ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

TIME–COST–QUALITY TRADE-OFF PROBLEMS USING EVOLUTIONARY ALGORITHMS

M.Sc. THESIS

Yashar ARMAGHANI

Department of Civil Engineering

Construction Management Program

SEPTEMBER 2014

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

TIME–COST–QUALITY TRADE-OFF PROBLEMS USING EVOLUTIONARY ALGORITHMS

M.Sc. THESIS

Yashar ARMAGHANI
(501111127)

Department of Civil Engineering

Construction Management Program

Thesis Advisor: Assoc. Prof. Gül POLAT TATAR

SEPTEMBER 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ZAMAN-MALİYET-KALİTE ÖDÜNLEŞİM SORUNLARININ EVRİMSEL ALGORİTMALAR KULLANIMI İLE ÇÖZÜLMESİ

YÜKSEK LİSANS TEZİ

Yashar ARMAGHANI
(501111127)

İnşaat Mühendisliği Anabilim Dalı

Yapı İşletmesi Programı

Tez Danışmanı: Doç. Dr. Gül POLAT TATAR

EYLUL 2014

Yashar Armaghani, a **M.Sc.** student of ITU **Institute of Science Engineering and Technology** student ID **501111127**, successfully defended the **thesis** entitled "**TIME–COST–QUALITY TRADE-OFF PROBLEMS USING EVOLUTIONARY ALGORITHMS**", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Gül Polat Tatar** İstanbul Technical University	....
Jury Members :	**Assoc. Prof. Dr. Esin Ergen Pehlevan** İstanbul Technical University	....
	Prof. Dr. Ekrem Manisalı İstanbul University	....

Date of Submission : 05 May 2014
Date of Defense : 23 September 2014

To my mother,

FOREWORD

I would like to acknowledge all the people who have shared the best and the worst moments of my Master's journey and have contribution to the completion of my thesis. One of the joys of completion is to look over the 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 and Management division of Istanbul 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. Gül POLAT TATAR, for her encouragements, guidance, and advice throughout this study. I would also like to thank the Research Asst. Befrin Neval BİNGÖL for her endless support, motivations, and kindness.

I would like to express my sincere appreciation to the examining committee members for their valuable comments.

I take this opportunity to offer my sincerest gratitude with all of my heart to my beloved parents, Ali Reza ARMAGHANI and Farideh HASSANZADEH. 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 and aunt for their love and affection, not to mention their frequent morale booster phone calls.

I would like to show my greatest appreciation to my best friend, greatest thing that has ever happened in my life, sibling, or maybe it is just better to call him my brother, Alireza Noroozi Roshan. He has been a wonderful mentor, his love, and continuous spiritual support have been great sources of motivation and encouragement at every stage of my life.

I kindly express my special gratitude to Mohammad Reza HASSANZADEH for his financial support and Prof. Arastou ARMAGHANI for his academic support during my undergraduate study that helped me to become a civil engineer.

I am especially grateful to Torkan Hosseinzadeh Bazargani helping me to get through the difficult times, for all the good memories and all the emotional support, and caring she provided.

I wish to express my warm and sincere gratitude to my dearest friends, Emre ASIL and Fetican KIRAZCI who were there whenever I needed them and helped me through the difficult times that I encountered whilst completing my thesis.

Last but not least, I would like to express my thankfulness to Emad GHAVAM, Soheil FARAZI JAM, and Shahrokh MEDGHALCHI 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.

September 2014 Yashar ARMAGHANI
Civil Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

CPM : Critical Path Method
LP : Linear programming
QP : Quadratic Program
DEA : Data Envelopment Analysis
DP : Dynamic Programming
SP : Separable Program
GP : Geometric Program
FP : Fractional Program
GO : Global Optimization
OR : Operations Research
EAs : Evolutionary Algorithms
GA : Genetic Algorithm
TSP : Traveling Salesman Problem
TCQ : Time Cost Quality
PSO : Particle Swarm Optimization
EDA : Estimation of Distribution Algorithms
ES : Evolutionary Strategies
NSS : Neighborhood search structure
SA : Simulated annealing

LIST OF TABLES

Page

LIST OF FIGURES

Page

TIME–COST–QUALITY TRADE-OFF PROBLEMS USING EVOLUTIONARY ALGORITHMS

SUMMARY

The time-costquality trade-off problem (TCQTP) is an important issue in the scheduling of construction and industrial projects. The construction projects are usually challenged with many conflicts, problems, and uncertainty due to a result of failure in selecting a competent contractor. To perform the project success, the owner's team has an important role to figure out these many conflicts and problems such as schedule delays, cost overrun, low productivity, and contractor selection. Contractor selection continues to be an area of significant importance and interest to organizations responsible for delivering project outcomes. The selection of contractors and their ability is complex, comprising high levels of ambiguity and uncertainty as a result of multiple conflicting objectives. Time-cost-quality trade-off for networks has proved very useful for the performance evaluation of certain types of projects. It aims to compress the project completion time by accelerating the activity durations at an expense of additional resources. The critical path method is an important tool to identify the critical activities in the critical path of an activity network in project management. The budget problem in discrete time-cost-quality trade-off scheduling selects the time-cost-quality system among the discrete set of specified modes for each activity so as to minimize the project completion time without exceeding the available budget. There may be alternative modes that solve the budget problem optimally, however each solution may have a different total cost value. The importance of the criteria used to evaluate and select contractors or suppliers has been examined. It will help construction managers to identify contractors with the best potential to deliver satisfactory outcomes in a final contractor selection process. The aim of this study is also to find the cost solutions among the optimal solutions without exceeding the available budget. The problem analyzed by writing a hybrid GA-SA code with Matlab. The solution space investigated for best options for time cost-quality trade-off without exceeding the defined budget. Optimal solutions for 18-activity network problem illustrate reasonable trade-offs.

ZAMAN-MALİYET-KALİTE KOORDINASYON SORUNU ÖNLEMLERI EVRIMSEL ALGORITMALAR KULLANIMI

ÖZET

Büyük ölçekli inşaat projelerini üstlenen ana yükleniciler, projeyi belirli iş paketlerine ayırırlar. Belirledikleri iş paketlerinin yapımı için alt yüklenicilik hizmetinden faydalanma yolunu tercih ederler. Alt yüklenicilik hizmeti alacakları firmaları belirlerken, ana yükleniciler öncelikli olarak firmaların uzmanlık alanını gözetirler. İnşaat projelerinin çok taraflı yapıda proje gerçekleştirilirken çeşitli çatışmalar ve probleme neden olur. Bunların başında da ana yüklenici firmanın iş paketleri için doğru alt yüklenicileri seçmesi gelir. İş paketleri için uygun alt yüklenicilerin seçilmemiş olmaması, projenin sözleşme koşullarına uygun olarak tamamlanamamasına yol açar. Ana yüklenici, alt yüklenicilerin ürettikleri işlerin sözleşme koşullarına uygun olmasını temin etmekle yükümlüdür. Bu nedenle, alt yüklenicilik hizmeti alınacak firmalar belirlenirken firmanın performansını doğru değerlendirebilecek ölçüm modellerine ihtiyaç vardır.

Ana yükleniciler, birlikte çalışacakları alt yüklenicileri belirlerken bu firmaların uzmanlık alanlarına göre yapımını gerçekleştirebilecekleri iş paketleri için teklif isterler. Alınan teklifler, zaman ve maliyet açısından incelenir. Ana yüklenici, iş paketi için teklif edilen "zaman" ve "maliyet" değerinden kendi açısından uygun olanını belirleyerek; teklif sahibi firmayı alt yüklenici olarak belirler. Bu seçimde "zaman ve maliyet" kriterleri kilit rol üstlenmektedirler. Ancak, inşaat projelerinde "zaman", "maliyet" kriterleri kadar önemli olan "kalite" kriteri, bu seçimde göz ardı edilmektedir. Alt yüklenici hizmeti alınacak firmalar belirlenirken "kalite" kriterinin göz ardı edilmesi çoğunlukla uygun olan adayların belirlememesine neden olur. Bu nedenle, ana yükleniciler birlikte çalışacakları alt yüklenici adaylarını belirlerken projenin sözleşme koşullarına uygun zaman, maliyet ve kalitede tamamlanabilmesini sağlayacak firmaları iş paketleri için seçmelidirler.

İnşaat sektöründe alt yüklenici seçimi, en düşük teklif fiyatı üzerinden yapılmaktadır. Seçimin sadece maliyet kriteri üzerinden gerçekleştirilmesi; çeşitli imalat hataları ve yeniden yapımlara neden olarak işin toplam maliyetinin artmasına neden olmaktadır. Bu nedenle, işverene karşı projenin belirlenen zaman ve maliyette ve istenilen kalite standardına uygun olarak tamamlanması ile sözleşme kapsamında sorumlu olan ana yüklenici birlikte çalışacağı alt yüklenici adaylarını belirlerken firmaların yapım performansını doğru değerlendirmelidir.

Literatürde, son on yıldır inşaat projelerinin "zaman-maliyet-kalite" kriterleri açısından performansını değerlendiren birçok çalışma bulunmaktadır. Ancak, birçok ana yüklenicinin üstlendikleri projelerin büyük bir kısmını alt yüklenicilik hizmeti ile gerçekleştirdiği bilinmektedir. Ana yüklenicilerin, hizmetinden faydalanacağı alt yüklenici adayını belirlerken kullanabileceği çok kriterli modellere ait yapılan

çalışmaların sayısı oldukça sınırlıdır. Bu nedenle, bu tezin amacı ana yüklenicilerin birlikte çalışacakları alt yüklenici adaylarını belirlerken kullanabilecekleri bir seçim modeli geliştirmektir. Geliştirilen modelde "zaman-maliyet-kalite" kriterleri seçim için kullanılacak temel amaçlardır. Bu açıdan tez kapsamında incelen problem bir ödünleşim problemi olup; "zaman-maliyet-kalite" kriterleri arasında uygun dengenin sağlandığı durumlar problemin çözüm kümesini teşkil eder.

İnşaat projelerindeki "zaman" açısından ele alınan ödünleşim problemlerinde, sürenin kısaltılması ek kaynaklar kullanılarak zamanın kısaltılması amaçlanır. Zamanın kısaltılması genellikle ek maliyet ve düşük kalite standardında üretime neden olur. İnşaat projelerinde toplam proje süresinin belirlenmesi için kullanılan yöntemlerden biri "Kritik Yol Yöntemidir". Kritik yol yöntemi, projede yer alan aktivitelerin kritikliğinin belirlenmesi açısından önemli bir araçtır. Bu yöntemle, projenin toplam süresi, kritik yol ve aynı zamanda projede bazı durumlarda geciktirebilecek veya daha uzun sürede yapımı gerçekleştirebilecek aktiviteler belirlenebilir. Tez çalışmasında incelenen problemde projenin toplam süresinin belirlenmesi için "Kritik Yol Yönteminden" faydalanılmıştır.

"Ayrık zaman-maliyet-kalite ödünleşim problemi" çizelgesi bütçe kısıtına uygun bütçeyi geçmeden, projenin tamamlanma süresini en aza indirecek şekilde her aktivite için özelleştirilmiş yöntemlerin ayrık kümeleri arasında en uygun "zaman-maliyet-kalite" değerlerini içeren çözümleri seçer. Bütçe problemini uygun şekilde çözen alternatif yöntemler olabilir ancak elde edilen her çözüm farklı toplam maliyet değerine sahip olabilir. Bu tez çalışması kapsamında, ana yüklenici, alt yüklenici ve tedarikçi seçiminde ve değerlendirilmesinde kullanılan kriterler ve bu kriterlerin önemi incelenmiştir. Bu çalışmanın temel amacı, ana yüklenicinin birlikte çalışacağı alt yüklenici adayını belirlerken kullanabileceği bir model geliştirmektir. Geliştirilen model ile yapım performansı yüksek olan alt yüklenicilerin iş paketlerinin gerçekleştirilmesi için seçimin sağlanacaktır. Ayrıca bu çalışmanın bir diğer amacı belirlenmiş olan bütçeyi aşmadan optimum çözümler arasından uygun maliyet değerine sahip alt yüklenici dizilimlerinin iş paketleri için belirlenmesidir.

"Ayrık zaman-maliyet-kalite ödünleşim probleminin" çözümü için Matlab yazılımında bir hibrit GA-SA (Hybrid Genetic Algortihm-Simmulated Annealing) algoritma geliştirilmiştir. Problemin çözüm uzayı belirlenmiş olan bütçeyi aşmadan, "zaman-maliyet-kalite ödünleşimi" için en iyi dizilimleri elde edilmesiyle oluşturulmuştur. Tez çalışmasında ele alınan inşaat projesi toplam 18-aktivite bulunmaktadır. Bu proje için optimum çözümleri içeren ödünleşimler araştırılmıştır.

Tez çalışmasında, inşaat projelerindeki "zaman-maliyet-kalite ödünleşim probleminin" çözümü için GA ve SA meta höristik yöntemlerini kullanarak optimizasyon ile bu probleme çözüm aramak amaçlanmıştır. Bu optimizasyon problemi MATLAB R2013b üzerinde geliştirilen bir hibrit algoritma kodalanarak çözülmüştür. "Zaman-maliyet-kalite ödünleşim (Time-Cost-Quality-Tradeoff) problemi" proje yönetimi alanında önemli bir araştırma alanını kapsar. Bu nedenle, , TCQT (Time-Cost-Quality-Tradeoff) problemi birçok araştırmacının ilgisini çekmiştir. TCQT problemlerinin; zaman problemi, bütçe problemi ve kalite problemi olmak üzere üç ana gruba kategorize edilebileceği gözlemlenmiştir. Bu çalışmada kurulmuş olan modelin amacı, modeldeki çeşitli kısıtlamalara ve sınırlara bağlı kalarak her bir aktivite için maksimum nitelikleri yanında projenin belirlenmiş olan süresinin altında çözümler bulmaktır. Model için birçok alternatif uygun çözüm olabilir. Pratik olarak amaç; toplam tamamlama zamanının azaltılması, minimum bütçe koşulu dikkate alınarak kalitenin artırılması ve en uygun bütçeli çözüme ulaşılmasıdır.

Tez çalışmasında, TCQT problemleri için daha verimli optimizasyon yöntemlerine ihtiyaç duyulduğunun görülmesi üzerine bu problemin çözümü için bir hibrit algoritma geliştirilmiştir. Geliştirilen hibrit algoritmada kullanılan üst sezgisel yöntemler benzetimli tavlama (Simulated Annealing-SA) ve genetik algoritma (Genetic Algortihm-GA) yöntemleridir. Benzetilmiş tavlama (SA) ve genetik algoritma (GA) tamamlayıcı güçleri ve zayıf yönleri ile modern sezgisel yaklaşımlar arasında, etkili ve güçlü birleştirilmiş optimizasyon algoritmalarını temsil eder. İki paradigmanın ilgili avantajlarının uygulanışı; genetik algoritmanın yerel arama yeteneğini geliştirmek, GA-SA algoritmaların etkili bir kombinasyonunu oluşturmak, hibrit GA-SA, zaman-maliyet-kalite ödünleşmesi (TCQT) problemini çözmek için geliştirilmiştir.

Tez çalışmasında kullanılan örnek olarak bir karayolu inşaat projesi kullanılmıştır. Bu projede çalışılacak alt yüklenicilerin ana yüklenici tarafından seçimi sırasında yazılan hibrit algoritma kullanılmıştır. Otoyol projesinde 18-aktivite için uygun alt yüklenicilerin belirlenmesi sağlanmıştır. Bu araştırmanın temel amacı, maliyeti ve zamanı minimize ederken ve projenin kalitesini maksimize eden çözümlere ulaşmaktır. Bu optimizasyon probleminin geliştirilen hibrit algoritma ile çözümünün gerçekleştirilmesi ile baskın olmayan çözümleri içeren Pareto Font kümesi elde edilmiştir. Tez çalışmasının en önemli katkılarından biri bütçe kısıtının probleme eklenmesidir. Yine bu çalışmada, baskın olmayan çözümleri kaybolmasını önlemek için en iyi çözümler bir havuzda her tekrarda tutulmuşlardır.

