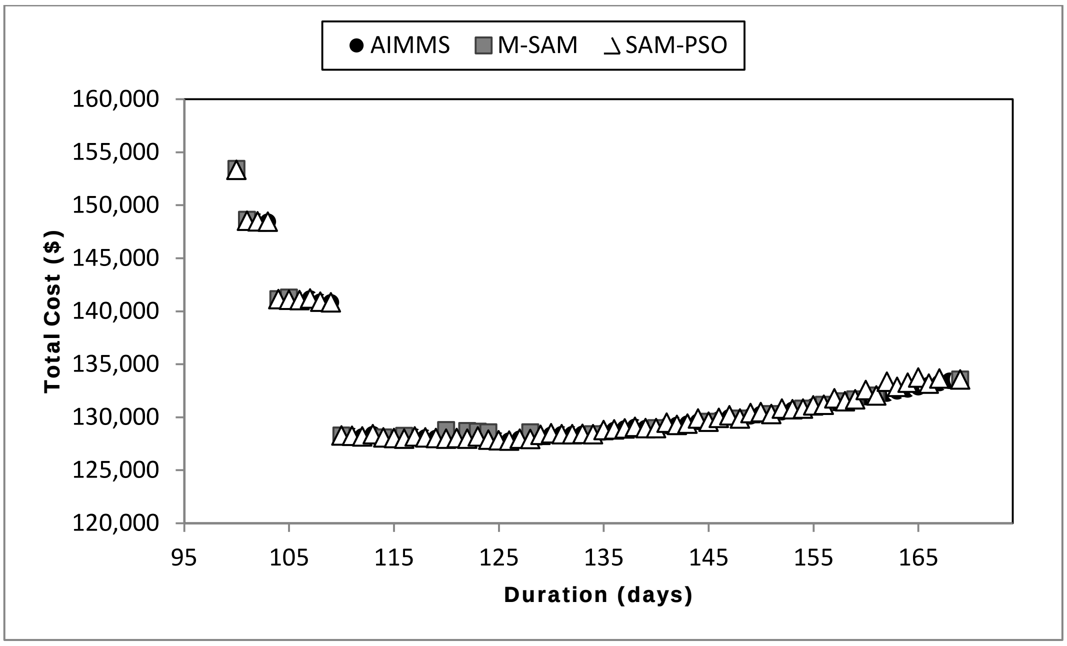

Figure 4.4 – Comparison of obtained Pareto fronts for 18-activity problem (daily indirect cost of 200$).

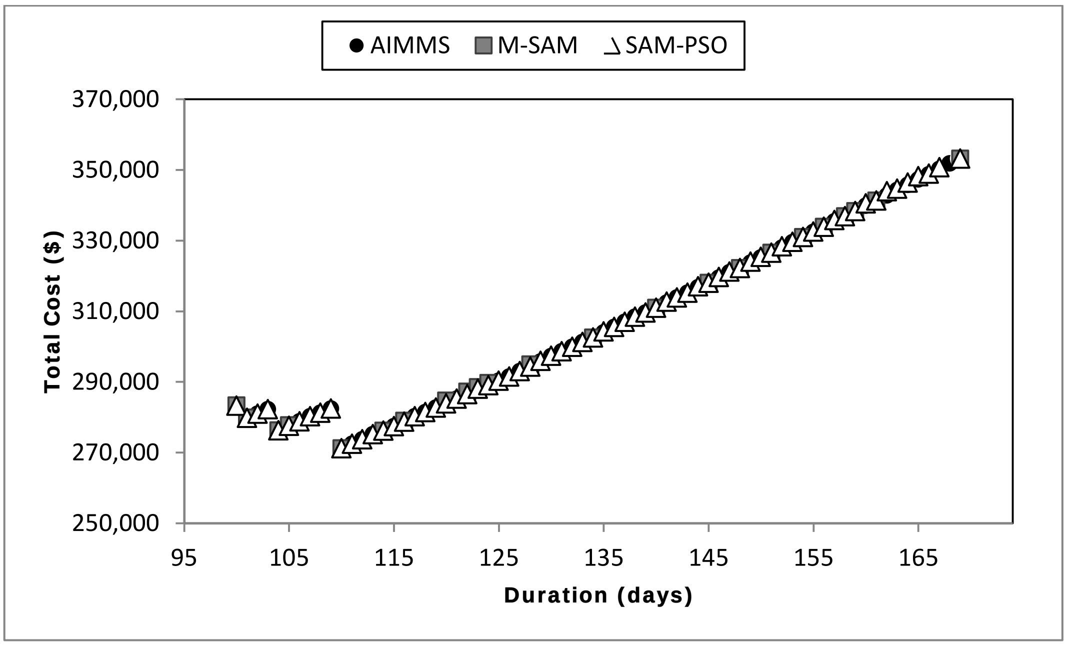

Figure 4.5 – Comparison of obtained Pareto fronts for 18-activity problem (daily indirect cost of 1,500$).

Concluding analyses of the 18-activity TCT problem, computational experiments of the proposed SAM-PSO model engaging a more complex problem have also been conducted; as a result, the hypothetical 63-activity project detailed in section 4.2.1 is fitted into the model. Experimentations have been initiated by determining durations associated with any feasible realization of the project. The feasible set of non-dominated solutions with corresponding amounts of duration and cost are identified by dint of the mixed integer programming technique derived from Sonmez and Bettemir (2012). Pareto fronts achieved through this technique have been tabulated in Table 4.20 and Table 4.21, involving two assumptions regarding the amount of daily indirect cost; the first case incorporates indirect cost of $2{,}300\$/day$, while the second case considers $3{,}500\$/day$.

Table 4.20 – Optimal solutions for 63-activity problem over feasible set of durations (daily indirect cost of 2,300$).

Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $
513	5,864,180	541	5,677,990	569	5,549,700	597	5,468,920	625	5,427,920	653	5,428,770	681	5,448,720
514	5,861,380	542	5,669,650	570	5,547,450	598	5,467,370	626	5,426,170	654	5,429,470	682	5,449,720
515	5,845,840	543	5,664,040	571	5,541,450	599	5,466,020	627	5,425,770	655	5,429,770	683	5,451,420
516	5,839,680	544	5,661,740	572	5,535,390	600	5,464,520	628	5,423,520	656	5,430,970	684	5,452,170
517	5,832,630	545	5,653,650	573	5,533,330	601	5,465,220	629	5,422,470	657	5,430,720	685	5,453,170
518	5,822,815	546	5,645,640	574	5,530,430	602	5,460,970	630	5,421,120	658	5,432,120	686	5,454,870
519	5,811,205	547	5,642,490	575	5,524,770	603	5,461,920	631	5,422,370	659	5,432,220	687	5,455,870
520	5,804,630	548	5,637,350	576	5,522,420	604	5,460,570	632	5,421,820	660	5,433,070	688	5,457,470
521	5,797,465	549	5,636,200	577	5,519,720	605	5,461,820	633	5,421,320	661	5,433,420	689	5,458,520
522	5,792,065	550	5,627,900	578	5,516,420	606	5,460,620	634	5,421,420	662	5,434,420	690	5,459,620
523	5,782,505	551	5,623,890	579	5,512,370	607	5,458,420	635	5,422,220	663	5,435,820	691	5,460,970
524	5,779,190	552	5,620,850	580	5,511,020	608	5,456,630	636	5,422,320	664	5,436,170	692	5,461,970
525	5,770,100	553	5,616,490	581	5,508,530	609	5,453,680	637	5,421,620	665	5,437,170	693	5,463,670
526	5,760,405	554	5,612,000	582	5,503,030	610	5,452,130	638	5,422,320	666	5,438,370	694	5,464,670
527	5,756,740	555	5,604,750	583	5,498,980	611	5,451,280	639	5,422,520	667	5,438,720	695	5,466,270
528	5,747,900	556	5,603,000	584	5,496,480	612	5,445,870	640	5,423,220	668	5,439,470	696	5,467,620
529	5,740,040	557	5,600,850	585	5,494,580	613	5,445,270	641	5,423,470	669	5,439,770	697	5,468,620
530	5,739,090	558	5,593,530	586	5,487,770	614	5,443,270	642	5,424,170	670	5,440,970	698	5,470,320
531	5,729,750	559	5,591,640	587	5,487,140	615	5,440,820	643	5,424,470	671	5,440,720	699	5,471,320
532	5,726,650	560	5,586,840	588	5,483,170	616	5,438,020	644	5,425,670	672	5,442,120	700	5,472,920
533	5,719,250	561	5,583,440	589	5,479,670	617	5,436,620	645	5,425,420	673	5,442,220	701	5,474,820
534	5,714,340	562	5,577,590	590	5,476,520	618	5,435,020	646	5,426,720	674	5,443,070	702	5,476,920
535	5,709,040	563	5,575,350	591	5,475,020	619	5,434,020	647	5,426,920	675	5,443,420	703	5,478,720
536	5,702,900	564	5,570,450	592	5,473,070	620	5,434,270	648	5,427,620	676	5,444,420	704	5,480,420
537	5,696,040	565	5,566,640	593	5,473,470	621	5,430,970	649	5,426,920	677	5,445,820	705	5,481,420
538	5,693,450	566	5,565,200	594	5,471,720	622	5,430,820	650	5,427,620	678	5,446,170	706	5,483,020
539	5,688,850	567	5,555,900	595	5,470,420	623	5,430,820	651	5,427,820	679	5,447,170	707	5,484,920
540	5,679,740	568	5,552,550	596	5,469,220	624	5,428,570	652	5,428,520	680	5,448,370	708	5,487,020

Table 4.21 – Optimal solutions for 63-activity problem over feasible set of durations (daily indirect cost of 3,500$).

Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $
513	6,479,780	541	6,327,190	569	6,232,500	597	6,185,320	625	6,177,920	653	6,212,370	681	6,265,920
514	6,478,180	542	6,320,050	570	6,231,450	598	6,184,970	626	6,177,370	654	6,214,270	682	6,268,120
515	6,463,840	543	6,315,640	571	6,226,650	599	6,184,820	627	6,178,170	655	6,215,770	683	6,271,020
516	6,458,880	544	6,314,540	572	6,221,790	600	6,184,520	628	6,177,120	656	6,218,170	684	6,272,970
517	6,453,030	545	6,307,650	573	6,220,930	601	6,186,420	629	6,177,270	657	6,219,120	685	6,275,170
518	6,444,415	546	6,300,840	574	6,219,230	602	6,183,370	630	6,177,120	658	6,221,720	686	6,278,070
519	6,434,005	547	6,298,890	575	6,214,770	603	6,185,520	631	6,179,570	659	6,223,020	687	6,280,270
520	6,428,630	548	6,294,950	576	6,213,620	604	6,185,370	632	6,180,220	660	6,225,070	688	6,283,070
521	6,422,665	549	6,295,000	577	6,212,120	605	6,187,820	633	6,180,920	661	6,226,620	689	6,285,320
522	6,418,465	550	6,287,900	578	6,210,020	606	6,187,820	634	6,182,220	662	6,228,820	690	6,287,620
523	6,410,105	551	6,285,090	579	6,207,170	607	6,186,820	635	6,184,220	663	6,231,420	691	6,290,170
524	6,407,990	552	6,283,250	580	6,207,020	608	6,186,230	636	6,185,520	664	6,232,970	692	6,292,370
525	6,400,100	553	6,280,090	581	6,205,730	609	6,184,480	637	6,186,020	665	6,235,170	693	6,295,270
526	6,391,605	554	6,276,800	582	6,201,430	610	6,184,130	638	6,187,920	666	6,237,570	694	6,297,470
527	6,389,140	555	6,270,750	583	6,198,580	611	6,184,480	639	6,189,320	667	6,239,120	695	6,300,270
528	6,381,500	556	6,270,200	584	6,197,280	612	6,180,270	640	6,191,220	668	6,241,070	696	6,302,820
529	6,374,840	557	6,269,250	585	6,196,580	613	6,180,870	641	6,192,670	669	6,242,570	697	6,305,020
530	6,375,090	558	6,263,130	586	6,190,970	614	6,180,070	642	6,194,570	670	6,244,970	698	6,307,920
531	6,366,950	559	6,262,440	587	6,191,540	615	6,178,820	643	6,196,070	671	6,245,920	699	6,310,120
532	6,365,050	560	6,258,840	588	6,188,770	616	6,177,220	644	6,198,470	672	6,248,520	700	6,312,920
533	6,358,850	561	6,256,640	589	6,186,470	617	6,177,020	645	6,199,420	673	6,249,820	701	6,316,020
534	6,355,140	562	6,251,990	590	6,184,520	618	6,176,620	646	6,201,920	674	6,251,870	702	6,319,320
535	6,351,040	563	6,250,950	591	6,184,220	619	6,176,820	647	6,203,320	675	6,253,420	703	6,322,320
536	6,346,100	564	6,247,250	592	6,183,470	620	6,178,270	648	6,205,220	676	6,255,620	704	6,325,220
537	6,340,440	565	6,244,640	593	6,185,070	621	6,176,170	649	6,205,720	677	6,258,220	705	6,327,420
538	6,339,050	566	6,244,400	594	6,184,520	622	6,177,220	650	6,207,620	678	6,259,770	706	6,330,220
539	6,335,650	567	6,236,300	595	6,184,420	623	6,178,420	651	6,209,020	679	6,261,970	707	6,333,320
540	6,327,740	568	6,234,150	596	6,184,420	624	6,177,370	652	6,210,920	680	6,264,370	708	6,336,620

Results given in Table 4.20 and Table 4.21 reveal that the optimal Pareto front comprises a total of 196 solutions for the 63-activity project. Thus, as stated previously, the external repository O, must be able to store 196 particles as $63 \times 2(5)$ matrices that represent solutions along the non-dominated front for any possible completion time. Accordingly, in behalf of increasing the probability of locating the entire Pareto front through each cycle, at least 196 particles must be generated. Respecting Table 4.22 and Table 4.23, throughout the first portion of the hybrid algorithm, modified-SAM locates 130 particles over the solution space leaving the rest of the population to be generated following a random scheme.

Table 4.22 – Results provided by modified-SAM for 63-activity problem through the first phase of hybrid algorithm (daily indirect cost of 2,300$).

Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $
513	5,913,780	566	5,718,280	584	5,572,140	615	5,515,230	668	5,455,970
516	5,914,930	566	5,721,430	584	5,575,240	616	5,504,580	668	5,461,420
519	5,916,480	567	5,673,230	585	5,572,990	616	5,510,230	671	5,460,220
522	5,918,130	569	5,659,680	586	5,573,340	618	5,500,830	672	5,456,870
526	5,912,630	569	5,662,930	587	5,562,420	618	5,506,130	673	5,457,770
529	5,913,430	569	5,674,680	587	5,567,370	619	5,498,230	674	5,458,770
530	5,870,480	570	5,659,580	587	5,572,620	620	5,491,880	677	5,463,120
534	5,874,830	571	5,656,530	588	5,559,170	620	5,495,330	679	5,465,220
537	5,866,230	572	5,633,930	590	5,554,680	629	5,499,080	680	5,465,070
539	5,827,580	572	5,639,830	590	5,560,380	634	5,488,380	681	5,457,420
539	5,868,080	572	5,653,680	592	5,552,830	639	5,490,080	681	5,463,920
540	5,820,830	573	5,619,805	597	5,559,930	645	5,497,130	682	5,455,320
541	5,820,680	573	5,624,730	600	5,563,730	646	5,480,930	682	5,456,170
542	5,808,330	575	5,615,905	602	5,544,030	648	5,482,880	683	5,456,020
542	5,817,530	575	5,620,705	602	5,566,330	650	5,469,330	684	5,457,170
544	5,799,180	576	5,611,855	604	5,540,180	650	5,475,430	685	5,457,870
546	5,793,980	577	5,603,680	604	5,544,930	650	5,481,430	686	5,459,670
548	5,789,830	577	5,608,455	605	5,531,130	653	5,473,330	687	5,456,170
552	5,781,530	578	5,595,230	605	5,537,930	655	5,463,330	690	5,459,620
553	5,775,330	578	5,602,830	608	5,533,330	655	5,471,280	692	5,461,970
555	5,772,630	579	5,586,290	609	5,519,780	656	5,458,880	697	5,468,620
556	5,729,180	579	5,590,480	609	5,528,930	658	5,450,670	699	5,471,320
559	5,727,330	580	5,585,740	611	5,516,430	658	5,458,230	700	5,472,920
560	5,720,030	581	5,579,040	611	5,520,730	658	5,461,280	706	5,483,020
564	5,720,230	581	5,583,990	613	5,516,330	661	5,453,420	707	5,484,920
566	5,707,430	582	5,575,140	615	5,509,730	664	5,455,970	708	5,487,020

Table 4.23 – Results provided by modified-SAM for 63-activity problem through the first phase of hybrid algorithm (daily indirect cost of 3,500$).

Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $	Dur. (days)	Cost $
513	6,529,380	566	6,397,480	584	6,272,940	615	6,253,230	668	6,257,570
516	6,534,130	566	6,400,630	584	6,276,040	616	6,243,780	668	6,263,020
519	6,539,280	567	6,353,630	585	6,274,990	616	6,249,430	671	6,265,420
522	6,544,530	569	6,342,480	586	6,276,540	618	6,242,430	672	6,263,270
526	6,543,830	569	6,345,730	587	6,266,820	618	6,247,730	673	6,265,370
529	6,548,230	569	6,357,480	587	6,271,770	619	6,241,030	674	6,267,570
530	6,506,480	570	6,343,580	587	6,277,020	620	6,235,880	677	6,275,520
534	6,515,630	571	6,341,730	588	6,264,770	620	6,239,330	679	6,280,020
537	6,510,630	572	6,320,330	590	6,262,680	629	6,253,880	680	6,281,070
539	6,474,380	572	6,326,230	590	6,268,380	634	6,249,180	681	6,274,620
539	6,514,880	572	6,340,080	592	6,263,230	639	6,256,880	681	6,281,120
540	6,468,830	573	6,307,405	597	6,276,330	645	6,271,130	682	6,273,720
541	6,469,880	573	6,312,330	600	6,283,730	646	6,256,130	682	6,274,570
542	6,458,730	575	6,305,905	602	6,266,430	648	6,260,480	683	6,275,620
542	6,467,930	575	6,310,705	602	6,288,730	650	6,249,330	684	6,277,970
544	6,451,980	576	6,303,055	604	6,264,980	650	6,255,430	685	6,279,870
546	6,449,180	577	6,296,080	604	6,269,730	650	6,261,430	686	6,282,870
548	6,447,430	577	6,300,855	605	6,257,130	653	6,256,930	687	6,280,570
552	6,443,930	578	6,288,830	605	6,263,930	655	6,249,330	690	6,287,620
553	6,438,930	578	6,296,430	608	6,262,930	655	6,257,280	692	6,292,370
555	6,438,630	579	6,281,090	609	6,250,580	656	6,246,080	697	6,305,020
556	6,396,380	579	6,285,280	609	6,259,730	658	6,240,270	699	6,310,120
559	6,398,130	580	6,281,740	611	6,249,630	658	6,247,830	700	6,312,920
560	6,392,030	581	6,276,240	611	6,253,930	658	6,250,880	706	6,330,220
564	6,397,030	581	6,281,190	613	6,251,930	661	6,246,620	707	6,333,320
566	6,386,630	582	6,273,540	615	6,247,730	664	6,252,770	708	6,336,620

Once again, the time intervals within which the feasible realizations of durations will be explored, have been evaluated using the total durations of the crashed (Z_{min}) and the all normal (Z_{max}) schedules. Thereupon, the minimum population sizes have been identified with respect to the evaluated time intervals. Subsequently, performing several trials with minimum swarm sizes of 196, the adequate values for the operators of the algorithm have been designated as shown in Table 4.24.

Table 4.24 – Parameters of the SAM-PSO model for the 63-activity problem.

Parameter	Value
t	500
i	500
c_1	2
c_2	2
w_{max}	2.2
w_{min}	1.0
v_{max}	2

After a series of trial and error, 500 generations with a swarm size of 500 are practiced for SAM-PSO, which provides sufficient convergence capabilities for solution of the 63-activity problem. The cognition and social coefficients remain untouched with selected values of integer 2, contributing weighted average of 1 for the second and the third terms of Eq. (1.18). For this experiment, the inertia weight during performance of the PSO phase is defined to linearly reduce from a maximum value of 2.2 to a minimum amount of 1.0. As depicted earlier, selection of region with larger values for the inertia weight enhances the PSO phase with broader explorations. Unrestrained escalation of velocities that promote swarm divergence is eliminated clamping the velocity to the feasible region of $[-2,2]$, which contributes calculations of probabilities of range 0.12 to 0.88 via Eq. (3.19). Smaller values of v_{max} provide more exploration by this algorithm.

Ten consecutive runs have been executed for any assumed value of daily indirect cost. Thereafter, average percent deviations (APD) from the optima have been measured for any solution located along the non-dominated front. The ultimate Pareto fronts archived by the hybrid algorithm, for two cases of the 63-activity problem are illustrated in Table 4.25 and Table 4.26. Akin to the previous tests, corresponding to any certain duration determined by the hybrid model, the least total cost observed over ten incessant experiments is recorded throughout the computational experiments. In addition to APD's of each solution, these tables also contain the overall APD's as an indication of the performance of the hybrid algorithm. In the same way as the former experiments, it has been observed that the overall APD is slightly larger for the case with smaller assumed value of indirect cost. The average processing times required by the proposed model to unravel the 63-activity problem are reported in the last rows of Table 4.25 and Table 4.26. The algorithm searching 1.82×10^{-35} fraction of the solution space, takes a reasonable CPU time of 108 to 111 seconds to locate the time-cost profile for the 63-activity project. These results further verify the robustness of the hybrid model in obtaining the entire Pareto front for the discrete TCT problems. Searching rather small portion of the solution space, the proposed model provides the complete efficient frontier with inconsiderable deviations from the optimal solutions.