Tüm aktivitelerin zamanı, maliyeti ve kalitesi belirleyici kriterler olarak değerlendirilmiştir. Uygulamada, projenin alt elemanlarını çoğu olasılık ve güncel model hesaba katmaz. Daha verimli algoritmalar ve yöntemler kullanılarak daha verimli Pareto-optimal çözümler üretilebilir. Yerel optimal çözümlere takılmamak için hibrid bir GA-SA geliştirilmiştir. Tez çalışmasında, problemin çözümü için geliştirilen Matlab kodu farklı ağları çözmek için genişletilebilir ve yeni “zaman-maliyet-kalite” problemleri için yeni girdiler ve değerler kullanılarak çözümler elde edilebilir.

1. INTRODUCTION

This chapter gives the description of the problem under study, an overview on Time-Cost-Quality trade-off problems, objectives of this research, road map and a brief outline of this thesis. In this study, Genetic Algorithm has been employed as a population based meta-heuristic search method to find the best possible trade-offs of mixed integer programming formulation representing the selections of subcontractors on a case study of highway construction project.

1.1 Problem Statement

Nowadays, several government agencies have started to utilize advanced contracting methods that present incentives in order to improve the construction quality as well as conforming to the required specifications. The implementation of advanced contracting methods accommodates huge pressure on contractors to improve construction quality. It is a complicated and challenging process for any general contractor to choose a proper bid among subcontractors, which complies with the prespecified time, cost and quality of the construction project (Mungle et al., 2013).

A primary attempt in early project planning stage is to determine and assign the main activities of the large project, to several subcontractors for its completion. According to Pells, "*the selection of a subcontractor is a rigorous, difficult and time-consuming process*" (Pells, 1993). Optimum selection of subcontracting possibilities is based on topology of the project network, and the extent of the project owner information depends on the offered set of subcontractors for each activity (Genaro & Anand, 2000). Project owners do not have sufficient information about the real cost and duration of the project before execution, meanwhile those criterions mainly rely on the capability of the subcontractors accomplishing the activities. Gutierrez and Anand, (2000) indicated "*the problem of designing the contract mechanism to allocate the activities of a large project to one or several subcontractors for execution has important implication for project success*".

They conducted study on two real life projects:

(1) Laying out a belt of highway from one point to another and,

(2) Construction of a gas distribution network in the city.

The outcome of their research implied that the project duration could be reduced significantly by efficient implementation of subcontracting alternatives for large construction projects. Lately, by implementing new contracting techniques by government organizations such as Department of Transportation increases pressure on the project owners to improve the performance quality (Mungle et al., 2013).

Proposed formulation for time–cost–quality trade-off by Johnson and Liberatore, (2006) incorporates quality into conventional time–cost trade-off. The main objective was to improve the project performance by subcontracting the project activities. Validation of their proposed formulation was based on a small-scale hypothetical model. They indicated that the effective implementation of a subcontracting method might increase the performance of the project e.g. project time, cost and quality. Since their proposed formulation is not based on the multi-objective one, it simply maximizes the quality of the project while project's duration and cost are formulated as limitations of the model. Incentive and penalty cost provided to the contractor based on their project performance.

However, construction projects in real life involve greater quantity of activities with more complicated network structure. For instance, if we consider 20 activities in the project and each activity has 4 subcontracting alternatives, it generates 1.1 trillion ($\mathrm{i.e.}, 4^{20}$) possible combinations to schedule the entire project. It is even computationally not practical to select optimal subcontracting option for each activity that satisfies all the objectives, e.g. project time, cost and quality among huge solutions space with imposed constraints. Moreover, according to (Morse, 1980) a Pareto set with three conflicting objectives for such multi-objective problems can be exceptionally large and may even consist of an infinite number of solutions. Consequently, decreasing the size of the Pareto set without destroying its characteristic is desirable from administrators' point of view. This has encouraged the authors to offer an innovative method to solve computationally complex time–cost–quality trade-off problems for large construction projects.

In this study, the critical path method (CPM) method is used for calculating the duration of the project. According to Geem, (2010) critical path is the longest path of planned activities in a project network. Hence, the minimal time to complete a project can be obtained based upon its critical path. In project management, the critical path is essential because any suspension of an activity on the critical path causes overall project delay. Although the time duration of each activity is present in a simple project network, it also can vary as a function of cost. For instance, employing more workforces for an excavation activity can reduce the duration of time, however this demands a greater financial expenditure. Thus, if a certain activity is a critical activity, we can vary the activity in order to make a trade-off between time, cost and quality. We can consider a time-cost-quality trade-off problem (TCQTP) in order to simultaneously minimize project cost and project time and maximize the quality by selecting the proper option for each activity.

Since the time-cost-quality trade-off problem (TCQTP) is an important issue in the scheduling of projects in the construction sector, to solve the problem involving conflicting objectives, a hybrid GA-SA approach is proposed in this study with regard to the budget of the projects.

1.2 Time-Cost-Quality Trade-Off

Each project has a target for accomplishment and needs to be succeeded within a certain budget, within a given timeline and with available resources. There is the relationship between time, cost and quality in construction projects. This is called time-cost-quality trade-off problems and has been examined broadly in the construction management literature. Limiting one of the three aspects necessarily affects on the other two. Time-cost-quality trade-off problems are complicated task for project managers, which are not easy to handle and needs careful choice among available subcontractors for each project task. According to Gholipour and Abadi, (2013), "*Time-cost-quality trade-off, is an important management tool for overcoming one of the critical path method limitations of being unable to bring the project schedule to a specified duration.*"

Rather than pursuing to meet an arranged completion deadline, there might be a couple of other reasons for a contractor to accelerate the project. For instance, a contractor with good economic conditions might expedite a project in order to start another project; this may involve compressing the current project in favour of releasing required resources so that they can be allocated to the new project. However, project acceleration to some certain level might be profitable; meanwhile, it gets costly by decreasing the project duration beneath a certain limit. Accelerating a project schedule realistically means decreasing project duration. However, the difference between these two attitudes must primarily be clarified. Though aiming to improve the completion date of the project does not necessarily mean that the acceleration aims to reach the least possible duration (Aminbakhs, 2013).

According to the Construction Industry Institute (Gehrig G. B. et al., 1988), there are more than 90 methods available to reduce the schedule. Examining these methods, (Mubarak, 2010) has abstracted the proceeding nine methods to accelerate the project schedule:

- Review or assess the schedule to determine any inaccuracies or inadequate logical relationship or constraints
- Allocate additional human resources
- Fulfill constructability studies and value engineering
- Prevent communications disruptions among parties
- Assign over-time schedule either by increasing the hours per day and/or days per week
- Proposing incentives for more productive workers or staffs
- Accelerate accomplishing project objectives
- Renovating project management and improving supervision
- Acquire unique construction approach using specific materials and/or equipment to carry out the project

The lowest achievable total cost in time–cost trade-off analysis ascertains the practical budget limitations, in consensus to a specific deadline. The aspect of a TCT problem subsequently considered as an optimization problem, which challenges to minimize the duration of project (Siemens, 1971). Time–cost trade-off aims at delivering an optimal weighing scale of project duration and cost by analysing different patterns of decisions.

There is a relationship between a project's time to accomplishment, and its cost, and the relationship is in direct proportion. The main expenses, which the contractor has to manage during a project, are often classified into two categories; Direct costs and Indirect costs.

1.3 Types of Costs

The costs related with the project can be categorized as direct costs or indirect costs. Direct costs are those directly associated with project activities, such as salaries, transportation, and direct resources and equipment used in projects. More resources must be allocated to accelerate the speed of activities. This leads to the increase of the direct costs. Overall, in case of project acceleration, direct costs get increased. As shown in Fig. (1.1), increase in project acceleration, results in more increases in the daily cost of acceleration.

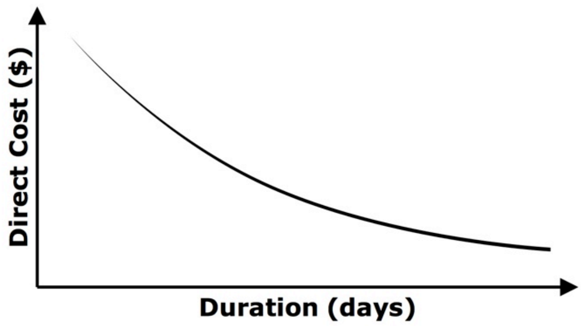

Figure 1.1 : Project acceleration results in a nonlinear escalation of direct costs.

Overhead costs that are not directly correlated with specific project activities are indirect costs. For instance, rent paid for office, building a company operates in administrative staff, and taxes considered as indirect costs.

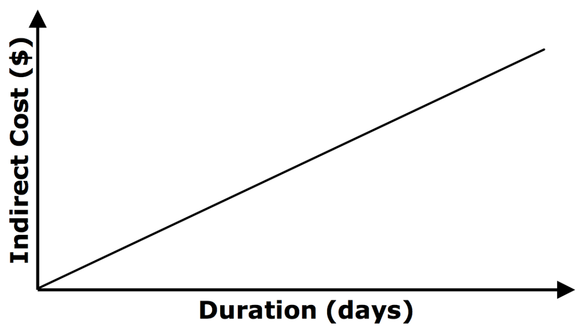

Figure 1.2 : Project acceleration cause linear decline in indirect costs.

Indirect costs tend to be reasonably balanced per time unit throughout the lifespan of the project. Cutting down project duration, results decrease in the total indirect costs.

The indirect costs decline at a constant rate in case of schedule acceleration, Fig. (1.2). Indirect costs involve mainly the overhead expenses. These may vary on daily bases such as climatic circumstances and the number of human resources (Aminbakhs, 2013).

1.4 Research Objectives

The main objectives of the research are:

- Examination of optimization, specifically, genetic algorithm (GA), its mechanism, and simulated annealing (SA),
- Construction of a computer code using Matlab R2013b for the optimization of a single-objective time-cost-quality trade-off problems by implementing of a hybrid GA-SA algorithm,
- Verification of the built code by implementing the code to a previously solved well-known model,
- A budget constraint added to this study in order to consider a practical budgeting criterion in each construction project.
- Illustrative example; implementing the code developed to an illustrative example, the time-cost-quality trade-off problems.

1.5 Research Plan

The preliminary part of the research includes a review and study with regards to the past researches in the fields of time-cost-quality trade-off problems, optimization of those systems and hybrid GA-SA algorithm.

Afterwards, the time-cost-quality trade-off optimization problem formulated as a mathematical model incorporating the decision variables, the objective function and the constraints. Following an extensive study on a new approach of hybrid GA-SA, the implementation is done in order to meet the problem objectives and optimization of the mathematical problem. Subsequently, computer code in above-mentioned stages has been developed in Matlab R2013b. Pursuing configuration of the code, the verification process has been administered by making use of a more complex network with 18 activities (Feng, Liu, & Burns, 1997), which has 2.952×10^9 candidate solutions (Geem Z. W., 2010).

The core of this study inspired from, a fuzzy clustering-based genetic algorithm approach for time–cost–quality trade-off problems, a case study of highway construction project (Mungle et al., 2013). It should be noted that there were some discrepancies between network data and computational results in the paper of Mungle et al. (2013), the pareto-optimal solution with optimal subcontracting plan. 23 out of the 25 calculations for time, cost and quality represented in that table are not matching with their optimal subcontracting alternative plans. The Tora software program, Microsoft Project 2010 and even manual calculations implemented to prove this statement.

Furthermore, the total budget constraint for the project that was not taken into consideration previously. Total budget criterion is added as a constraint which is the main contribution of this study.

1.6 Outline of the Thesis

The literature review has been carried out with regards to the previous researches concerning the time-cost-quality trade-off problem; optimization, genetic algorithm and simulated annealing topics have been studied in Chapter 2. Hybrid GA-SA algorithm which represents powerful combinatorial optimization methods, has been examined and implemented in Chapter 3, Methodology. Chapter 4 incorporates computational results. Chapter 5 represents the conclusion and future works followed by “References” and “Appendices”.

2. LITERATURE REVIEW

This chapter presents a literature survey focused on optimization, genetic algorithm and simulated annealing for Time-Cost-Quality trade-off problems. Defining optimization rules and many different techniques have been studied with regards to illustrated optimization example in this chapter.

2.1 Introduction

Optimization, also known as, mathematical programming is an arithmetical discipline that aimed at finding minimum and maximum of functions, subject to constraints or obstacles/limiting factors (Lee, Uwe, & Shin, 2009). Optimization emanated in the 1940s, American mathematical scientist, Dantzig used these methods for creating some programs (training timetables and schedules) for military application. His "linear programming" methods were practiced in an extensive range of problems such as, scheduling of production facilities, to yield management in airlines. Nowadays, optimization includes a broad diversity of methods in many fields, and is used to develop commercial processes in practically all industries (Url-3). Meanwhile, many problems evidently occurring in various fields can be modelled and resolved through mathematical programming techniques, optimization has involved large amount of attention from the research community (Horst & Pardalos, 2002).

In management science, mathematical optimization is the selection of the best element from some set of presented options with the most cost effective or highest achievable performance under the given constraints, by maximizing desired factors and minimizing undesired ones (Colburn & James, 2013).

2.2 Critical Path Method (CPM)

The critical path method (CPM) or Critical Path Analysis is a mathematically based algorithm for planning a set of project activities (Kelley J. E., 1961). Scheduling can be described as the evaluation of timing and sequence of a project's actions, which enables defining of the overall completion date (Mubarak, 2010).

As one of the most prevalent scheduling and network analysis methods, the critical path method (CPM), implicates determination of the longest path within the network. It facilitates defining a project's duration in conjunction with the shortest amount of time required to complete the project (Kerzner, 2013). According to (Samuel L. Baker, 2004) and (Armstrong & MICE, 1969) the crucial approach for using the CPM method is building a model of the project that contains the following:

- It is necessary to have a list of all activities to complete the project,
- The time (duration) which each activity needs to be completed,
- The relationship (dependencies) among the activities and,
- Logical end points like targets or deliverable items.

Implementation of the upper-mentioned principles, Critical Path Method calculates the longest path of activities to the end of the project's schedule. Each activity's earliest and latest point can start and finish without making the project longer. This method resolves which activities are on the longest path (critical) and which have can be delayed without making the project longer (total float). In project management and engineering, a critical path is order of project network activities that sum up to the longest total period. This method concludes the shortest feasible time for project completion (Santiago & Magallon, 2009). In other words, critical path is the sequential activities from start to the end of a project. Every suspension and delay of an activity on the critical path directly influences the due date, which is a planned project completion date (i.e. there is no float on the critical path). It should be considered that a project could have many, parallel, and near critical paths. Essentially every construction project has an overall completion deadline determined by the client that is specified in the contract. Some of deadlines, which are assigned in contracts, are not acceptable for any delays in the completion of the project (Halpin, 2010).

In spite of finding the earliest possible completion date appointed through network evaluation and exploration specified by the Critical Path Method, the due date might also be targeted on a later date. According to (Aminbakhs, 2013) several of these limits are appointed severely not permitting for any delays in the accomplishment of the project. Consequently, in scheduling stage of a construction project, the process should be time-limited, if the due date is considered as the main objective. The main purpose in this circumstance is to make sure that the project to be accomplished on the specific date.

2.3 Optimization Problems and Their Classification

Optimization, also has been named mathematical programming which assists to find the solution that yields the best result, the one that reaches the highest profit, output, or delight and, the one that achieves the lowest cost. Most of the times these problems associate with the most effective use of resources containing money, time, equipment, staff, and more (Upreti, 2009). Optimization problems are frequently categorized as linear or nonlinear. It depends on whether the relationship with the problem is linear with regard to the variables (Lingo Software, 2010). There has been an attempt to depict complicated human and social situations. So, the problems should be inscribed in a mathematical expression, including one or more variables. The question then arises that, what values should these variables have, in order to guarantee the mathematical expression has the highest possible numerical value (maximization) or the least possible numerical value (minimization). This procedure of maximizing or minimizing is referred to as optimization (Upreti, 2009).

Mathematical Programming resolves the problem of controlling the optimal distributions of limited resources required to adopt a given objective. The objective must symbolize the goal of the decision-maker. For instance, the resources may correspond to people, resources, money, or land. It is preferred to find the one or ones that maximize or minimize various numerical quantities like profit or cost. Since they try to find the best possible approach for decision-makers, optimization models are also called Prescriptive or Normative models (Olarenwaju, Daniel, & Segun, 2014).

Many optimization algorithms are available. However, some approaches are only suitable for specific types of problems. It is vital to recognize the characteristics of the problem and classify a suitable solution method. Each problem has different minimization methods, which differ in computational requisites, convergence properties. Optimization problems are categorized due to the mathematical characteristics of the objective function, the constraints, and the convenient decision variables (Arsham, 2014).

In the simulation-driven optimization and modelling, there are three main issues: the algorithm's effectiveness, and the efficiency and accuracy of a numerical simulator, and assign the right algorithms to the right problem. Despite their importance, there is no satisfactory rule or guidelines for that kind of issues. Obviously, the attempt is to implement the most efficient algorithms available, but the actual efficiency of an algorithm may depend on many factors (Yang X. S., 2013).

There are three basic ingredients in Optimization problems (Arsham, 2014):

The quantity we want to maximize or minimize is called the objective function, which we want to minimize or maximize it. Most optimization problems may have a single objective function, if they have more, they can be reformulated. The two remarkable exceptions to this rule are:

The goal-seeking problem: In most business applications, the administrator desires to attain a specific goal, while fulfilling the constraints of the model. The user does not principally want to optimize anything. So, there is no aim to define an objective function. This type of problem is typically entitled a feasibility problem.

Multiple objective functions: which most of the time, the user would like to optimize several various objectives at once. Typically, the dissimilar objectives are not compatible. Optimizing one objective of variables may be far from optimal for the others. In usual procedure, problems with multiple objectives could be reformulated as single-objective problems by either forming a weighted combination of the different objectives or likewise by employing some objectives as "desirable" constraints.

The manageable inputs considered as the set of decision variables, which influence the value of the objective function. In the engineering problems, the variables might include the distribution of different available resources, or the labor spent on each activity. Decision variables are essential. Consequently, we cannot define the objective function and the problem constraints, if there are no variables.

Parameters are called the uncontrollable inputs. The input values related to the particular problem might be fixed numbers. These values called, parameters of the model. There might be numerous "cases" or variations of the similar problem to resolve, and in each problem variation, the parameter values will be exchanged.

The relations between decision variables and the parameters are constraints. A set of constraints admits some of the decision variables to take on certain values, and eliminate others. For the industrial problem, it does not make sense to spend an undesirable amount of time on any activity; therefore all the "time" variables considered to be non-negative. Constraints are not necessary constantly. Indeed, the field of unconstrained optimization is a great and important and there are a lot of algorithms and software available.

2.3.1 Feasible and optimal solutions

Feasible solution can be defined a solution value for the decision variables, where all of the constraints are fulfilled. The proceeds of most solution algorithms start by first finding a feasible solution and then looking for improvement upon, and lastly altering the decision variables to transfer from one feasible solution to another feasible solution. This process is repetitive until the objective function has obtained its maximum or minimum. This result is named an optimal solution (Upreti, 2009).

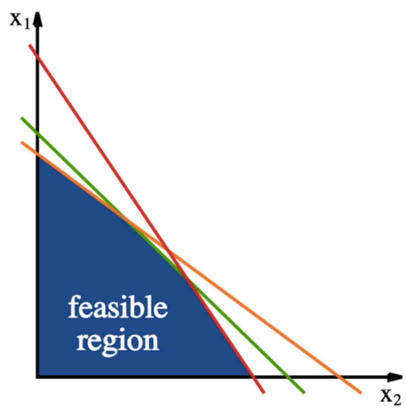

Figure 2.1 : Representation of a feasible region.

2.3.2 Linear programming

Linear programming is a technique that deals with some optimization problems, which objective function has to be optimized while the entire constraints are linear in terms of the decision variables. Moreover, all interactions between the variables related to resources are linear. In the late 1940's a linear problem was first formulated and resolved (Klotz & Newman, 2013). In mathematics, linear programming (LP) is a practice for optimization of a linear objective function, subject to linear equivalence and linear inequality constraints. Usually, linear programming establishes the technique to accomplish the best result (such as maximum efficiency or lowest cost) in a given mathematical model and given a certain list of necessities represented as linear equations. More officially, given a polytope (for example, a polygon or polyhedron), and a real-valued function assigning finite values of finite quantities, eq. (2.1) (Harshbarger, 2014).

$$f\left(x_1, x_2, \dots, x_n\right) = c_1x_1 + c_2x_2 + \cdots + c_nx_n + d \qquad \textbf{(2.1)}$$

Linear programming has been known as an enormously formidable tool, in modelling real-world problems as well as a broadly applicable mathematical philosophy. However, several impressive optimization problems are nonlinear. The analysis of such difficulties includes a various mixture of linear algebra, multivariate calculus, numerical analysis, and computing techniques. Essential regions contain the design of computational algorithms (containing interior point methods for linear programming), the geometry and examination of convex sets and functions, and the study of structured problems particularly. Nonlinear optimization delivers primary insights into mathematical analysis and is extensively used in a variety of fields such as engineering design, geophysical exploration, economics regression analysis, and inventory control. When the optimization problem has a linear objective function and also its constraints linear Programming will be applied. The most accepted method is the Simplex Algorithm wherever first the inequalities are converted into equalities by adding slack variables and afterward incorporating and excluding the base variables until the optimum is found. Although for many instances, its worst case of computational complication is exponential that it is found to work quite effectively. Several parametric models for portfolio assortment consequently favor linear risk measures (accepting that these risk measures have less desirable properties than the variance) (Kangas, Kangas, & Kurttila, 2008).

2.3.3 Convex program

Many engineering problems can be formulated as linearly constrained convex program problems. Constraint Satisfaction problems are great in size and in most cases include transcendental functions. They are broadly involved in chemical practices and cost restrictions modelling and optimization (Kalouptisdis, 1997).

2.3.4 Separable program

Separable Program (SP) incorporates a unique case of convex programs, where the objective function and the constraints are separable functions, i.e., each term involves just a single variable. Separable programming is important since it lets a convex nonlinear program to be approximated with arbitrary accuracy with a linear programming model. The idea is to exchange each nonlinear function with a piecewise linear estimation. Separable problems appear often in practice, especially for time-dependent optimization (Jensen & Bard, 2002).

2.3.5 Heuristic optimization

Heuristic methods are developed for solving complex problems in a reasonable amount of time (polynomial order) as opposed to the exhaustive search of mathematical methods in which they might be slow to make progress. The obtained solution among all the actual solutions to this problem may not be the best of, or it may simply estimate the exact solution. However, using it is valuable for finding a solution, which does not require long time unreasonably. Heuristics are systems to explore to find optimal or near-optimal solutions (Murty, 2003).

It should be noted that heuristic algorithms do not guarantee therefore, the optimality heuristic arguments are used to demonstrate whatever we might later try to prove, or what we might presume to find in a computer run. They are, at best, educated guesses (Daly & Wilson, 1988).

2.3.6 Search techniques

Search is an expression used for developing/improving solutions to gain the optimum or near optimum. Search methods might be deterministic (constantly reach at the same final solution within the same order of solutions, although they may be contingent on the preliminary solution). Search methods might be stochastic where the solutions considered and their order is different depending on random variables. Examples are simulated annealing, ant colony optimization and genetic algorithms. Considering that search methods might be local, they find the nearest optimum, which may not be the real optimum. Instance: greedy heuristic (local optimizers).

Considering that search methods might be global, the true optimum would be found even if it engages affecting to worst solutions during search (non-greedy) (Murty, 2003).

2.3.7 Global optimization (GO)

The purpose of Global Optimization (GO) is to catch the elite solution of decision models, in existence of the multiple local solutions. In other words, the global optimization used to seek the globally best solution of models, in the potentiality of multiple local optima. Unconstrained optimization looks for the global maximum or minimum of a function over its whole searching area, exclusive of any restrictions on decision variables, whereas optimization is handling to find the optimum of the objective function subject to constraints on its decision variables, Presented global optimization approaches can be categorized as either deterministic or stochastic. (Horst & Pardalos, 2002)

2.3.8 Multi-objective program (MP)

Multi-objective Program (MP) also recognized as Multi-objective optimization, vector optimization, multi-criteria optimization, multi-attribute optimization or Pareto optimization is an area of multiple criteria decision making, that is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. In solving multi-objective optimization problems, some of the objectives as constraints may be satisfied through the additional objectives can be weighted to make a combined single objective function. Multiple-objective optimizations are different from the single objective instance, in some ways (Arsham, 2014):

Optimizing the entire objectives simultaneously is impractical generally. That is because the general concept of the optimum makes no sense in the multiple objective examples. Alternatively, a search for a feasible solution is launched accommodating the best compromise including objectives on a set of effective solutions;

To identify the best compromise solution demands the consideration of preferences expressed by decision-maker;

The multiple objective problems, which confronted in real-life situations, are generally mathematical functions of contrasting forms (Arsham, 2014).

In a goal-programming model a crucial element is the achievement function, which is the function that compute the level of minimization of the undesirable deviation variables of the goals reflected in the model.

For example, forecasting the cardholder's spending behaviour in credit card portfolio management is an important factor to decrease the risk of bankruptcy in business application example (Arsham, 2014).

2.3.8.1 Basic concepts of multi-objective optimization

The general multi-objective optimization problem, in a minimization case, can be formulated as follows, eq. (2.2) (Caramia & Dell'Olmo, 2008).

$$\mathrm{Min} f(x) = \left\{ f_1(x), f_2(x), \ \ldots, f_j(x) \right\} \qquad \textbf{(2.2)}$$

$$\text{s.t.} \qquad x \in X$$

Where the integer $J \geq 2$ is the number of objective, $x = \{x_1, x_2, \ldots, x_n\}$ is the vector demonstrating the decision variables, and X is the set of possible solutions. A multi-objective optimization problem generally does not have an exclusive optimal solution, however a set of solutions that are known as the Pareto-optimal set.

Pareto-optimal solution represents a settlement between different objectives, and the component of the corresponding vector of objectives cannot all be enhanced simultaneously. In multi-objective optimization problems, analysing two solutions is more complicated than in the single-objective optimization problems.

2.3.8.2 Pareto optimality

Pareto optimality explains a subject of area that resources have distributed in a way, that it is not feasible to improve a single individual without also affecting at least one other individual to become worse off than before the change. The studies of Pareto optimality also have a substantial impression in the field of engineering (Url-1).

2.3.8.3 Pareto improvement

Pareto improvement is related to the idea of Pareto optimality. A Pareto improvement is an expression used when considering the distribution of things within a system from the perspective of Pareto optimality. A Pareto improvement is assumed to have taken place if an alteration has been made in the distribution of resources those consequences in at least one individual being improved than before the alteration while not making any other individual worse. In other words, describing Pareto optimality is to explain any state as Pareto optimal while there is not any possibility for Pareto improvement. This efficiently illustrates that it not possible to improve the state of any single individual exclusive of harming the condition of another individual (Url-1).

2.3.9 Combinatorial optimization

Combinatorial optimization mostly means finding an optimal object from a discrete set of objects. This area could be finite or countable sets. For instance, the discrete problem considered as combinatorial optimization. Applying “numerical” approaches can regularly solve problems by representing them to the integers where the state space is entirely well organized. This indicates the importance of the heuristics methods, such as simulated annealing. Combinatorial optimization is the analysis of sorting, covering, and apportioning, which are uses of integer programs. The analyses of finite or countable discrete structures are the principle mathematical topics in the interface between combinatorial and optimization.

According to the complication of known algorithms, these problems deal with the categorization of integer programming problems and the proposal of decent algorithms for solving special domain. It has key applications in some fields, and the traveling salesman problem (TSP) is an example of common problems concerning combinatorial optimization (Arsham, 2014).

2.3.10 Evolutionary techniques

Evolutionary techniques, practices mechanisms influenced by biological evolution, such as reproduction, mutation, recombination (crossover), and selection. Since the nature considered as a robust optimizer, examining and determining nature's optimization structure assists to find adequate solution systems for difficult problems. Both simulated annealing and the genetic algorithm are two impressions that have most promise in this regard. Development and scheduling are among the most effective applications of evolutionary techniques. An extremely effective tool for solving hard optimization problems, Genetic Algorithms (GAs) has become a highly efficient mechanism for solving hard optimization problems (Arsham, 2014).

2.3.11 Ant colony optimization

Ant colony optimization (ACO) is method for solving computational problems that can find good paths through graphs. It is the most effective example of artificial engineering swarm intelligence system with several applications to everyday problems. Marco Dorigo proposed it in 1992 in his PhD thesis aiming to search for an optimal path in a graph (Dorigo, 1992). His thesis was based on the behavior of ants looking for a path between their colony and a source of food.

2.3.12 Metaheuristics

A To find an adequate and good solution to an optimization problem, particularly with incomplete or insufficient information or limited computation facility a metaheuristic procedure or heuristic methods used to find and provide a sufficiently good solution to an optimization problem (Leonora et al., 2009). It should be noted that metaheuristic methods in comparison with optimization algorithms and iterative methods do not guarantee to find a globally optimal solution for some class of problems. However, metaheuristics might build some assumptions approaching the optimization problem to solve, and consequently they might be practical for many of problems (Blum & Roli, 2003).

Metaheuristic is repetitive generation process which leads a dependent heuristic by mixing cleverly various concepts derived from traditional and classical heuristics, biological evolution, artificial intelligence, natural and physical sciences for exploring the search areas exploiting learning strategies to build information in order to discover effectively near-optimal solutions (Osman & Kelly, 1996). Most meta-heuristics have been built for resolving discrete combinatorial optimization problems (Solnon & Bridge, 2005). The applications that applied in engineering, nevertheless, typically require methods, which engage continuous variables, or discrete variables. The overall goal of metaheuristics recognized to alter discrete domains of application into continuous ones, in the manner of: Methodological developments intended at adapting several meta-heuristics, especially Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithms (GA), and Ant Colony Optimization (ACO), variable neighborhood search, guided local search, scatter search, to continuous or discrete variable problems. Software applications and algorithms for metaheuristics adjusted to continuous optimization and real implementation of discrete metaheuristics modified to continuous optimization (Blum & Roli, 2003).

2.4 Evolutionary Algorithms

Evolutionary algorithm concerns the attitudes of evolution originate in nature to find an optimal solution for a computer-based problem solving systems, which use computational examples of evolutionary practises as fundamental elements in their design and application. Specifically, evolutionary algorithm’s structure implements a structure that improves, according to selection systems such as recombination and mutation that are referred to as "search operators" (or genetic operators). Each individual in the population obtains a degree of its fitness in the environment. Reproduction in EAs keeps focusing attention on high fitness individuals, accordingly developing the available fitness information. Recombination and mutation unsettle the individuals and propose wide-ranging heuristics for exploration. Considering biologist's point of view, evolutionary algorithms are sufficiently complex to deliver robust and powerful adaptive search systems (Spears et al., 1993).

Problems including multiple objectives occur indeed in many cases nowadays and demanding their solution for researchers has been a challenge for a long time. In spite of the extensive diversity of methods established in Operations Research (OR) to deal with these problems, the difficulties of their solution demands for alternative approaches. Since EAs have population-based, which helps to generate many elements of the Pareto optimal set in a single run, motivated many to solve problems of this nature. Moreover, the complication of some multi-objective optimization problems may prevent the use of traditional operational research multi-objective optimization problems solution techniques (Coello, Lamont, & Veldhuizen, 2007).

In solving a problem, we are mostly seeking for a solution, the solution that is the best among others. The area containing possible solutions, in other words, objects that are among the desired solution called search space. Each point belongs to one feasible solution in the search area and can be measured by fitness for the problem or its value.