Table 4.25 – Complete Pareto front of 63-activity problem obtained by SAM-PSO model (daily indirect cost of 2,300$).

Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %
513	5,898,380	0.61	562	5,589,880	0.27	611	5,452,480	0.04	660	5,438,270	0.11
514	5,901,380	0.68	563	5,583,880	0.19	612	5,451,730	0.13	661	5,441,320	0.16
515	5,872,390	0.59	564	5,576,580	0.18	613	5,453,380	0.17	662	5,441,670	0.16
516	5,850,755	0.31	565	5,573,930	0.22	614	5,445,570	0.13	663	5,440,920	0.11
517	5,837,830	0.17	566	5,571,480	0.20	615	5,446,620	0.22	664	5,440,970	0.10
518	5,833,380	0.23	567	5,567,640	0.22	616	5,459,970	0.75	665	5,441,970	0.09
519	5,815,230	0.30	568	5,564,890	0.23	617	5,454,630	0.62	666	5,440,820	0.05
520	5,810,530	0.30	569	5,555,280	0.15	618	5,450,930	0.59	667	5,441,170	0.05
521	5,800,030	0.23	570	5,550,180	0.10	619	5,448,730	0.56	668	5,441,270	0.04
522	5,794,605	0.19	571	5,635,740	1.75	620	5,445,980	0.39	669	5,439,770	0.00
523	5,788,055	0.15	572	5,627,480	1.68	621	5,441,820	0.26	670	5,441,120	0.02
524	5,785,565	0.17	573	5,613,880	1.48	622	5,439,470	0.17	671	5,441,470	0.01
525	5,778,715	0.17	574	5,592,650	1.26	623	5,433,870	0.12	672	5,442,170	0.00
526	5,788,165	0.51	575	5,587,300	1.27	624	5,433,320	0.10	673	5,443,170	0.04
527	5,772,905	0.40	576	5,580,700	1.20	625	5,431,120	0.08	674	5,444,570	0.04
528	5,764,290	0.39	577	5,579,490	1.21	626	5,428,820	0.05	675	5,444,920	0.05
529	5,740,040	0.23	578	5,576,990	1.14	627	5,429,020	0.06	676	5,458,520	0.34
530	5,741,240	0.14	579	5,580,900	1.24	628	5,426,420	0.09	677	5,447,120	0.13
531	5,737,290	0.21	580	5,565,680	1.08	629	5,425,220	0.06	678	5,447,470	0.09
532	5,728,050	0.16	581	5,555,200	1.04	630	5,424,920	0.11	679	5,447,170	0.01
533	5,722,200	0.15	582	5,556,300	1.06	631	5,471,830	1.22	680	5,449,170	0.04
534	5,720,400	0.23	583	5,548,980	1.03	632	5,446,770	0.55	681	5,450,970	0.04
535	5,719,000	0.23	584	5,545,970	1.00	633	5,438,420	0.39	682	5,449,720	0.00
536	5,709,150	0.17	585	5,539,240	0.94	634	5,435,220	0.26	683	5,454,220	0.07
537	5,705,450	0.20	586	5,551,940	1.17	635	5,426,170	0.15	684	5,454,920	0.07
538	5,698,400	0.09	587	5,536,830	0.92	636	5,425,270	0.13	685	5,454,820	0.06
539	5,693,450	0.09	588	5,531,830	0.90	637	5,423,870	0.10	686	5,455,170	0.03
540	5,685,600	0.12	589	5,531,670	0.95	638	5,425,870	0.15	687	5,456,170	0.01
541	5,682,280	0.25	590	5,526,170	0.91	639	5,424,770	0.11	688	5,457,470	0.00

Table 4.25 – Complete Pareto front of 63-activity problem obtained by SAM-PSO model (daily indirect cost of 2,300$). (*Continued*)

Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %
542	5,677,290	0.30	591	5,522,170	0.88	640	5,425,570	0.12	689	5,458,520	0.00
543	5,672,750	0.32	592	5,514,780	0.78	641	5,426,920	0.10	690	5,459,620	0.00
544	5,668,580	0.24	593	5,511,220	0.69	642	5,424,870	0.08	691	5,461,220	0.01
545	5,665,080	0.33	594	5,504,880	0.64	643	5,425,170	0.06	692	5,461,970	0.00
546	5,657,080	0.31	595	5,504,680	0.66	644	5,428,570	0.08	693	5,464,370	0.04
547	5,653,840	0.30	596	5,488,230	0.46	645	5,429,670	0.09	694	5,465,270	0.03
548	5,653,070	0.38	597	5,485,630	0.39	646	5,465,780	0.85	695	5,466,270	0.02
549	5,640,230	0.23	598	5,476,120	0.32	647	5,455,970	0.57	696	5,468,170	0.01
550	5,633,630	0.23	599	5,470,670	0.25	648	5,438,270	0.42	697	5,468,620	0.00
551	5,627,280	0.21	600	5,468,170	0.25	649	5,432,320	0.36	698	5,470,320	0.01
552	5,624,180	0.17	601	5,490,030	0.88	650	5,432,070	0.12	699	5,471,320	0.00
553	5,622,480	0.19	602	5,480,420	0.37	651	5,430,470	0.09	700	5,472,920	0.00
554	5,617,130	0.18	603	5,476,180	0.28	652	5,430,720	0.08	701	5,474,820	0.00
555	5,609,630	0.24	604	5,466,880	0.19	653	5,431,870	0.08	702	5,476,920	0.00
556	5,714,780	2.02	605	5,465,580	0.18	654	5,432,220	0.06	703	5,478,720	0.00
557	5,615,640	0.33	606	5,465,180	0.16	655	5,431,470	0.03	704	5,481,470	0.02
558	5,611,230	0.36	607	5,461,280	0.12	656	5,431,970	0.07	705	5,481,420	0.01
559	5,609,570	0.35	608	5,465,770	0.17	657	5,431,820	0.09	706	5,483,020	0.00
560	5,603,240	0.31	609	5,456,620	0.10	658	5,438,670	0.12	707	5,484,920	0.00
561	5,593,740	0.23	610	5,452,320	0.02	659	5,439,620	0.14	708	5,487,020	0.00
							Overall APD (%)			0.31	
							Avg. CPU time (s)			111.51	

Table 4.26 – Complete Pareto front of 63-activity problem obtained by SAM-PSO model (daily indirect cost of 3,500$).

Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %
513	6,513,980	0.55	562	6,255,800	0.10	611	6,189,230	0.08	660	6,230,020	0.10
514	6,516,030	0.60	563	6,256,000	0.09	612	6,189,070	0.15	661	6,244,620	0.29
515	6,496,330	0.53	564	6,248,600	0.05	613	6,181,980	0.09	662	6,243,520	0.26
516	6,481,780	0.38	565	6,249,800	0.09	614	6,181,620	0.09	663	6,240,920	0.18

Table 4.26 – Complete Pareto front of 63-activity problem obtained by SAM-PSO model (daily indirect cost of 3,500$). (*Continued*)

Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %
517	6,478,730	0.41	566	6,248,900	0.10	615	6,179,720	0.09	664	6,238,420	0.10
518	6,468,565	0.43	567	6,244,850	0.15	616	6,204,470	0.68	665	6,237,820	0.08
519	6,458,280	0.42	568	6,240,100	0.12	617	6,200,720	0.60	666	6,240,020	0.04
520	6,454,590	0.41	569	6,235,730	0.10	618	6,190,970	0.47	667	6,239,170	0.03
521	6,447,865	0.42	570	6,235,280	0.07	619	6,187,470	0.39	668	6,241,770	0.03
522	6,439,215	0.36	571	6,302,440	1.40	620	6,187,520	0.42	669	6,243,070	0.03
523	6,428,955	0.33	572	6,290,290	1.31	621	6,183,420	0.38	670	6,245,120	0.03
524	6,431,055	0.37	573	6,304,005	1.34	622	6,182,370	0.37	671	6,246,670	0.04
525	6,423,050	0.38	574	6,279,600	1.13	623	6,181,670	0.34	672	6,248,570	0.02
526	6,403,850	0.35	575	6,277,950	1.16	624	6,180,120	0.25	673	6,252,920	0.05
527	6,399,350	0.38	576	6,271,000	1.08	625	6,181,320	0.23	674	6,254,820	0.05
528	6,388,140	0.22	577	6,270,790	1.09	626	6,178,970	0.16	675	6,257,020	0.06
529	6,391,390	0.30	578	6,265,950	1.04	627	6,178,770	0.09	676	6,259,070	0.12
530	6,376,890	0.05	579	6,273,940	1.08	628	6,181,170	0.15	677	6,261,270	0.06
531	6,373,400	0.14	580	6,271,900	1.07	629	6,180,120	0.12	678	6,263,170	0.07
532	6,369,100	0.10	581	6,260,040	0.98	630	6,182,720	0.13	679	6,264,220	0.05
533	6,361,850	0.05	582	6,254,590	0.94	631	6,208,080	0.78	680	6,264,370	0.03
534	6,361,940	0.14	583	6,221,450	0.65	632	6,189,480	0.41	681	6,265,920	0.04
535	6,355,440	0.11	584	6,202,320	0.52	633	6,188,480	0.21	682	6,268,120	0.00
536	6,351,650	0.10	585	6,249,940	0.87	634	6,188,420	0.18	683	6,271,020	0.01
537	6,342,890	0.09	586	6,248,420	0.95	635	6,190,620	0.13	684	6,273,220	0.02
538	6,341,130	0.08	587	6,242,370	0.86	636	6,191,230	0.11	685	6,275,170	0.01
539	6,340,980	0.11	588	6,238,730	0.83	637	6,192,220	0.11	686	6,278,070	0.00
540	6,335,180	0.14	589	6,234,570	0.80	638	6,189,820	0.07	687	6,280,270	0.00
541	6,345,940	0.54	590	6,229,880	0.82	639	6,192,170	0.14	688	6,283,070	0.00
542	6,340,405	0.38	591	6,196,600	0.55	640	6,193,570	0.07	689	6,285,320	0.00
543	6,333,465	0.35	592	6,192,380	0.42	641	6,195,470	0.06	690	6,287,620	0.00
544	6,326,400	0.26	593	6,196,830	0.43	642	6,197,170	0.07	691	6,291,670	0.03
545	6,325,400	0.30	594	6,188,480	0.32	643	6,198,420	0.06	692	6,292,370	0.00
546	6,318,000	0.30	595	6,187,470	0.32	644	6,200,470	0.06	693	6,295,270	0.01
547	6,315,330	0.27	596	6,190,070	0.24	645	6,202,020	0.06	694	6,297,470	0.00

Table 4.26 – Complete Pareto front of 63-activity problem obtained by SAM-PSO model (daily indirect cost of 3,500$). (*Continued*)

Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %	Dur. (days)	Cost $	APD %
548	6,303,350	0.16	597	6,205,120	0.35	646	6,240,980	0.75	695	6,300,270	0.02
549	6,304,200	0.16	598	6,208,930	0.41	647	6,214,770	0.42	696	6,302,820	0.00
550	6,299,430	0.19	599	6,202,280	0.34	648	6,225,270	0.48	697	6,305,020	0.00
551	6,295,600	0.26	600	6,197,580	0.29	649	6,219,320	0.37	698	6,307,920	0.00
552	6,290,500	0.18	601	6,206,330	0.33	650	6,217,570	0.29	699	6,310,120	0.00
553	6,287,450	0.21	602	6,199,630	0.27	651	6,218,870	0.29	700	6,312,920	0.00
554	6,287,400	0.22	603	6,197,230	0.25	652	6,210,920	0.18	701	6,316,020	0.01
555	6,280,290	0.24	604	6,190,930	0.20	653	6,213,870	0.16	702	6,319,320	0.01
556	6,300,180	1.16	605	6,191,530	0.15	654	6,216,170	0.14	703	6,322,320	0.00
557	6,281,540	0.29	606	6,189,830	0.08	655	6,218,070	0.16	704	6,325,220	0.01
558	6,282,210	0.34	607	6,191,930	0.14	656	6,220,020	0.13	705	6,327,420	0.00
559	6,268,940	0.18	608	6,187,530	0.09	657	6,220,820	0.16	706	6,330,220	0.00
560	6,263,040	0.14	609	6,185,930	0.04	658	6,228,270	0.12	707	6,333,320	0.00
561	6,263,450	0.12	610	6,185,470	0.02	659	6,230,420	0.12	708	6,336,620	0.00
							Overall APD (%)			0.27	
							Avg. CPU time (s)			108.02	

The solutions obtained from mixed integer programming (Sonmez and Bettemir 2012), modified-SAM, and SAM-PSO procedures are once more plotted against each other in the ensuing figures. Figure 4.6 and Figure 4.7 illuminate the solutions obtained via SAM-PSO model for the 63-activity problem with indirect costs of $2{,}300\$/day$ and $3{,}500\$/day$ respectively. These figures unveil a reasonably good fit for 130 solutions provided by modified-SAM for any of the experimented cases. Followed by the overhauled PSO algorithm, these solutions, in essence, impart the particles with somewhat of a head-start. Thus, solutions located through this phase provide the second stage with seeds of higher quality. The best non-dominated solutions located within ten consecutive trials establish the final time-cost profile. This profile, to some extent, lies over the optimal frontier obtained through the mixed integer programming; besides, the deviations occur within an acceptable range from the optima.