Looking for a solution in search space is identical to a seeking for some extreme (maximum or minimum). Generally, by the time of solving a problem, just a few points of the search space is known and other points can be generated until the process of finding solution continues. However, the search can be very complicated. The important part of solving a problem is where to look for the answer and where to start. There are several approaches to find some appropriate solution (i.e. not necessarily the best solution). The hill climbing, tabu search, simulated annealing and genetic algorithm are the best-known instances of this matter. The solution discovered by these techniques is often debated as a good solution, because it is not possible to prove what is the real optimum normally (Url-2).

2.5 NP-Hard Problems

The decision problems in extreme complexity class (computational problems with the same bound such as time and space), which are inherently harder than those that can be solved by a nondeterministic in polynomial time, are called NP-hard problems. Difficult problems, which cannot be solved in "traditional" way, are examples of the NP-hard problems. Some problems are not possible to be solved algorithmically. Moreover, some problems were ascertained that they are not solvable in polynomial time. However, despite being difficult to find a solution in many important tasks, it is easy to check the solution once we have it. This point led to NP-complete problems. NP is the abbreviation of the word nondeterministic polynomial. Nondeterministic polynomial means that it is probable to "guess" the solution (by some nondeterministic algorithm) and then check it, both in polynomial time. We would be able to find a solution in some reasonable time if we had a machine that could guess. The reason for examining the NP-complete problems is for the simplicity limited to the problems, where there is one answer: “yes” or “no”. Therefor NP-hard problems have been introduced in order to deal with tasks with complicated outputs. Classification of NP- hard problems according to their approximability to optimization problems is another important endeavor. Each NP-complete problem can be unravelled by exhaustive search. Unfortunately, when the size of the cases rises the running time for exhaustive search, soon becomes severely large, even for examples of moderately small size (Woeginger, 2003). Nobody knows if some faster exact algorithm exists nowadays. Proving or disproving the existence of the availability of faster algorithms, remains as a big mission for new researchers. Since many people believe in not existence of such an algorithm, they are looking for some alternative techniques. For instance, a genetic algorithm is in the category of these approaches (Princeton, 1998).

2.6 Genetic Algorithm

The fundamental theme of all evolutionary algorithms is the use of Darwin's theory of evolution, which can be considered by the major components mutation, crossover, recombination, selection (Fogarty, 1996). Genetic Algorithm includes a population, which is a set of solutions exhibited by chromosomes. Solution is taken from one population and implemented to construct a new population. This encouraged by hope that the new constructed population be better than the previous one. In order to select new solutions, offspring are selected from new population with regard to their fitness function. The more appropriate offspring, the more chances they have to reproduce. This is continual until some condition (for instance number of populations or improvement of the best solution) is fulfilled.

In the 1960s Rechenberg presented idea of evolutionary computing in his research "Evolution strategies". Genetic Algorithms (GAs) developed by John Holland and his students and colleagues. This subject studied examining on the natural genetic selection mechanism and adaptive progressions of natural systems, and then he targets to design artificial systems software established on important mechanisms of natural systems. (Negnevitsky, 2005). Genetic algorithms are considered as search algorithms, which are established on the procedure of natural mechanism and natural genetics (Goldberg D. E., 1989). In this technique, strong species that have more chance to survive and continue for next generation duplicate to the next generation and the chromosomes with the least value of fitness function removed throughout this procedure. By process of selection, crossover and mutation new chromosomes are constructed in which they inherit decent properties of their parents, so the average quality of solution favors to get better individuals as the generation's development. The algorithm presumes to discontinue when some predetermined principles are met (Myung & Kim, 1996). These techniques also use the genetic mutations techniques, which the basis stimulated from the nature in order to search exterior of the current region of variables space, and it additionally prevents the fast convergence of the algorithm until to reach to the popular solution. The Genetic Algorithm is also capable to overcome to the restrictive assumptions, is broadly applied in various situations of complex problem such as engineering, manufacturing scheduling, business planning, mathematics. Various analyses and surveys confirm the efficiency of genetic algorithm for NP-hard problems (Goldberg, 1989).

2.6.1 Chromosome

All living creatures consisted of cells. There is the identical set of chromosomes in each cell. Chromosomes are sequences of DNA strings and perform as a model for the entire organism. Genes and blocks of DNA are a part of a chromosome. Each gene encrypts a specific protein. Essentially, each gene encodes a trait, for instance color of eyes. Every single gene has its particular position in the chromosome that is called locus. All chromosomes (complete set of genetic material) are named genome.

2.6.2 Initial population

The first stage of the implementation of a GA is the generation of an initial population. Each member of this population instructs a potential solution to a problem (Diaz-Gomez & Hougen, 2007). Population is described as a group of chromosomes that GA starts searching procedure based on a cost function of each chromosome. The population size has a significant effect on the GA's performance. Generally the population is described by two parameters which N_{pop} represents the number of chromosomes and N_{bit} illustrates a number of gens in a chromosome (Haupt & Haupt, 1998). In spite of providing more variety of potential solutions by a large population and leading an algorithm to perform more informed searches, it possibly reduces the algorithm convergence due to numerous required assessment for each generation. With a population size in the range of 30-100, the best on-line performance can be achieved (Grefenstette, 1986).

2.6.3 Genetic algorithm operators

The Genetic Algorithm operators are mainly as follows (G K Joshi, 2008):

1. Selection
2. Crossover
3. Mutation

The population, constructed with collection of chromosomes, which each string is encoded the problem solution as a finite-length of gens. As demonstrated in Fig (2.2), the GA procedure starts with defining cost function and decision variables of the problem. The initial population is generated and the corresponding cost for all of the chromosomes should be computed in the following and the best individuals are selected for mating. Mutation also helps GA to seek further problem space to prevent a premature convergence, which is a relatively common problem in Evolutionary algorithms. In the last step, the algorithm goes for checking the convergence, whether the stopping criteria has been satisfied or not. Although several stopping criteria can be defined based on problem requirements, the most popular stopping criteria in GA can be defined based on the number of generations, time limitation, or convergence checking. A flowchart through the components of the simple GA is shown Fig. (2.2).

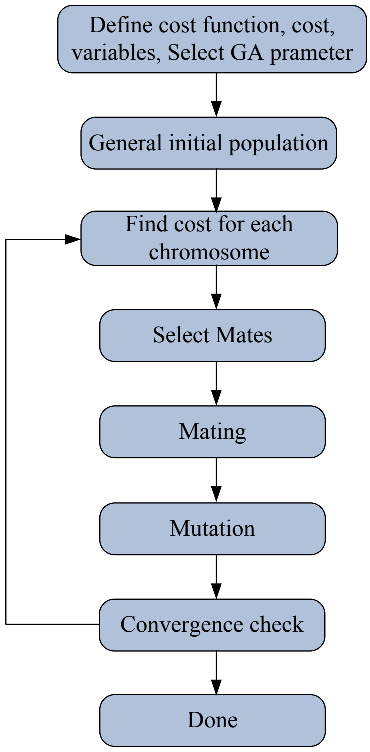

Figure 2.2 : Flowchart of simple GA procedure.

2.6.4 Other parameters

Some other parameters of GA are also available. Population size is one additional important parameter. Population size indicates the number of chromosomes in one generation of a population. In case of availability of too few chromosomes, consequently, genetic algorithm has a few possibilities to accomplish crossover and just a narrow region of the search space is explored. Additionally, in case of availability of too many chromosomes, the computation of GA takes a long time. Research confirms that regarding certain threshold (which depends mostly on encoding and the problem) increasing the population size it is not useful, because it doesn't solve the problem faster (Kumar et al., 2010).

2.6.5 Crossover operator

In Genetic Algorithm, crossover is considered as the most essential operator. The procedure contains creating strings from the provided information within two (or more) parents solution.

The objectives of all crossover operators in all forms of the GA classified as binary, continues, and permutation. GA algorithms are transferring the information contained in the parents as much as possible (Eiben & Smith, 2003). It is regularly expressed as crossover, recombination or mating operators. A high crossover rate with a low mutation rate can deliver effective performance of the algorithm with population size (20 to 40), which is classified in the small population group. In the mid-sized populations, which the population ranges from 30 to 90, the optimal crossover rate decreases as the population size increases. The best crossover rate is obtained by 0.50 of the population with size of 50 and the optimal value of the crossover rate decrease to 0.30 as the population size increase to 80 (Grefenstette, 1986). Although several approaches of permutation crossover operator have been recommended.

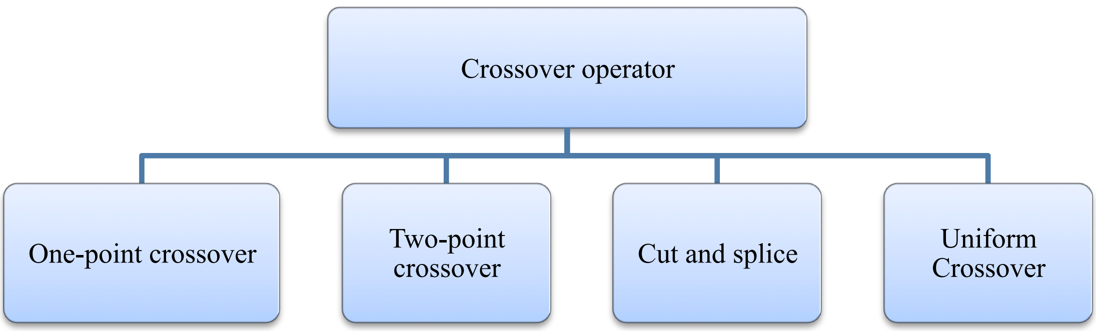

Figure 2.3 : The most important crossover operators.

Fig. (2.3) shows the most important crossover operators. Initially, crossover (recombination) happens during reproduction. Genes from parents contribute in constructing whole new chromosome. The produced new offspring can be mutated then. In the mutation process, some of the elements of DNA are exchanging slightly. These changes are mostly caused by inaccuracies in replication genes from parents. The measurement of the fitness of an organism is the consequences of the organism's success in its life (Mahalakshmi, Kalaivani, & Nesamalar, 2013).

2.6.5.1 One point crossover

In a single crossover point, crossover point on both parents' organism strings is selected. All data beyond that point in either organism string is swapped between the two parent organisms. The experimental study depicts that one point crossover represents the best results over the wide range of the problems comparing to other crossover operators. Meanwhile, in this research one point crossover has been utilized (Magalhães-Mendes, 2013).

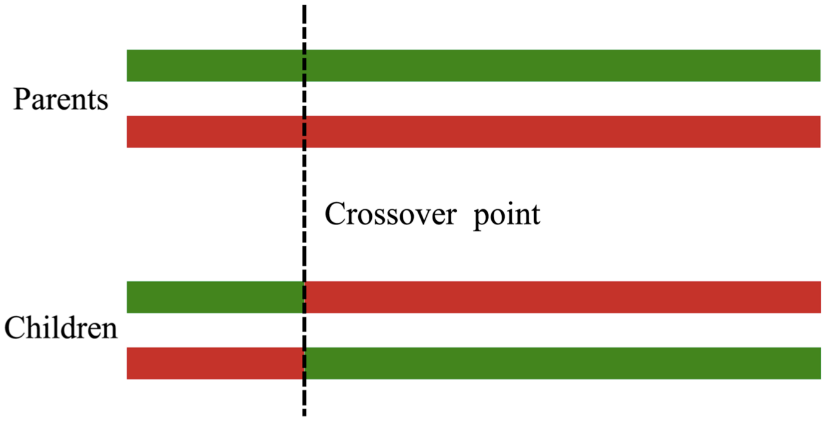

Figure 2.4 : One point crossover operator.

2.6.5.2 Two point crossover

In two-point crossover, two points have to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two child organisms Fig 2.5 (Magalhães-Mendes, 2013).

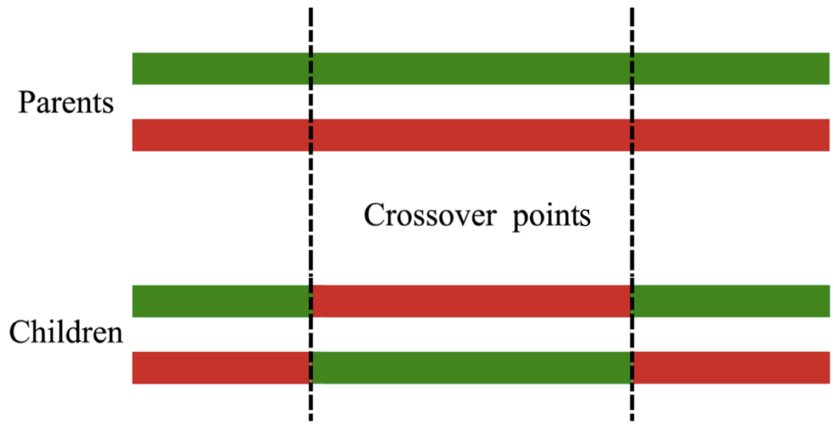

Figure 2.5 : Two point crossover operator.

2.6.5.3 Cut and splice crossover

Another crossover variant, the "cut and splice" approach, results in a change in length of the children strings, Fig 2.6. The reason for this difference is that each parent string has a separate choice of crossover point (Chen, Jiang, Li, & Shushen, 2013).

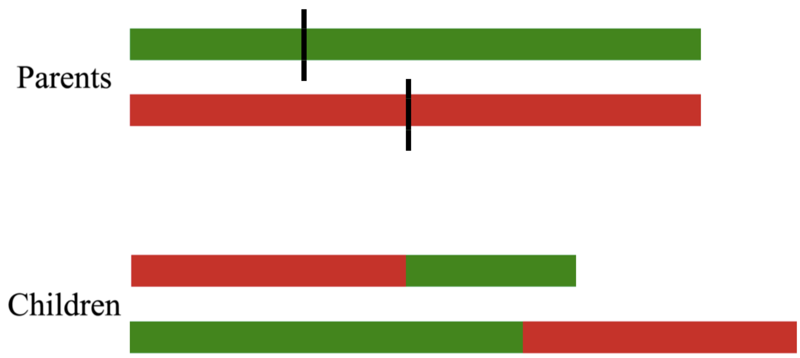

Figure 2.6 : Cut and splice crossover operator.

2.6.5.4 Uniform crossover

The Uniform Crossover uses a fixed mixing ratio between two parents. Unlike one- and two-point crossover, the Uniform Crossover enables the parent chromosomes to contribute the gene level rather than the segment level. If the mixing ratio is 0.5, the offspring has approximately half of the genes from first parent and the other half from second parent, although cross over points can be randomly chosen as shown in Fig. (2.7) (Syswerda, 1989).

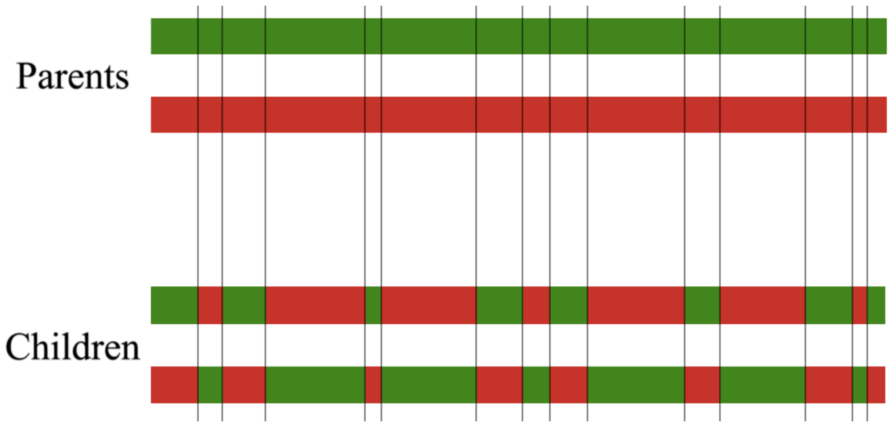

Figure 2.7 : Uniform crossover operator.

2.6.6 Mutation

After performing a crossover, mutation occurs which avoids sticking all solutions (in population) into a local optimum of solving problem. The new offspring alters randomly by mutation.

Mutation alters randomly the new offspring. Some of randomly elected bits can be transferred from 1 to 0 or from 0 to 1 in a binary encoding. By implementing mutation operator, only one parent is used to generate one offspring by applying some random changes in genotype. The fundamental goal of mutation operator is making an algorithm to search further feasible problem space by preventing the rapid convergence or premature convergence of the algorithm to the solution (Eiben & Smith, 2003).