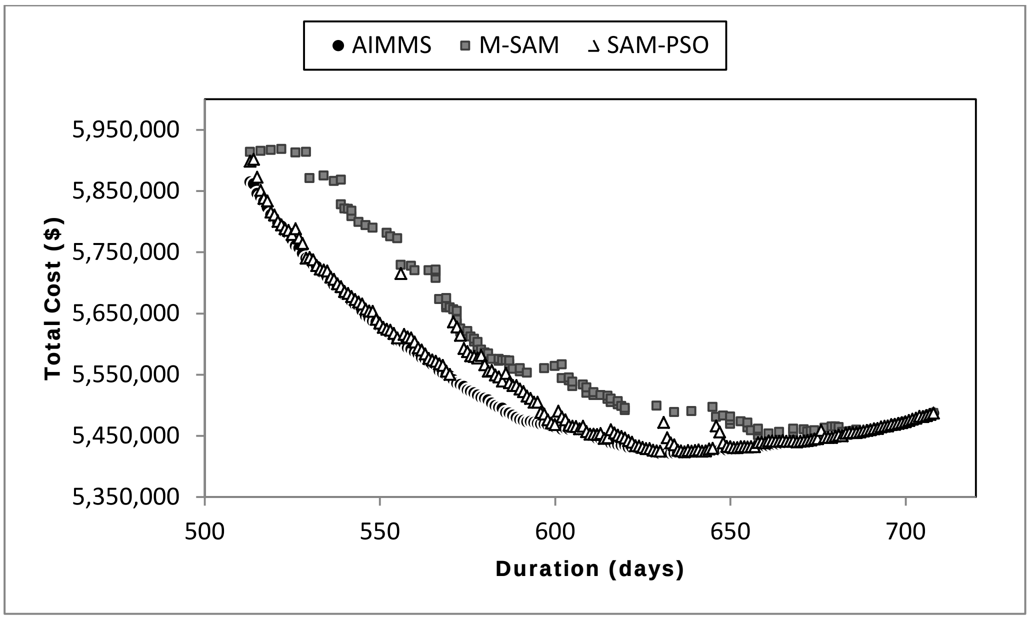

Figure 4.6 – Comparison of obtained Pareto fronts for 63-activity problem (daily indirect cost of 2,300$).

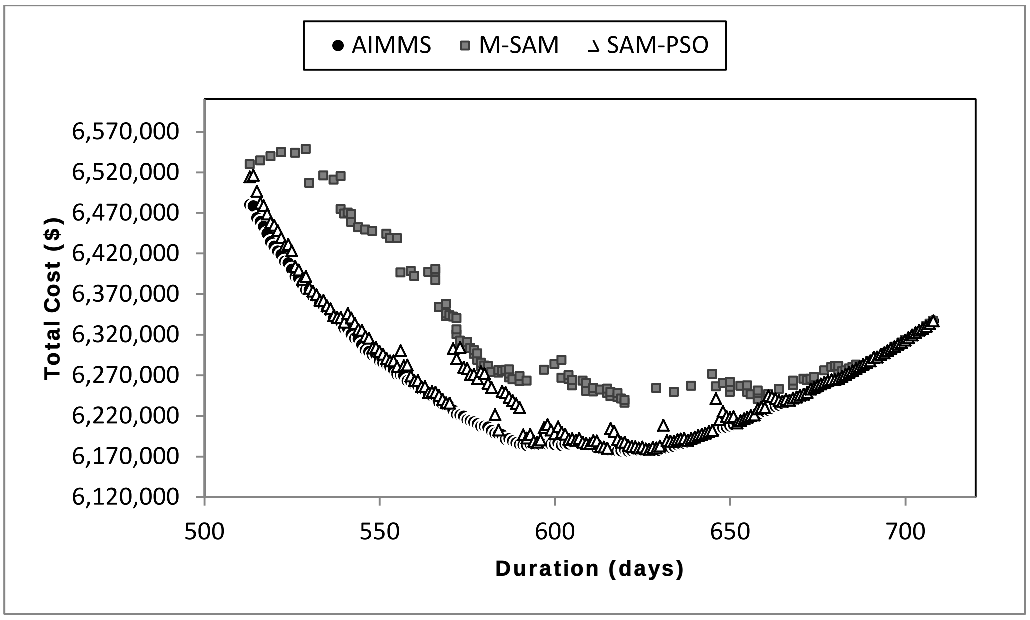

Figure 4.7 – Comparison of obtained Pareto fronts for 63-activity problem (daily indirect cost of 3,500$).

Throughout the computational experiments, the optimal non-dominated time-cost profiles of the 18-activity and the 63-activity problems were introduced for the very first time in the literature. Due to sound convergence speed and quality of the proposed algorithm in locating the entire Pareto fronts of the noted instances, it is considered to be a pioneering technique in the construction management spectrum.

CHAPTER 5

CONCLUSIONS

In this thesis, significance of adequate schedules for construction projects has been discussed. Of the scheduling techniques, the TCT analyses have been brought to light along with the inadequacies of existing commercial scheduling software packages for such analyses. Stated motives tied with eminence of discretization, initiated development of PSO based algorithms for discrete TCT optimizations. Two particle swarm optimizers, as well as a hybrid meta-heuristic algorithm have been proposed for solution of TCT problems, explicitly emphasizing time-cost curve extension of these analyses. Respectively, a state-of-the-art model with decent capabilities in identifying the entire Pareto front for discrete TCT problems has been presented.

On the verge of validation and performance assessments of the proposed algorithms, empirical analyses have been conducted by implementing a well-known 18-activity benchmark problem, as well as a more complex 63-activity problem into the models. The adequate values for the operators of the algorithms have been fine-tuned via sequences of trial and error, with regard to the solutions provided for these instances within the literature. Amidst computational experiments, implementing mixed integer programming technique in the AIMMS 3.11 optimization software, the optimal non-dominated time-cost profiles of the 18-activity and the 63-activity problems have been introduced for the very first time in the literature. Therewith, the closeness of solutions obtained from the proposed particle swarm optimizers have been measured compared to the optimal solutions; the average deviations have been reported for ten consecutive experimental runs. The average processing times required to unravel the test problems have been also documented. The efficiency and robustness of the proposed algorithms have been verified concerning the results attained from these analyses.

The discrete sole-PSO algorithm introduced in this thesis has been established upon the classical version proposed by the founders, with trajectories and velocities defined as probabilities of state selection for the bits. However, modifications have been applied to the equations and an alternate scheme for position update has been adopted. The implemented scheme contrasting with the original method, in lieu of using uniformly distributed random numbers, involves determination of the alternatives holding the maximum probabilities. Similarly, notwithstanding the absence of the inertia weight (w) in the classical velocity update equation, the proposed algorithm has been treated with this parameter.

Compared to solutions of well-developed algorithms along with the optimal solution attained from the exact method, it has been verified that the proposed discrete PSO algorithm is capable of finding optimum or near-optimum solutions for the medium-sized problems with insignificant deviations from the optimal solutions. It has been observed that the quality of the obtained solutions for larger instances slightly deteriorate as exposed to larger daily indirect costs. It has been shown that this algorithm operates within acceptable processing time by searching merely small fractions of the search space. As a result, the proposed algorithm has been proven to outperform all the earlier optimizers with regard to both the convergence speed and the quality of the solutions. However, generic to all meta-heuristic algorithms, it is impossible to ensure quality of the obtained solutions short of the exact procedure.

Another paradigm of the discrete PSO algorithm has been introduced within the context of this thesis for solution of the time-constraint TCT problems. Minor modifications mainly engaging the fitness functions have been applied to the previously specified particle swarm optimizer. Performed revisions enhancing assessment of problems with provisions of incentives and liquidated damages have been demonstrated. It has been observed that in spite of extra calculations imposed to the algorithm, this procedure demands inconsiderable processing time due to searching small portions of the solution space. The efficiency of this algorithm has been confirmed in providing sound solutions throughout the empirical analyses. The optimality of the solutions obtained through experimentations has been verified in comparison with the results of the exact procedure. In consequence, the proposed algorithm has been proven to outdo all the earlier optimizers concerning its convergence capabilities.