2.6.6.1 Mutation rate

Mutation rate is described as probability of a mutation happening in an individual in the population. Mutation rate is achieved by multiplying the mutation factor to the number of gens in each generation. Although mutations can have favourable and unfavorable effect on the individuals, however the changes are mostly unfavourable and lead to the string damage. Mutation rate is usually selected to be less than 5% of population. Mutation rates above 5% are usually harmful and provide a high level of distraction for the algorithm (Grefenstette, 1986). In order to prevent the loss of the current fittest member of the population due to disruption, the best chromosome is kept and mutation is applied to the rest of population. In this case the total number of mutation is given by, eg. (2.3).

$$\text{Mutation.Num} = \mu * (\text{Npop} - 1) * \text{Nbits} \tag{2.3}$$

N_{pop} indicates the number of population in each generation and N_{bits} denotes the total number of gens in one chromosome. μ is defined as mutation rate which illustrates the presumed distraction level for search algorithm (Haupt & Haupt, Practical genetic algorithms., 1998).

2.6.6.2 Point mutation

The classical instance of a mutation operator is the probability of the involvement of an arbitrary bit and changing its original state in a genetic sequence. The generated random variable indicates if the modification of a particular bit occurs. This mutation technique is named single point mutation, which is originated from the biological point mutation (Prebys, 2007).

2.6.6.3 Swap mutation

This operator randomly picks two gens in the chromosome and makes new offspring by swapping their positions, Fig. (2.8) (Misola & Navarro, 2002).

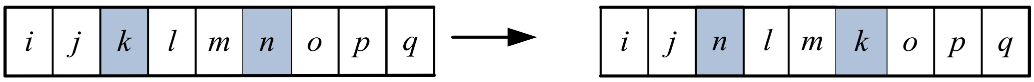

Figure 2.8 : Swap mutation sample.

2.6.6.4 Insert mutation

This operator starts by generating two random integers on the set {1…n} which n is defined as chromosome length. In the next step picks two positions and moves one so that it is next to other. The below figure simply represent the insert mutation method, Fig. (2.9) (Kusum Deep, 2011).

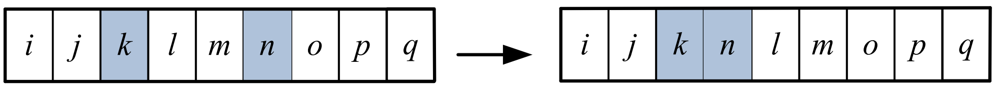

Figure 2.9 : Insert mutation sample.

2.6.6.5 Crossover and mutation probability

Crossover probability exhibits the number of times, which crossover will be performed. In case of not implementing the crossover operator, offspring is the same and exact duplication of parents. By implementation of crossover operator, offspring will be made from parts of parents' chromosome. If the probability of crossover is 100%, subsequently all offspring will be made by crossover. If its probability will be 0%, entire new generation will be made from exact duplications of chromosomes from previous population (however this does not indicate the sameness of new generation with the old one). The concept of using a crossover operator is hopefulness to have the new generated chromosomes inherit good segments of old chromosomes and possibly the new chromosomes will be better. However, it is worthy to permit some part of population survive to the next generation. By implementing the mutation in GA, sticking in local extreme will be prevented, because GA will in fact change into random search (Eiben & Smith, 2003).

2.6.7 Selection

In genetic algorithm, selection is the phase that genomes of an individual are selected from a population for next mating step (recombination or crossover). Essentially, we would like to maximize the fitness function, which is some measure of profit, value, or goodness. Strings with a higher value of the fitness function have a higher probability to be copied (survive for the next generation) to the next generation. Chromosomes are chosen from the population (to be parents) to crossover. The question is how to select these chromosomes. Each string probability to be selected depends on its fitness value. The selected strings construct a mating pool with the same size of the population (Yin , 1999). According to the principles of natural selection and the survival of the fittest individuals in Darwin's evolution theory, the best strings should survive and construct new offspring. Many methods are available to select the best chromosomes, for instance, tournament Selection, roulette wheel selection, rank selection, steady state selection, which are described in this chapter.

2.6.7.1 Tournament Selection

Tournament Selection is among the most popular used selection strategies in evolutionary algorithms that aim to imitate the natural competition of the species (Michalewicz, 1996). This method is very suitable for an extensive range of problems. In tournament selection, two individuals are randomly selected from the mating pool. The number of individuals in the mating pool (N_{pool}) depends on the values of the selection rate (μ) and population size N_{pop}, Eq. (2.3).

$$N_{pool} = \mu * N_{pop} \tag{2.4}$$

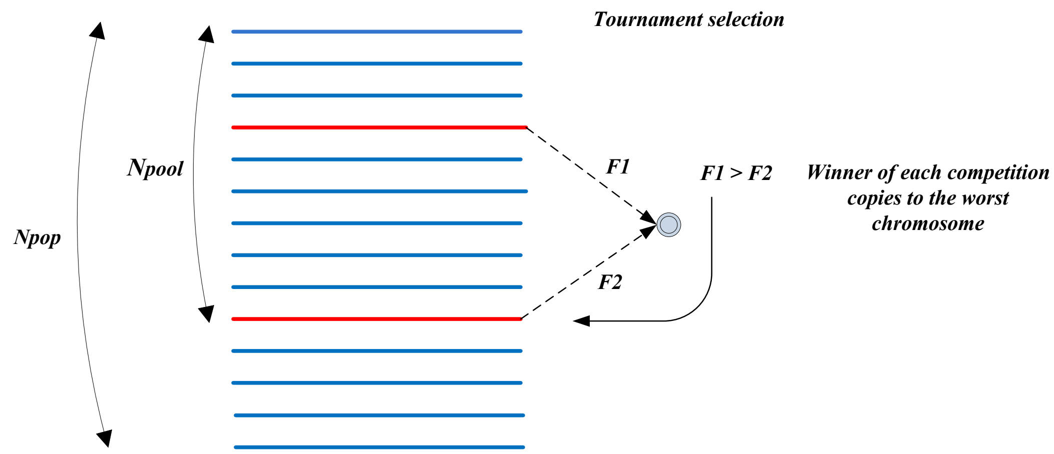

Figure 2.10 : Illustrates the tournament selection procedure.

Tournament selection comprises randomly selection of two individuals from the population, which the individual with the highest fitness value is selected as the winner of the tournament and the selection process continues by selecting a new tournament group randomly until all the individuals are selected. Tournament selection mechanism is represented in Fig (2.10).

```
BEGIN
  Set Tournament_selection=1;
  i = Tournament_selection;
   While (Tournament selection ≤ μ) DO
        Generate two random number between [1,Npop]
        Pick the related individuals
        Select the best individual by comparing their fitness functions
        Copy the best chromosome to the worse one
  i=i+1
END
```

Figure 2.11 : Pseudo code for the tournament selection algorithm.

As demonstrated in the figure 2.10, two individuals are randomly selected from population and finally the worst chromosomes are exchanged with tournament winner. In spite of the popularity and implementation of a Roulette wheel and Tournament selection in Genetic Algorithm, however, Tournament selection shows well performing for large population sizes because sorting becomes very time consuming because the population never needs to be sorted (Haupt & Haupt, 1998). Figure 2.11 shows the applied pseudo code for tournament selection mechanism in GA programming.

2.6.7.2 Roulette wheel

Parents are selected according to their fitness. The better the chromosomes are, the more chances they have to be selected. The selection mechanism is a very popular, which performs based on the probability of the corresponding fitness function to create the evolutionary process. Parents are selected according to their fitness. Those chromosomes with higher value of the fitness function have a higher chance of surviving for the next generation. Imagine all chromosomes in the population placed in a roulette wheel, each chromosome has its place with regard to its fitness function, as illustrated in Fig (2.12).

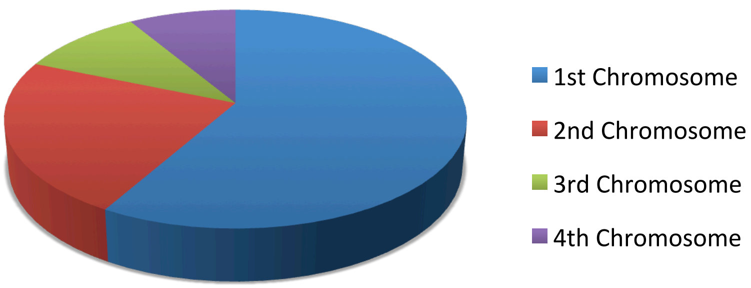

Figure 2.12 : Roulette wheel selection

The selection probability for each chromosome on the roulette-wheel can be computed as follows, Eq. (2.5) (Bäck & Michalewicz, 1997).

$$F_i = \frac{P_i}{\sum_{l=1}^{L} P_i} \qquad i = 1, \ldots L \qquad \textbf{(2.5)}$$

Where P_i represents the value of fitness function for i_{th} chromosome, since the whole probability of roulette wheel is computed by accumulation probability of all slices that is equal to 1 and each slice has a width corresponding to the individual's selection probability (Bäck & Michalewicz, 1997).

2.6.7.3 Rank selection

There are some problems in Roulette wheel selection when the fitness fluctuates considerably. For instance, some chromosomes will have very few chances to be selected if the entire roulette wheel (90% of it) contains the best chromosomes. However, Rank selection ranks the population initially and then each chromosome achieves fitness from this ranking. The best chromosome in rank selection will have N fitness, which indicates the number of chromosomes in a population. Fig. (2.13) and 2.14 illustrates how the situation of fitness before and after ranking (R.Sivaraj, 2011).

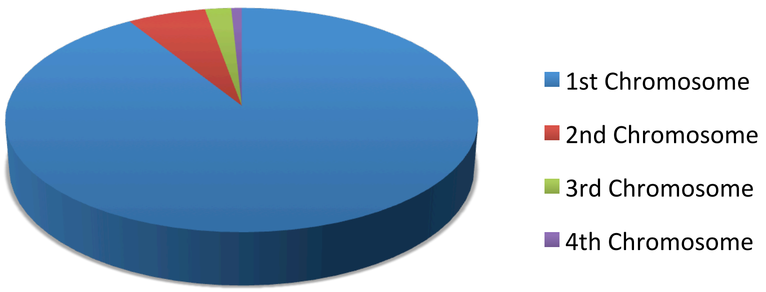

Figure 2.13 : Situation before ranking (graph of finesses).

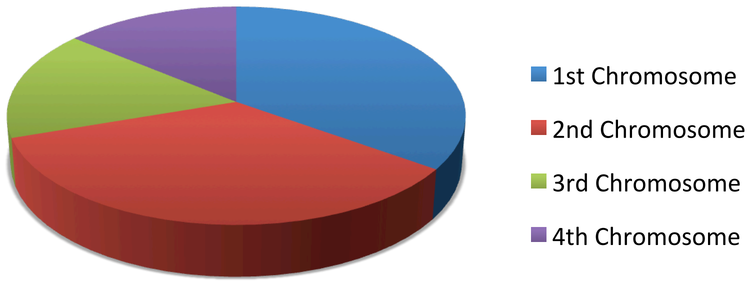

Figure 2.14 : Situation after ranking (graph of order numbers).

Subsequently, rank selection makes it possible for all of the chromosomes to have a chance to be selected. However, since the best chromosomes do not vary considerably from other ones, this technique can lead to slower convergence (R.Sivaraj, 2011).

2.6.7.4 Steady-state selection

The key idea of steady-state selection is outline of the big part of chromosomes to next the generation. This is not a specific technique of selecting parents. The steady-state selection approach in GA works in following system. For constructing a new offspring, few (good - with high fitness) chromosomes have to be selected. Formerly, constructed offspring is located in their place by removing some (bad - with low fitness) chromosomes and the remaining population survives to a new generation (Durillo et al., 2009).

2.6.8 Encoding

Finding an appropriate way to encode a chromosome is the most critical problem in implementing a genetic algorithm. Encoding extremely is contingent to the problem.

2.6.8.1 Binary encoding

The most common type of encoding is binary encoding. It was the first type of encoding which implemented in GA. Every chromosome in binary encoding has a string of bits, 0 or 1, Fig. (2.15) (Url-2).

Chromosome A	1	0	1	1	0	0	1	0	1	1	0	0	1	1	1	1	0
Chromosome B	1	1	1	1	1	0	0	0	0	0	1	1	0	1	0	1	1

Figure 2.15 : Chromosomes with binary encoding.

The original encoded alleles of GA are either zero or one, symbolized by a specific computer bit. In spite of the possibility of encoding the small number of alleles in chromosomes, the binary encoding is not probable for several problems, many corrections and adjustments should be made after crossover and / or mutation (Url-2).

2.6.8.2 Discrete encoding

Discrete encoding provides the possibility for chromosomes to take natural numbers. In discrete encoding all natural numbers in a chromosome demonstrate a solution concerning a problem, Fig (2.16) (Url-2).

Chromosome A	4	2	3	3	1	2	3	1	3	2	1	3	1	3	4	3	5
Chromosome B	4	4	3	3	1	2	1	3	3	3	1	1	4	1	4	1	1

Figure 2.16 : Chromosomes with discrete encoding.

2.6.8.3 Permutation encoding

In ordering problems permutation encoding can be implemented. For instance, travelling salesman problem or task ordering problem. Each chromosome in permutation encoding, consist of a string of numbers, which demonstrates a number in a sequence, Fig (2.17) (Url-2)

Chromosome A	1	5	3	2	6	4	7	9	8
Chromosome B	8	5	6	7	2	3	1	4	9

Figure 2.17 : Chromosomes with permutation encoding.

Permutation encoding is very convenient for ordering problems. However, for some types of problems, crossover and mutation operator should be corrected in order to keep the chromosome consistent (i.e. have a real sequence in it).

2.6.8.4 Value encoding

Value encoding can be implemented when some complex values like real numbers has applied. In this case, using a binary encoding method for this kind of problems will be extremely difficult. Each chromosome in a value encoding is a string of some values, Fig. (2.18) (Url-2).

Chromosome A	1.2324	5.3243	0.4556	2.3293	2.4545
Chromosome B	ABDJE	IFJDH	DIERJ	FDLDF	LFEGT
Chromosome C	(Back)	(Back)	(Right)	(Forward)	(Left)

Figure 2.18 : Chromosomes with value encoding.

2.6.9 Elitism

After crossover or mutation operators phase while creating a new population in GA, we can lose good candidates, and its outcome is the offspring that are weaker than the parents which might leads to losing the best chromosome. The Elitism is used to avoid the loss of the best chromosome crossover or mutation phase and keep the best individual from generation to generation (Haupt & Haupt, 1998). In this pattern, the best individual in the population is copied to the following and the next generation that it constantly attempts to preserve the best individual in each generation.

In other words, Elitism is term of technique, which keeps the best chromosome (or a few elite chromosomes) and copies it to a new population.

Elitism will precisely enhance the performance of Genetic Algorithm since it prevents missing the best-found solution in each generation and consequently increase the convergence rate for large-scale problems. The elitism implementation mechanism is represented in Fig. (2.19) (Champion, 2009)

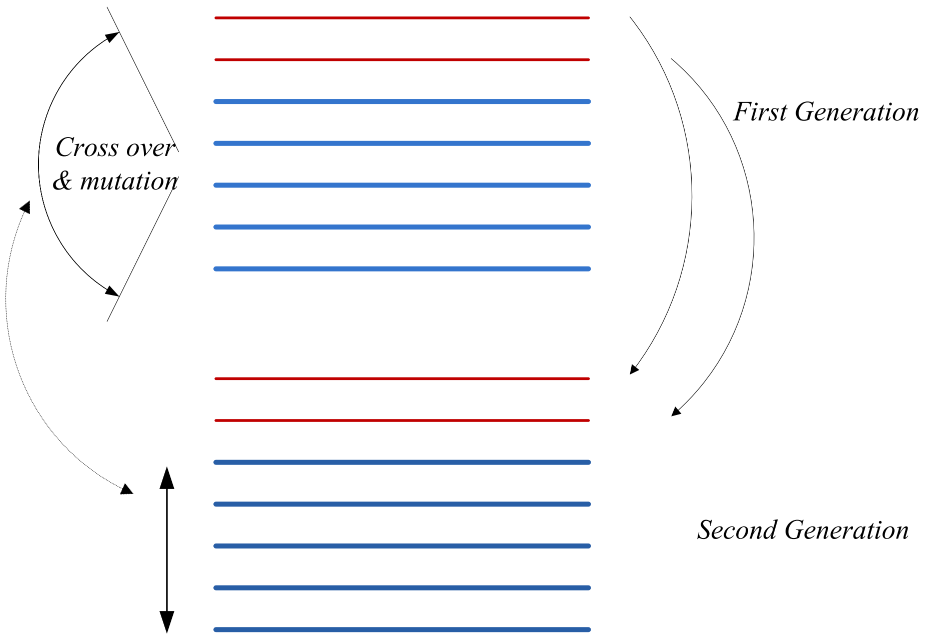

Figure 2.19 : Elitism implementation in Genetic programming.

2.6.10 Fitness function

The evaluation of the cost of the created chromosomes to measure the optimality of solutions and control the genetic operators during the solving procedures is the responsibility of the fitness function. Mostly, the applicable fitness function thoroughly associates with arithmetical objective function, which is capable of computing the cost for each chromosome quicker (Norozi et al., 2011)

2.7 Simulated Annealing

The simulated annealing algorithm originated from the procedure of annealing in metal work. Annealing involves heating and cooling a material to change its physical properties due to the variations in its interior structure. New structure of metal turns out to be stable as the metal cools down, subsequently, imitating the metal to keep its recently engaged properties. To simulate this heating process, we set a temperature variation. It has to be set primarily high and as the algorithm runs, allow it to cool down gradually.

While the appointed temperature variable is high, the simulated annealing algorithm will tolerate more density and admit solutions that are inferior to our existing solution. This provides the algorithm the capability to jump out of any local optimums in early stage of running of the program (if it sticks to any local optimums) and by gradually cooling the temperature down, it is probable to obtain worse solutions, so, it admits the algorithm to gradually concentrate on the search space and nearby enough to an area of an optimum solution. The Simulated Annealing Algorithm is outstandingly effective at estimating a near to the optimum solution while solving large problems, which contain numerous local optimums by using the gradually cooling process technique. The traveling salesman problem is a perfect example of its nature.

Simulated annealing algorithm is also well known as Mont Carlo annealing which works based on Monte Carlo methodology that could be used for simulating the behavior of a set of atoms. Simulated annealing is capable of handling noisy search space and complicated mathematical model such as the mixed-integer model, nonlinear models. The thermal equilibrium at a given temperature is getting through applying small random perturbation to the atomic structure. If this perturbation results in lower energy state, the algorithm is repeated by using new energy state. However, in case of achievement of higher energy state, a new state is recognized with a certain possibility, which it depends on the history of the search (Winston, 2003). The acceptance probability of worse move is called uphill probability. The uphill probability changes during the annealing process. The initial temperature must be sufficiently high to deliver appropriate degree of exploration towards its "freezing point" as the search progresses.

It is a consequence of getting the problem sufficiently large and the need to investigate through a vast amount of possible solutions to determine the optimal result. According to (Aarts & Laarhoven, 1987), since a search for an optimum requires prohibitive amounts of time for computation, in reality several wide-ranging combinatorial optimization problems find solution to reach optimality. Nevertheless, we cannot truthfully grant to discover the optimal solution within a realistic length of time. Therefore, we have to solve this kind of problems to get results that are adequately near the optimum solution. The best example in this area is the traveling salesman problem (TSP), the need to use an algorithm, which can find a decent enough solution in an acceptable amount of time, is necessary.

2.7.1 Advantages of simulated annealing

The simulated annealing algorithm reveals an outstanding performance at preventing from premature convergence and ending up to a local optimum. On average, it is much more practical approach (Kalai, 2006).

2.7.2 Neighborhood search structure

Neighborhood search structure (NSS) produces a new atomic construction by altering the candidate solution in direction to visit more potential solutions within the search space. The Neighborhood search structure is a SWAP operator in which randomly exchange position of two elements of the solution (Alizamir, Rebennack, & Pardalos, 2008). In our case, the contractors' potential solutions will randomly swapped by generating indicated possible contractors.

2.7.3 Cooling scheduling

In order to preclude algorithm to stick in a local optimum, worse moves might be accepted based on the current temperature, which is gradually, decreasing based on cooling schedule. The acceptance probability for worse moves decreasing as the algorithm proceeds. The exponential cooling scheduling is believed to be an appropriate cooling schedule for the SA because of the ability to compromise between fast schedule and also the ability to reach lower energy state. The exponential cooling schedule is given by where is current temperature and is the temperature decrease rate (Wang & Zheng, 2001).

2.8 A Hybrid GA-SA Algorithm

Simulated annealing and genetic algorithms are two very similar, natural motivated methods (Goldberg D. , 1990). The theory of both is derived from the nature. Both of them work fine on various problems and it is crucial to determine the fitness or cost of a solution for both algorithms. Both of them generate random solutions in the search space and attempt to jump from one solution to another probabilistically.

Both methods are not guaranteed to give optimal solutions. By implementing the annealing scheme to construct a satisfactory number of iterations, the convergence of a Simulated Annealing can be controlled. On the contrary, Genetic Algorithms do not possess such proof of convergence. Since the Genetic Algorithms are severely reliant on random operators, and they deal with populations and large number of individuals, GAs convergence cannot be easily controlled.

It is easy to parallelize Genetic Algorithms since they work on a population of individuals that can be assessed and processed in parallel (Sorokin et al., 2012). SA, on the contrary, operates on a single solution at a time. It moves from one solution to the next in a consecutive approach. Further more, it is possible to learn from previous searches in GA, since the population of individuals provide useful redundant information. To form high quality solutions and exploration of the search space, critical components of previous decent solutions can be captured and merged together via crossover. On the contrary, since SA runs on only single solution at a time and exploration is narrow to the immediate neighbourhood (Leung et al., 1998).

3. METHODOLOGY

This chapter covers definition of the problem of this study intending to determine the optimal subcontracting alternative plan.

3.1 Problem Description and Formulation

This study uses the formulas of a single-objective mathematical model for time–cost–quality trade-off problem, which involves a number of equality and inequality constraints (Mungle et al., 2013). Furthermore, penalty and incentive cost are explicitly considered in the formulation to determine the project cost. The proposed mathematical model is believed to mimic the practical time–cost–quality trade-off problem in the construction industry. Quality measurement approach is presented by using analytic hierarchy process (AHP) technique to evaluate an anticipated quality of the proposed work by subcontractors for each activity in the project. Finally, a hybrid GA-SA algorithm (GA-SA) is developed to solve the time–cost–quality trade-off problem.

According to (Mungle et al., 2013), The time–cost–quality trade-off problem in highway construction can be described as follows: Consider a general contractor to start planning of a highway construction project. The project consists of total N activities. Due to precedence relationships among those activities, project forms an activity network. Usually, contractors are not able to complete the whole projects by their own capacity and resources. Therefore, they put each of the various activities individually up for bids, and receive bids with both duration and cost from different subcontractors. For each time and cost option, contractors apply the AHP based on the information that they have about the subcontractor to evaluate the anticipated quality of the activity. Table 3.1 summarizes mathematical notations used in the proposed formulation in this study. Furthermore, it is a general practice to include the indirect cost (IC) per time period. The indirect cost (IC) comprises expenditure on management during the project implementation, which depends heavily upon the project duration, i.e., the longer the duration, the higher the indirect cost (IC). In real world, the project owner or government agency forces the contractor to finish the project within the due-date (D). If the project is tardy, the main contractor will receive the tardiness penalty of β for per tardy period. If the project is in the earliness, the main contractor will receive the incentive cost of I for per earliness period as a reward. For activity *i*, there are m_i qualified subcontractors responded to the tender invitation. Qualified subcontractor means their credit and quality meets the initial requirement of the project. For subcontractor *j* of activity *i*, its bid cost is C_{ij} and duration is d_{ij}. Also, for subcontractor j of activity i, its anticipated quality is q_{ij}. The contractor uses AHP to determine q_{ij}, which evaluates the work proposed by the subcontractor. In this study, the objective is to select the optimal/near optimal combination of subcontractors for all the activities to minimize the project time (***T***), total cost of the project (C) while maximizing the project quality (Q_α) with respect to the total budget of the project, (TB).

Table 3.1 : defined parameters for illustrative example.

Notation	Description
B	Total budget of the project defined as $140000
D	Due-date of the project defined as 121 days
IC	Indirect cost for activities in the project defined as $50 per day
β	Penalty cost for a tardiness of the project defined as $200 per day
I	Incentive cost for a earliness of the project defined as $120 per day
Min_{Dur}	104
Max_{Dur}	169
Min_{Cost}	109965
Max_{Cost}	178920
$Min_{Quality}$	62.67
$Max_{Quality}$	97.04

$$\min T = \max_{L_p \in L} \sum_{i \in L_p} \sum_{j=1}^{m_i} d_{ij}\, x_{ij} \qquad \textbf{(3.1)}$$

$$\min C = \sum_{i=1}^{N} \sum_{j=1}^{m_i} c_{ij} d_{ij} + IC \times T + \beta \left[T - D\right]^+ - I \left[D - T\right]^+ \qquad \textbf{(3.2)}$$

$$\max_{Q_\alpha} = \alpha Q_{min} + (1 - \alpha)\, Q_{avg} \qquad \textbf{(3.3)}$$

Subject to

$$Q_{min} = \min \left\{ q_{ij} : \; x_{ij} = 1 \right\} \qquad 1 \le i \le N, \;\; 1 \le j \le m_i \qquad \textbf{(3.4)}$$

$$Q_{min} \le q_{ij} \times x_{ij} + M\,(1 - x_{ij}) \qquad 1 \le i \le N, \;\; 1 \le j \le m_i \qquad \textbf{(3.5)}$$

$$Q_{avg} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{m_i} q_{ij} \times x_{ij}}{N} \qquad 1 \le i \le N, \;\; 1 \le j \le m_i \qquad \textbf{(3.6)}$$

$$t_k \geq t_i + \sum_{j=1}^{m_i} d_{ij} x_{ij} \qquad 1 \leq i \leq N, \quad \forall\, k \in S_i \tag{3.7}$$

$$T \leq D \tag{3.8}$$

$$\sum_{j=1}^{m_i} x_{ij} = 1, \qquad 1 \leq i \leq N \tag{3.9}$$

$$\sum_{i=1}^{m_i} \cdot \sum_{j=1}^{N} c_{ij} \cdot x_{ij} \leq B \tag{3.10}$$

$$Z_{max} = \mu_{time} + \mu_{cost} + \mu_{quality} \tag{3.11}$$

$$\mu_{time} = \frac{Max_{duration} - Duration}{Max_{duration} - Min_{duration}} \tag{3.12}$$

$$\mu_{cost} = \frac{Max_{cost} - Cost}{Max_{cost} - Min_{cost}} \tag{3.13}$$

$$\mu_{quality} = \frac{Quality - Min_{quality}}{Max_{quality} - Min_{quality}} \tag{3.14}$$

Table 3.2 : Mathematical notations.

Notation	Description
T	Total duration of the project
D	Due-date of the project
L	Sets of all the network path (1, 2,..., P)
P	Total number of the network paths
N	Total number of activities in the project
L_p	Activity sequence of the p_{th} path={ $i_{1,p}, i_{2,p}, \dots, i_{n,p}$}
$i_{n,p}$	Sequence number of activity n on the p_{th} path
d_{ij}	Duration of option j for activity i, for $i = 1,\ldots,N$, for $j = 1, \dots, m_i$
x_{ij}	Index variable of activity i when performing the jth option = if $x_{ij} = 1$ then activity i performs the jth option, while $x_{ij} = 0$ means not
C_{ij}	Cost of subcontracting option j for activity i, for $i = 1$, for $j = 1, \dots, m_i$
m_i	Number of subcontracting (time–cost–quality) option for activity i, for $i = 1, \dots, N$
IC	Indirect cost for activities in the project
β	Penalty cost for a tardiness of the project
I	Incentive cost for a earliness of the project
Q_α	Project quality
A	Relative importance between Q_{min} and Q_{avg}
Q_{min}	Minimum quality of all subcontracting option selected for all the activities in the project
Q_{avg}	Average quality of all subcontracting option selected for project
q_{ij}	Quality of option j of activity i, for $i = 1, \ldots, N$, for $j = 1, \dots, m_i$
M	Arbitrary large number, so that constraint becomes non-binding. For example M could be set at 100 if our quality scale has maximum of 100
t_i	Scheduled starting time of activity i, for $i = 1, \ldots, N$
S_i	Set of activities that are immediate successor of activity i
TB	Total budget of the project

The formulas from (3.1) to (3.9) are taken from (Santosh et al., 2013). The constraint (3.10) added to this study in order to consider a practical budgeting constraint in each construction project. The linear normalization used for both cost and benefit attributes are represented in Eq. (3.11) to (3.14) (Shih, Shyur, & Lee, 2007). In this study, the objective is to select the optimal/near optimal combination of subcontractors for all the activities to minimize the project time (T), total cost of the project (C) while maximizing the project quality (Q_α) with respect to the total budget of the project, (TB).

It is assumed that each activity has subcontracting alternatives submitted by different subcontractors for its completion that represents time, cost and quality. The above single-objective optimization model is developed to quantify and measure the impact of subcontracting alternatives on the project performance. The objective function (3.1) sums the duration of all the activities in the project network. The objective function (3.2) represents the total cost of a project that includes the cost of subcontracting alternatives selected for all the activities, indirect cost, incentive cost for earliness the project and penalty cost for tardiness of the project. The objective function (3.3) estimates the project quality, which depends on the Q_{min} (minimum quality of all activity alternatives selected), Q_{avg} (average quality of all subcontracting alternatives selected for the project) and α (deviation between $Q_{min}$and Q_{avg}). Putting a higher weight on Q_{min} (higher α value) would mean a greater desire to ensure that no individual task has a quality level that is too low, while a lower α value would focus more on the overall average quality of all tasks. For example, if Q_{min} is judged to be $3/4$ times as important as Q_{avg} then the appropriate value would be 0.75. Constraint (3.4) ensures that Q_{min} is the minimum quality of all selected activities. Constraint (3.5) indicates that if x_{ij}=1 (if subcontracting option j for activity i is selected) then $Q_{min} \leq q_{ij}$ as required, while if x_{ij}=0 then $Q_{min} \leq M$. Thus, this constraint ensures that Q_{min} cannot exceed the quality of any selected subcontracting alternative selected. Constraint (3.6) specifies that Q_{avg} is the average quality of the entire selected subcontracting alternatives in construction project. Constraints (3.7) play a role to enforce the successor relationship among the activities in the project network. Constraints (3.8) ensure that the project time is always within the due-date of project. Constraint (3.9) guarantees only one subcontracting alternative for each activity in the project. Constraint (3.10) guarantees only subcontracting alternative, which are below the defined total budget of the project. It should be noted that the mentioned problem considering all the constraints in the (Santosh et al., 2013), with the proposed extra budget constrain represented at Eq. (3.10). In spite of making the problem more difficult to handle, the proposed additional budget constraint, helps for better practical results in optimal subcontracting solutions, which consequences concentration on the results lower than the total budget of the project. Constraints (3.11, 3.12, 3.13 and 3.14) are used to calculate the fitness function.

3.2 Suggested Quality Measurement Method

In this study, the quality measurement and allocated quality scales for 18-network activity and corresponding subcontractors is taken from (Santosh et al., 2013).

3.3 Illustrative Example of Highway Construction Project

In furtherance of validating the mathematical formulation and efficiency of the proposed hybrid GA-SA, an example with 18 activities (Fig. 3.1) is used. The example is adjusted from Feng, (1997) and it is further modified to accommodate the subcontracting quality, as given in Table 3.3. The project consists of 18 activities with average 3.4 subcontracting alternatives that create near about 3.6 billion possible combinations for scheduling the entire project (Santosh et al., 2013) Each of these possible combination leads to a unique impact on project performance such as project time, cost and quality. The present optimization model is developed to search the large search space of possible solutions.