Reckoned as the chief contribution of this thesis, a novel hybrid SAM-PSO algorithm has been introduced for solution of the time-cost curve extension of discrete TCT problems. Complementary strengths of the Siemens Approximation Method (SAM) and the discrete PSO algorithm have been combined to develop a hybrid algorithm. To this end, a new approach has been taken toward cost slopes measurements of the SAM method; thereupon, the modified-SAM has been embedded to an overhauled discrete PSO algorithm. Principles for selection of the pbest and the gbest positions have been totally revamped for the overhauled PSO phase of the hybrid model. A semi-deterministic (semi-random) initialization scheme has been incorporated into the SAM-PSO model, seeding a certain portion of the initial population by dint of the modified-SAM method, followed by a random scheme for generation of the rest of the population.

Convergence capabilities of the proposed SAM-PSO model in locating the Pareto front have been demonstrated in a scatter chart, plotted against the results of the mixed integer programming. It has been observed that the solutions provided via modified-SAM phase of the hybrid model have reasonably good fit compared to the optimal frontier, thus, providing the second phase with initial seeds of higher quality. The ultimate time-cost profiles located by the hybrid algorithm have been experienced to roughly lie over the efficient frontier with deviations within acceptable ranges from the optima. However, it has been observed that the quality of the obtained solutions slightly deteriorate for instances with smaller daily indirect costs. Robustness of this model has been confirmed regarding its competency in locating the entire non-dominated front for the medium-sized instances. As a result, operating within reasonable time-frames, the proposed algorithm has been proven to outperform the results of the previous researches reported in the literature. Albeit, as stated earlier, optimality of the obtained solutions through the SAM-PSO model cannot be ensured unless an exact procedure is adopted. Moreover, despite the efficiency of the hybrid algorithm in solving instances having up to 63 activities, the same level of performance cannot be guaranteed extending its application for real-life projects that comprise hundreds or even thousands of activities.

The SAM-PSO model is envisioned to support decision makers in competent evaluations of the subsequent "what if" scenarios. Due to sound convergence capabilities of the proposed algorithm in locating the entire Pareto fronts of the represented instances, it is considered to be a pioneering technique in the construction management spectrum. Further, it has been noted that the performance of SAM-PSO is sensitive to parameter selection, and that setting larger values for the generations and/or swarms in behalf of augmented solution quality is advisable. However, setting larger values for these parameters necessitate greater processing times. As a result, benefitting from supercomputers or grid-computing systems that render more processing units would be an alternative study area.

Despite potencies of the proposed model, a need remain for methods that incorporates resources availabilities during analyses. A comprehensive research extending optimization of time-cost to other perspectives of the projects such as quality, productivity, safety, etc., is also an investigation area that deserves further devotion. Including second order cost components – such as insurance or bond expenses that are functions of both the overall duration and cost of the project – within TCT analyses would be another remarkable focus in this spectrum. Last but not least, it is a common practice to assume deterministic amounts for the durations and costs of the time-cost alternatives; however, a model can be designed to reflect uncertainties of the actual practices.

REFERENCES

Afshar, A., Ziaraty, A. K., Kaveh, A., & Sharifi, F. (2009). Nondominated Archiving Multicolony Ant Algorithm in Time-Cost Trade-Off Optimization. Journal of Construction Engineering and Management-ASCE, 135(7), 668-674. doi: 10.1061/(ASCE)0733-9364(2009)135:7(668)

Anagnostopoulos, K. P., & Kotsikas, L. (2010). Experimental evaluation of simulated annealing algorithms for the time-cost trade-off problem. Applied Mathematics and Computation, 217(1), 260-270. doi: 10.1016/j.amc.2010.05.056

Ashuri, B., & Tavakolan, M. (2012). Fuzzy Enabled Hybrid Genetic Algorithm–Particle Swarm Optimization Approach to Solve TCRO Problems in Construction Project Planning. Journal of Construction Engineering and Management, 138(9), 1065-1074. doi: 10.1061/(ASCE)CO.1943-7862.0000513

Chassiakos, A. P., & Sakellaropoulos, S. P. (2005). Time-cost optimization of construction projects with generalized activity constraints. Journal of Construction Engineering and Management-ASCE, 131(10), 1115-1124. doi: 10.1061/(ASCE)0733-9364(2005)131:10(1115)

Colorni, A., Dorigo, M., & Maniezzo, V. (1992). Distributed Optimization by Ant Colonies. Toward a Practice of Autonomous Systems, 134-142.

De, P., James Dunne, E., Ghosh, J. B., & Wells, C. E. (1995). The discrete time-cost tradeoff problem revisited. European Journal of Operational Research, 81(2), 225-238. doi: 10.1016/0377-2217(94)00187-H

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. doi: 10.1016/0360-8352(94)00199-W

Demeulemeester, E., De Reyck, B., Foubert, B., Herroelen, W., & Vanhoucke, M. (1998). New computational results on the discrete time/cost trade-off problem in project networks. Journal of the Operational Research Society, 49(11), 1153-1163. doi: 10.1057/palgrave.jors.2600634

Demeulemeester, E. L., Herroelen, W. S., & Elmaghraby, S. E. (1996). Optimal procedures for the discrete time cost trade-off problem in project networks. European Journal of Operational Research, 88(1), 50-68. doi: 10.1016/0377-2217(94)00181-2

Eberhart, R., & Kennedy, J. (1995, 4-6 Oct 1995). A new optimizer using particle swarm theory. Paper presented at the Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 39-43.

El-Rayes, K., & Moselhi, O. (1998). Resource-driven scheduling of repetitive activities. Construction Management and Economics, 16(4), 433-446. doi: 10.1080/014461998372213

Elbeltagi, E., Hegazy, T., & Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. Advanced Engineering Informatics, 19(1), 43-53. doi: 10.1016/j.aei.2005.01.004

Elbeltagi, E., Hegazy, T., & Grierson, D. (2007). A modified shuffled frog-leaping optimization algorithm: applications to project management. Structure and Infrastructure Engineering, 3(1), 53-60. doi: 10.1080/15732470500254535

Eshtehardian, E., Afshar, A., & Abbasnia, R. (2008). Time–cost optimization: using GA and fuzzy sets theory for uncertainties in cost. Construction Management and Economics, 26(7), 679-691. doi: 10.1080/01446190802036128

Falk, J. E., & Horowitz, J. L. (1972). Critical Path Problems with Concave Cost-Time Curves. Management Science Series B-Application, 19(4), 446-455. doi: 10.1287/mnsc.19.4.446

Feng, Liu, L., & Burns. (1997). Using Genetic Algorithms to Solve Construction Time-Cost Trade-Off Problems. Journal of Computing in Civil Engineering, 11(3), 184-189. doi: 10.1061/(ASCE)0887-3801(1997)11:3(184)

Foldes, S., & Soumis, F. (1993). Pert and Crashing Revisited - Mathematical Generalizations. European Journal of Operational Research, 64(2), 286-294. doi: 10.1016/0377-2217(93)90183-N

Force, University of Texas at Austin. Construction Industry Institute. Cost/Schedule Controls Task Force (1988). Concepts and Methods of Schedule Compression: The Institute.

Fulkerson, D. R. (1961). A Network Flow Computation for Project Cost Curves. Management Science, 7(2), 167-178. doi: 10.1287/Mnsc.7.2.167

Goldberg, D. E., & Segrest, P. (1987). Finite Markov chain analysis of genetic algorithms. Paper presented at the Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application, Cambridge, Massachusetts, United States.