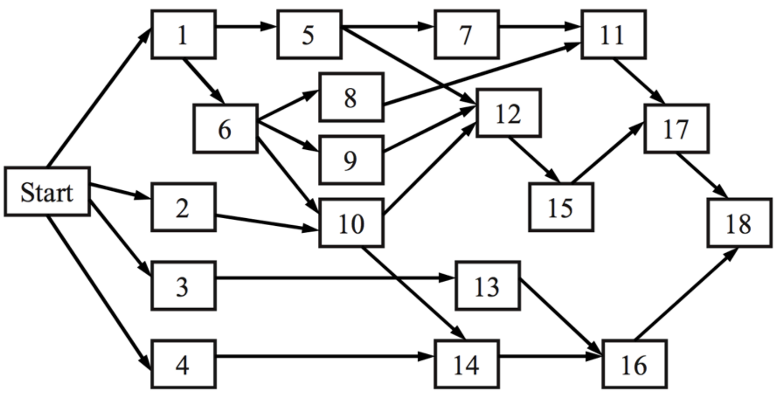

Figure 3.1 : 18-activity project planning network.

The algorithm has been implemented in Matlab R2013b. The experimental study has been conducted on laptop MacBook air 1.3 GHz Intel Core i5 with 4 GB Ram and OS X Mavericks 10.9.2.

Table 3.3 : Time–cost–quality options for 18-activity example.

Activity	Time-cost-quality options (j)														
	Option 1			Option 2			Option 3			Option 4			Option 5		
	d	c	q	d	c	q	d	c	q	d	c	q	d	c	q
1	14	2400	100	15	2150	90	16	1900	86	21	1500	75	24	1200	63
2	15	3000	98	18	2400	87	20	1800	81	23	1500	77	25	1000	60
3	15	4500	100	22	4000	80	33	3200	62	–	–	–	–	–	–
4	12	45000	99	16	35000	74	20	30000	59	–	–	–	–	–	–
5	22	20000	100	24	17500	93	28	15000	77	30	10000	61	–	–	–
6	14	40000	95	18	32000	76	24	18000	59	–	–	–	–	–	–
7	9	30000	97	15	24000	70	18	22000	61	–	–	–	–	–	–
8	14	220	95	15	215	83	16	200	75	21	208	68	24	120	61
9	15	300	100	18	240	97	20	180	81	23	150	71	25	100	63
10	15	450	94	22	400	79	33	320	63	–	–	–	–	–	–
11	12	450	96	16	350	72	20	300	61	–	–	–	–	–	–
12	22	2000	99	24	1750	89	28	1500	70	30	1000	62	–	–	–
13	14	4000	99	18	3200	73	24	1800	60	–	–	–	–	–	–
14	9	3000	100	15	2400	79	18	2200	63	–	–	–	–	–	–
15	16	3500	100	–	–	–	–	–	–	–	–	–	–	–	–
16	20	3000	97	22	2000	89	24	1750	81	28	1500	72	30	1000	67
17	14	4000	98	18	3200	73	24	1800	62	–	–	–	–	–	–
18	9	3000	98	15	2400	75	18	2200	63	–	–	–	–	–	–

3.4 Parameters for Integer Programming Formulation

Parameters defined for programming the genetic algorithm based on the Table 3.1. The tardiness penalty is \$ 200 per day. The incentive cost is \$120 per day. The indirect cost is \$50 per day. The due date for this example is taken as 121 days. For Pareto-optimization purpose, the linear normalization method requires the tentative maximum, minimum and the current value of the objective function. A preliminary run of Genetic algorithm over every single objective function can provide us the maximum and minimum values of the both cost (Time, Cost) and benefit attributes (Quality) of the model. As shown in table 3.1, the minimum duration is 104; the maximum duration is 169; the minimum cost is 109965; the maximum cost is 178920; the minimum quality is 62.67 and the maximum quality is 97.04.

3.5 Initial Population

The initial population of 300 is considered for the genetic algorithm of the hybrid search.

3.6 Encoding of Chromosome

The discrete encoding has been used in this study, which described in second chapter 2.7. Each solution (s) is coded as an integer chromosome. Number of genes in the chromosome is the same as the number of activities in the project network. Each gene in the chromosome represents subcontracting option that corresponds to the time, cost and quality of the activity. For 18-activity project network, Fig. (3.1) and Table (3.3) the length of the chromosome is 18 and each gene in this chromosome represents subcontractor option. For example, if the first activity has five subcontracting options then the first gene in the chromosome represents a subcontracting option that is single random integer from 1 to 5. In this way, remaining genes are generated for the remaining activities to form one integer chromosome. As a result, each chromosome represents one possible combination of subcontractors to finish the project.

3.7 Random Number Generation

Random numbers are essential in genetic algorithm as in simulation of majority of numerical computations. There are two important statistical properties for a sequence of random numbers, which are uniformity and independence. In other words, each random number generated is an independent sample drawn from a discrete numbers between 1 and 5. However, some activities have less than 5 solutions. For instance, Activity number 3 in Table (3.3) has only three options. To overcome this problem in programming and rather than penalizing the generated numbers for each activities, each chromosome has a defined possibility for generating the numbers.

For activity 1, 2, 8, 9 and 16 the range of possibilities for gens, is taking only five numbers (between 1 and 5).

For activity 5 and 12 the range of possibilities for gens, is taking only four numbers (between 1 and 4).

For activity 3, 4, 6, 7, 10, 11, 13, 14, 17 and 18 the range of possibilities for gens, is taking only three numbers (between 1 and 3).

For activity 15, the only possible number to take is only one (Number 1) since there is only 1 option available for the activity number 15 as shown in Fig. (3.3) (based on the availability of only one option for the 15th activity according to the Table (3.3).

Population chromosome	**4**	**2**	**3**	**3**	**1**	**2**	**3**	**1**	**3**	**2**	**2**	**3**	**3**	**3**	**1**	**1**	**2**	**3**

Figure 3.2 : Illustration of generated random numbers for 18 activities.

Consequently, there is only one option for activity number 15. Therefore, 15th column in all of the rows of the population just indicates the option number 1.

3.8 Calculating Time, Cost and Quality

In order to run the GA, the generated random numbers has to be calculated for time, cost and quality in separate matrices with the same size of the population. As shown in Fig 3.2 which illustrates an example of a row in a population matrix, Fig 3.3, 3.4 and 3.5 represents an example, that how the values of duration, quality and cost replaces by the described value in initial population for time, cost quality alternatives taken from the Table (3.3).

Duration for chromosome	**21**	**18**	**33**	**20**	**22**	**18**	**18**	**14**	**20**	**22**	**16**	**28**	**24**	**18**	**16**	**20**	**18**	**18**

Figure 3.3 : Duration for exemplary chromosome.

Quality for chromosome	**75**	**87**	**62**	**59**	**100**	**76**	**61**	**95**	**81**	**79**	**72**	**70**	**60**	**63**	**100**	**97**	**73**	**63**

Figure 3.4 : Quality for exemplary chromosome.

Cost for chromosome	**1500**	**2400**	**3200**	**30000**	**20000**	**32000**	**22000**	**220**	**180**	**400**	**350**	**1500**	**1800**	**2200**	**3500**	**3000**	**3200**	**2200**

Figure 3.5 : Cost for exemplary chromosome.

CPM method has been used to calculate the duration of the network. To reach to this goal, the duration of the network calculated.

In order to calculate the cost of the project using the formula 3.2, the project's duration should be taken into consideration to observe if the incentive cost or the penalty cost has to be applied. Referring to Fig 3.3, the CPM chromosome of the network, indicates that for this row of the population, the penalty cost has to be applied since the CPM for this string is 141 and it is more than 121 days (the due date of the project). The incentive cost has to be considered zero since there is 20 days delay for that string. In contrast, the penalty cost has to be applied. The indirect cost, which is \$50, has to multiply by 141 and added to the raw cost. Finally, adding the penalty cost for tardiness of the project, which it is \$200 multiply by 20.

$$\text{Cost} = 129650 + (50 \times 141) + 200 \times (141 - 121) - 120 \times (0) = 140700$$

To calculate the quality, we should determine the minimum and average quality from the provided exemplary chromosome Fig 3.5. As presented in Fig 3.5, the minimum quality is 59 and the average quality calculated as follows:

$$Q_{avg} = \frac{\sum Q_i}{N} = \frac{1373}{18} = 76.28$$

As described in (Santosh Mungle, 2013) the amount of deviation from Q_{min} is left to the expert's discretion and meanwhile the Alpha takes on a value of 1/4.

$$\text{Quality} = (\text{MinQualityOfRows} \times \frac{1}{4}) + (Q_{avg} \times \frac{3}{4})$$

$$\text{Quality} = (59 \times \frac{1}{4}) + (76.28 \times \frac{3}{4}) = 71.96$$

3.9 Evaluate Fitness Function

Fitness function is responsible for measuring the optimality level of the generated solutions in evolutionary algorithms. Since, every sub-elements of the fitness function has different range of variation, they should be linearly normalized to make trade-off between sub-elements of the fitness function. Then, all the values have to be sorted; finally, the beneath mentioned calculations just illustrate a sample calculation for the fitness function.

$$\mu_{time} = \frac{Max_{duration} - Duration}{Max_{duration} - Min_{duration}} \qquad \mu_{time} = \frac{169 - 114}{169 - 104} = 0.85$$

$$\mu_{cost} = \frac{Max_{cost} - Cost}{Max_{cost} - Min_{cost}} \qquad \mu_{cost} = \frac{178920 - 116480}{178920 - 109965} = 0.90$$

$$\mu_{quality} = \frac{Quality - Min_{quality}}{Max_{quality} - Min_{quality}} \qquad \mu_{quality} = \frac{79.38 - 62.67}{97.04 - 62.67} = 0.49$$

$$Z_{max} = \mu_{time} + \mu_{cost} + \mu_{quality} \qquad Z_{max} = 2.24$$

3.10 Elitism

In order to keep the best result of the generations in Matlab, we sort the population, based on its objective functions. To avoid the loss of the best chromosome, we always keep the largest value of the objective function.

3.11 Tournament Selection

As discussed in section 2.11, two individuals are randomly selected from the generated population. For this problem, a pair of chromosome is selected from population sorted by their fitness values. A chromosome with a higher fitness value dominates the other one and so the weaker chromosome will be replaced by the stronger and apply this procedure for the 50% of our population.

Once Tournament selection is completed the fitness value will be computed and sorted accordingly.

3.12 Crossover

As discussed in previous chapter, crossover considered as an important step in genetic algorithm. For this problem, four gens are selected randomly from a row of a population and exchanged with the four gens of other row. In each generation off-springs are responsible for composing the 50% of the population.

3.13 Mutation

Mutation rate set to be 0.05 for this problem based on the studies and experiments (Santosh et al., 2013). The total number mutation in the population in each generation is calculated by Eq. (2.2). As described in, Table 3.3, since all activities cannot be assigned to all sub-contractors, this limitation should be taken into consideration in mutation operation.

With regard to the 18-activity network Table (3.3), there are five possible options for 1^{st}, 2^{nd}, 8^{th}, 9^{th} and 16^{th} gen for mutation. Likewise, the availability of three possible options for gen 3^{rd}, 4^{th}, 6^{th}, 7^{th}, 10^{th}, 11^{th}, 13^{th}, 14^{th}, 17^{th} and 18^{th}. There are four possibilities for gen 5^{th} and 12^{th} for mutation. Further more, the 15^{th} gen cannot accept mutation since there is only one possible option for that gene.

3.14 Hybrid Search

Simulated annealing (SA) and genetic algorithms (GA) reproduce powerful combinatorial optimization techniques.

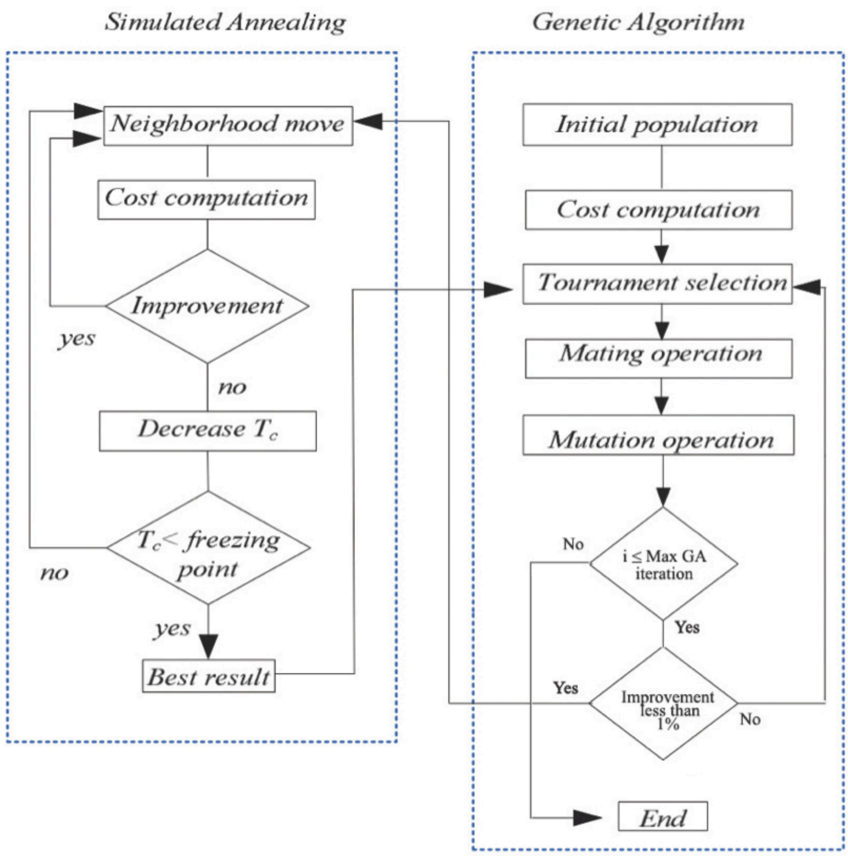

Figure 3.6 : The hybrid GA-SA Algorithm.

The structure of the simulated annealing is very similar to the genetic algorithm. So, in this research simulated annealing coded as a function, which will be used in combination with genetic algorithm Fig. (3.6). The Genetic algorithm uses the simulated annealing function, when the Z_{max} remains constant and the Pareto difference is less than a 0.01.

3.15 Total Budget

According to the Eq. (3.10), total budget (TB) of the project has to be taken into consideration. The total budget limitation is considered for the problem under study is $140,000. Therefore, in order to generate the population, the algorithm repeats to generate a result, less than upper-mentioned amount. If there might be solutions with better qualities and durations with more than the defined budget, they would not be taken into consideration.

3.16 Improving the Solution

As opposed to the Genetic Algorithm, which is a population-based algorithm, Simulated Annealing consists of an atomic structure. The atomic structure is adopted from the GA's elite chromosome. The neighborhood search operator of the Simulated Annealing will do the further improvements.

As described in, Table 3.3, since all activities cannot be assigned to all sub-contractors, this limitation should be taken into consideration through the improvement of the solution.

3.17 Parameters for Simulated Annealing

Parameters defined for programming the Simulate Annealing is based on the Table (4.2). The initial temperature sets for SA to start is 200. The stopping temperature, T_0 , is set to 1 with a cooling rate of 0.01.

3.18 Calculation of Energy State, Time-Cost-Quality in Simulated Annealing

All the calculations for the time, cost, and quality are the same as the genetic algorithm, described in section 3.8. Also, the computation of the energy state function for the Simulate Annealing is exactly the same as the fitness function calculation in GA, which has explained in sect 3.9.

3.19 Neigborhood Search in Simulated Annealing

As discussed in 2.6.6.3 Swap Mutation, the neigborhood search for our problem selects the two atoms in a solution and then exchanges the atoms. The problem is, there are limitations for some atoms to be exchanged. Each time when the two random numbers are selected, the program checks the availability of the options for exchange procedure. After the performing the neigborhood search, since the atoms are changed, we have to calculate the time, cost and quality of the updated solution (atomic structure) in order to calculate the energy state function.