Hegazy, T. (1999). Optimization of construction time-cost trade-off analysis using genetic algorithms. Canadian Journal of Civil Engineering, 26(6), 685-697. doi: 10.1139/cjce-26-6-685

Hegazy, T., Elbeltagi, E., & El-Behairy, H. (2004). Bridge deck management system with integrated life-cycle cost optimization. Maintenance and Management of Pavement and Structures(1866), 44-50.

Heppner, F., & Grenander, U. (1990). A Stochastic Nonlinear Model for Coordinated Bird Flocks. Ubiquity of Chaos, 233-238.

Herroelen, W. S., VanDommelen, P., & Demeulemeester, E. L. (1997). Project network models with discounted cash flows a guided tour through recent developments. European Journal of Operational Research, 100(1), 97-121. doi: 10.1016/S0377-2217(96)00112-9

Holland, J. H. (1975). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence: University of Michigan Press.

Izakian, H., Ladani, B. T., Abraham, A., & Snasel, V. (2010). A Discrete Particle Swarm Optimization Approach for Grid Job Scheduling. International Journal of Innovative Computing Information and Control, 6(9), 4219-4233.

Izakian, H., Ladani, B. T., Zamanifar, K., & Abraham, A. (2009). A Novel Particle Swarm Optimization Approach for Grid Job Scheduling. Information Systems, Technology and Management-Third International Conference, Icistm 2009, 31, 100-109.

Juang, C. F. (2004). A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans Syst Man Cybern B Cybern, 34(2), 997-1006.

Keeney, R. L., & Raiffa, H. (1976). Decisions with multiple objectives : preferences and value tradeoffs. Sydney.

Kelley, J. E. (1961). Critical-Path Planning and Scheduling - Mathematical Basis. Operations Research, 9(3), 296-320. doi: 10.1287/Opre.9.3.296

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. 1995 Ieee International Conference on Neural Networks Proceedings, Vols 1-6, 1942-1948.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. Smc '97 Conference Proceedings - 1997 Ieee International Conference on Systems, Man, and Cybernetics, Vols 1-5, 4104-4108.

Kerzner, H. (2009). Project Management: A Systems Approach to Planning, Scheduling, and Controlling: John Wiley & Sons.

Lock, D. (2007). Project Management: 9Th Edition: Gower.

Meyer, W. L., & Shaffer, L. R. (1965). Extending CPM for Multifirm Project Time-Cost Curves. J. Construct. Div., ASCE, 91(1), 45-68.

Millonas, M. M. (1994). Swarms, phase transitions and collective intelligence. In C. Langton (Ed.), Artificial Life III: Addison-Wesley.

Moder, J. J., Phillips, C. R., & Davis, E. W. (1983). Project management with CPM, PERT, and precedence diagramming (3rd ed.). New York: Van Nostrand Reinhold.

Moussourakis, J., & Haksever, C. (2004). Flexible model for time/cost tradeoff problem. Journal of Construction Engineering and Management-ASCE, 130(3), 307-314. doi: 10.1061/(ASCE)0733-9364(2004)130:3(307)

Mubarak, S. (2010). Construction Project Scheduling and Control: John Wiley & Sons.

Ng, S. T., & Zhang, Y. S. (2008). Optimizing construction time and cost using ant colony optimization approach. Journal of Construction Engineering and Management-ASCE, 134(9), 721-728. doi: 10.1061/(ASCE)0733-9364(2008)134:9(721)

Parsopoulos, K. E., & Vrahatis, M. N. (2009). Particle Swarm Optimization and Intelligence: Advances and Applications: IGI Global.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. Paper presented at the Proceedings of the 14th annual conference on Computer graphics and interactive techniques.

Shi, Y. H., & Eberhart, R. (1998). A modified particle swarm optimizer. 1998 Ieee International Conference on Evolutionary Computation - Proceedings, 69-73. doi: 10.1109/Icec.1998.699146

Siemens, N. (1971). A Simple CPM Time-Cost Tradeoff Algorithm. Management Science, 17(6), B354-B363. doi: 10.2307/2629138

Skutella, M. (1998). Approximation algorithms for the discrete time-cost tradeoff problem. Mathematics of Operations Research, 23(4), 909-929. doi: 10.1287/moor.23.4.909

Sonmez, R., & Bettemir, O. H. (2012). A hybrid genetic algorithm for the discrete time-cost trade-off problem. Expert Systems with Applications, 39(13), 11428-11434. doi: 10.1016/j.eswa.2012.04.019

Vanhoucke, M. (2005). New computational results for the discrete time/cost trade-off problem with time-switch constraints. European Journal of Operational Research, 165(2), 359-374. doi: 10.1016/j.ejor.2004.04.007

Vanhoucke, M., & Debels, D. (2007). The discrete time/cost trade-off problem: extensions and heuristic procedures. Journal of Scheduling, 10(4-5), 311-326. doi: 10.1007/s10951-007-0031-y

Vanhoucke, M., Demeulemeester, E., & Herroelen, W. (2002). Discrete time/cost trade-offs in project scheduling with time-switch constraints. Journal of the Operational Research Society, 53(7), 741-751. doi: 10.1057/palgrave.jors.2601351

Xiong, Y., & Kuang, Y. P. (2008). Applying an ant colony optimization algorithm-based multiobjective approach for time-cost trade-off. Journal of Construction Engineering and Management-ASCE, 134(2), 153-156. doi: 10.1061/(ASCE)0733-9364(2008)134:2(153)

Yang, H. H., & Chen, Y. L. (2000). Finding the critical path in an activity network with time-switch constraints. European Journal of Operational Research, 120(3), 603-613. doi: 10.1016/S0377-2217(98)00390-7

Yang, I. T. (2007a). Using elitist particle swarm optimization to facilitate bicriterion time-cost trade-off analysis. Journal of Construction Engineering and Management-ASCE, 133(7), 498-505. doi: 10.1061/(ASCE)0733-9364(2007)133:7(498)

Yang, I. T. (2007b). Performing complex project crashing analysis with aid of particle swarm optimization algorithm. International Journal of Project Management, 25(6), 637-646. doi: http://dx.doi.org/10.1016/j.ijproman.2006.11.001

Zadeh, L. A. (1965). Fuzzy sets. Information and Control, 8(3), 338-353. doi: http://dx.doi.org/10.1016/S0019-9958(65)90241-X

Zhang, H., & Xing, F. (2010). Fuzzy-multi-objective particle swarm optimization for time-cost-quality tradeoff in construction. Automation in Construction, 19(8), 1067-1075. doi: 10.1016/j.autcon.2010.07.014

Zheng, D. X. M., & Ng, S. T. (2005). Stochastic time-cost optimization model incorporating fuzzy sets theory and nonreplaceable front. Journal of Construction Engineering and Management-ASCE, 131(2), 176-186. doi: 10.1061/(ASCE)0733-9364(2005)131:2(176)

Zheng, D. X. M., Ng, S. T., & Kumaraswamy, M. M. (2004). Applying a genetic algorithm-based multiobjective approach for time-cost optimization. Journal of Construction Engineering and Management-ASCE, 130(2), 168-176. doi: 10.1061/(ASCE)0733-9364(2004)130:2(168)

Zheng, D. X. M., Ng, S. T., & Kumaraswamy, M. M. (2005). Applying pareto ranking and niche formation to genetic algorithm-based multiobjective time-cost optimization. Journal of Construction Engineering and Management-ASCE, 131(1), 81-91. doi: 10.1061/(ASCE)0733-9364(2005)131:1(81)