In order to avoid premature convergence, worse move might be accepted by a certain probability function, which depends on the current energy state level.

The uphill probability is as Eq. (4.1) (Jun, 2005).

$$Uphill = exp(\frac{-\Delta}{T}) \tag{4.1}$$

After a worse move is detected, the value calculated by the uphill probability is compared with a randomly generated number. Worse move is accepted if uphill probability is smaller than the random number. Otherwise, a worse move will be rejected.

The SA algorithm will be terminated when the temperature reaches to 1 and the solution will be passed to the GA. The hybrid algorithm terminates the number of iterations reach to 1500.

4. COMPUTATIONAL RESULTS

The time-cost-quality trade-off (TCQ) problem is an important research area in project management and TCQ problems have attracted the attention of many researchers in recent decades. In this study, a hybrid GA-SA algorithm is coded in MATLAB R2013b (8.2.0.701) and run on a 1.3 GHz Intel Core i5 CPU computer. The GA/SA finds the optimal solutions for the TCQ problems almost every time in less than a minute on average. The obtained results represent the maximum values for the corresponding objectives. An appropriate initial parameter setting of GA/SA has a significant impact on the solving progress, such as the exploitation or exploration rates of the search space, and therefore the quality of the solution. As a result, it is required to perform a trial and error experimentation to find a suitable value of the initial parameter. The best parameters for GA are shown in Table (4.1). The best Pareto-optimal solution is found out at 1500 generations for population size 300. However, in real-world environments, the experimental process to define the best values for the parameters may become reasonably lengthy and time consuming. In simulated annealing part of the hybrid search, the starting temperature is set to 200 and the stopping temperature is assumed to be 1 and the cooling rate, fixed to 0.01. The parameters for SA are summarized in Table (4.2).

Table 4.1: Best GA parameter for 18-activity example

GA Parameters	Parameters Value
Population size	300
Number of generations	1500
Crossover probability	0.5
Mutation probability	0.05

Table 4.2 : SA parameter for 18-activity example

SA Parameters	Parameters Value
Starting Temperature	200
Stopping Temperature	1
Descending Temperature Rate	0.001

The repository size is fixed as 30 in order to perform the optimization experiment. In this study, we performed 20 optimization trials to demonstrate the effectiveness of the proposed GA-SA. The average computational time of GA/SA over 20 optimization trials is 4 min. The best solution with its subcontracting plan and project objectives is given in Table 4.3. As illustrated in Table (4.3), four solutions with the optimum value for the three objectives are highlighted. The results indicate that the 26th solution exhibit the minimum project time, the 2nd solution exhibit minimum project cost and 30th solution exhibit the maximum project quality, and the 1st solution is the best compromise one.

If the main priority is placed on maximizing the quality, the last solution can provide us the maximum project quality of 85.42. The project can be completed in 108 days with a cost of $142360.

In the case of minimizing the project cost, the second solution is a proper alternative with cost of $111160 and the quality of 73.92. However, the completion time increases to 118 days.

The minimum duration obtained by GA-SA hybrid search is 104 days. In addition, to these three solutions, the proposed approach provides additional 30 solutions, which represents the trade-off between the time, cost and quality.

The presented hybrid search algorithm presented in Appendices and consists of two parts; the cod for the Genetic Algorithm is presented in first part and the second part contains the Simulated Annealing code.

The best solution with its subcontracting plan subject to the budget constraint is given in Table (4.4). As illustrated in that table, four solutions with the optimum value for the three objectives are highlighted. The results indicate that the 17th solution exhibit the minimum project time, the 4th solution exhibit minimum project cost and 30th solution exhibit the maximum project quality, and the 5th solution is the best compromise one.

If the main priority is placed on maximizing the quality, the last solution can provide us the maximum project quality of 81.58. The project can be completed in 108 days with a cost of $137355. In the case of minimizing the project cost, the second solution is a proper alternative with cost of $110930 and the quality of 74.33. However, the completion time increases to 114 days.

Table 4.3 : The hybrid GA-SA Pareto-optimal solution with optimal subcontracting plan without consideration of the total budget constraint.

Optimal subcontracting alternative plan																						
Solutions																			Project Performance			
																			Time days	Cost ($)	Quality (%)	Pareto
1	**1**	**4**	**1**	**3**	**4**	**3**	**3**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**2**	**1**	**1**	**114**	**116180**	**80.79**	**2.28**
2	**1**	**3**	**3**	**3**	**4**	**3**	**3**	**1**	**1**	**1**	**2**	**1**	**3**	**2**	**1**	**5**	**2**	**1**	**118**	**111160**	**73.92**	**2.09**
3	1	3	3	3	4	3	3	1	1	1	1	4	3	1	1	2	1	1	122	113420	76.21	2.07
4	2	2	1	3	4	3	3	1	1	1	1	1	3	2	1	5	1	2	121	113620	76.42	2.09
5	1	5	3	3	4	3	3	1	1	1	1	1	2	2	1	2	1	1	114	112980	76.54	2.21
6	1	4	3	3	4	1	3	2	1	1	1	1	2	3	1	2	1	2	111	134165	76.62	1.95
7	1	1	3	3	4	3	3	3	1	1	1	1	3	2	1	2	1	1	114	113560	76.75	2.2
8	1	4	2	3	4	3	3	2	1	1	1	1	3	2	1	2	1	1	114	112875	76.96	2.22
9	1	4	1	3	4	3	3	2	1	1	2	1	3	1	1	3	1	1	114	113625	77.33	2.22
10	1	4	2	3	4	1	3	2	1	1	1	1	3	2	1	4	1	1	105	132845	77.75	2.09
11	1	4	1	3	4	3	3	2	1	1	1	1	3	2	1	2	1	1	114	113375	77.79	2.24
12	1	4	2	3	4	3	3	2	1	1	1	1	3	1	1	2	1	1	114	113475	77.83	2.24
13	1	4	3	1	4	2	2	1	3	1	2	1	2	1	1	4	1	1	113	144190	78.88	1.84
14	4	5	2	1	4	1	3	1	1	1	3	1	1	2	1	3	1	1	112	149940	78.96	1.77
15	1	4	2	3	4	2	3	1	1	1	1	1	3	1	1	2	1	1	108	126460	79.04	2.18
16	1	3	1	3	4	1	3	1	1	1	3	2	2	1	1	3	1	1	107	135840	79.17	2.06
17	2	4	1	3	4	1	3	1	1	1	1	1	2	3	1	2	1	1	106	134970	79.25	2.09
18	2	2	1	3	4	2	2	5	1	1	1	1	2	1	1	2	1	1	109	131080	79.38	2.1
19	1	4	2	3	4	2	3	1	1	1	1	2	1	1	1	1	1	1	110	129750	80.58	2.14
20	1	3	1	1	2	3	3	2	1	1	1	1	1	2	1	5	1	1	114	137375	81.67	2
21	1	4	1	2	4	2	2	1	1	1	1	1	1	2	1	3	1	1	108	135310	81.79	2.13
22	1	5	1	1	2	3	3	1	1	1	1	2	2	1	1	1	1	1	116	138470	81.92	1.96
23	1	3	1	3	4	2	3	1	1	1	1	1	1	1	1	1	1	1	108	130460	82	2.2
24	1	2	1	3	1	2	3	2	2	1	1	2	1	1	1	3	1	1	113	140345	82.17	1.99
25	1	3	1	2	3	3	2	1	1	1	1	1	1	1	1	2	1	1	114	128480	82.62	2.16
26	**1**	**1**	**1**	**3**	**3**	**1**	**2**	**1**	**1**	**1**	**1**	**1**	**3**	**1**	**1**	**1**	**1**	**1**	**104**	**143780**	**82.92**	**2.1**
27	1	5	3	3	4	2	3	2	1	1	1	1	3	2	1	3	1	1	108	124305	75.88	2.11
28	1	2	1	1	4	2	1	1	2	1	2	1	2	1	1	2	1	1	111	152610	83.38	1.88
29	1	1	2	2	3	2	2	3	1	1	1	3	1	1	1	1	1	1	114	143660	84.25	1.99
30	**1**	**3**	**1**	**2**	**3**	**2**	**2**	**1**	**1**	**1**	**2**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**108**	**142360**	**85.42**	**2.13**

The minimum duration achieved by GA-SA hybrid search for the problem with total budget constraint is 105 days.

The proposed method provides extra 30 solutions, which represents the trade-off between the time, cost and quality.

Table 4.4 : The hybrid GA-SA Pareto-optimal solution with optimal subcontracting plan with subject to the total budget constraint.

Solutions	Optimal subcontracting alternative plan																		Project Performance			
																			Time days	Cost ($)	Quality (%)	Pareto
1	1	4	3	3	4	3	3	2	1	1	1	1	3	3	1	2	2	2	124	112415	73.54	1.97
2	1	5	3	3	4	3	3	4	1	1	1	1	2	3	1	5	1	1	114	111768	73.83	2.14
3	1	4	2	3	4	3	3	4	3	1	1	1	3	3	1	4	1	1	119	112898	74.17	2.06
4	**1**	**4**	**3**	**3**	**4**	**3**	**3**	**1**	**1**	**1**	**3**	**1**	**3**	**2**	**1**	**5**	**1**	**1**	**114**	**110930**	**74.33**	**2.17**
5	**1**	**3**	**1**	**3**	**4**	**3**	**3**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**5**	**1**	**1**	**114**	**115480**	**80.04**	**2.27**
6	2	2	3	3	4	3	3	3	1	1	1	2	3	2	1	4	1	1	117	112470	74.75	2.12
7	1	3	2	3	4	3	3	4	1	1	1	1	3	3	1	2	2	1	118	112848	74.79	2.1
8	1	3	2	3	4	3	3	4	1	1	2	1	3	2	1	3	1	1	114	112818	75.17	2.17
9	2	3	2	3	4	3	3	4	1	1	1	2	3	2	1	3	1	1	117	112928	75.33	2.13
10	1	5	3	3	4	2	3	2	1	1	1	1	1	3	1	4	2	1	112	125935	75.42	2.02
11	1	5	3	3	4	2	3	2	1	1	1	1	3	2	1	3	1	1	108	124305	75.88	2.11
12	2	3	3	2	4	2	2	4	1	1	1	2	3	2	1	4	1	1	111	131858	76.17	1.97
13	2	3	2	3	4	3	3	1	1	1	1	2	3	2	1	3	1	1	117	112940	76.46	2.16
14	2	3	2	3	4	3	3	1	1	1	1	2	3	2	1	3	1	1	117	112940	76.46	2.16
15	1	3	1	3	4	3	3	1	1	1	1	2	3	3	1	4	1	1	116	113070	76.67	2.18
16	1	4	2	3	4	3	3	2	1	1	1	1	3	2	1	2	1	1	114	112875	76.96	2.22
17	**1**	**4**	**1**	**3**	**4**	**1**	**3**	**2**	**1**	**1**	**3**	**1**	**2**	**2**	**1**	**5**	**1**	**1**	**105**	**134095**	**77.46**	**2.06**
18	2	2	3	2	4	2	2	2	1	1	1	1	3	3	1	2	1	1	109	132675	77.5	2.03
19	1	5	1	3	4	2	3	4	1	1	2	2	2	1	1	1	1	1	110	128838	77.5	2.07
20	2	4	1	3	4	3	2	1	1	1	1	1	3	2	1	2	1	1	115	115300	78.25	2.21
21	1	1	3	1	3	3	2	2	1	1	1	1	3	1	1	2	1	1	114	136175	80.67	1.99
22	1	3	1	3	4	1	3	3	1	1	1	1	1	1	1	2	1	2	111	137350	80.67	2.02
23	1	2	2	3	1	3	2	2	1	1	1	1	2	1	1	2	1	1	114	127775	80.79	2.12
24	1	1	1	3	1	3	1	1	1	1	1	1	3	3	1	4	1	1	114	132180	80.92	2.05
25	1	3	2	3	4	3	1	1	2	1	1	1	1	2	1	1	1	1	117	124830	80.96	2.12
26	1	3	1	3	1	3	2	2	2	2	1	1	1	1	1	4	1	1	121	129055	81	1.99
27	1	2	1	3	3	3	3	1	3	1	1	1	1	1	1	2	1	1	119	122810	81.08	2.12
28	1	2	1	3	4	2	1	2	1	1	1	1	2	1	1	4	1	1	108	136755	81.12	2.09
29	1	3	1	3	3	3	1	1	3	1	1	1	2	1	1	2	1	1	119	129410	81.25	2.03
30	**1**	**1**	**1**	**3**	**4**	**2**	**1**	**2**	**1**	**1**	**1**	**1**	**2**	**1**	**1**	**4**	**1**	**1**	**108**	**137355**	**81.58**	**2.09**

The top thirty best Compromise solutions obtained by the hybrid GA-SA among the 300 solutions are shown in a following table. It should be noted that, it is not possible to improve one of the objectives without reducing at least on of the objectives. As can be seen from the Table (4.5), the maximum value of pareto achieved is 2.278. It represents a relatively equal trade-off between the time, cost, and quality. The project quality has reached to 80.4% while the project has completed within the project due date and it costs around \$115930. The table also depicts that in general, for any increment of the project's quality it accompanies with the increase of the project cost.

Table 4.5: The best compromise solution obtained by the hybrid GA-SA with subject to the total budget constraint.

Solutions	Optimal subcontracting alternative plan																		Project Performance			
																			Time days	Cost ($)	Quality (%)	Pareto
1	**1**	**4**	**1**	**3**	**4**	**3**	**3**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**3**	**1**	**1**	**114**	**115930**	**80.46**	**2.278**
2	1	4	1	3	4	3	3	1	1	1	1	1	1	1	1	1	1	1	114	117180	81.12	2.277
3	1	4	1	3	4	3	3	1	1	1	1	1	1	1	1	2	1	1	114	116180	80.79	2.273
4	1	4	1	3	4	3	2	1	1	1	1	1	1	1	1	2	1	1	114	118180	81.17	2.268
5	1	4	1	3	4	3	3	1	1	1	1	1	1	2	1	2	1	1	114	115580	79.92	2.266
6	1	4	1	3	4	3	3	2	1	1	1	1	1	1	1	2	1	1	114	116175	80.29	2.265
7	1	1	1	3	4	3	3	1	1	1	1	1	1	1	1	4	1	1	114	117180	80.96	2.262
8	1	3	1	3	4	3	3	1	1	1	1	1	3	1	1	4	1	1	114	113780	78.62	2.260
9	1	4	1	3	4	3	3	1	1	1	1	1	2	1	1	3	1	1	114	115130	79.38	2.259
10	1	3	1	3	4	3	3	3	1	1	1	1	1	1	1	2	1	1	114	116460	80.12	2.257
11	1	4	1	3	4	3	2	1	1	1	1	1	1	1	1	1	1	1	114	119180	81.5	2.255
12	1	1	1	3	4	3	3	1	1	1	1	1	2	1	1	1	1	1	114	117880	80.92	2.252
13	1	4	1	3	4	3	2	1	1	1	1	1	2	1	1	2	1	1	114	117380	80.08	2.251
14	1	3	1	3	4	3	3	1	1	1	1	1	2	2	1	2	1	1	114	115080	79	2.247
15	1	1	2	3	4	3	3	1	1	1	1	1	1	2	1	3	1	1	114	116330	79.62	2.247
16	1	1	2	3	4	3	3	1	1	1	1	1	1	2	1	3	1	1	114	116330	79.62	2.247
17	1	1	3	3	4	3	3	1	1	1	1	1	1	1	1	5	1	1	114	115380	79.17	2.247
18	1	1	1	3	4	3	3	2	1	1	1	1	2	1	1	1	1	1	114	117875	80.42	2.247
19	1	4	2	3	4	3	3	2	1	1	1	1	1	1	1	2	1	1	114	115675	79.46	2.245
20	1	1	1	3	4	3	3	2	1	1	1	1	3	1	1	1	1	1	114	116475	79.88	2.242
21	1	4	1	3	4	3	3	4	1	1	1	1	3	1	1	2	1	1	114	113968	78.04	2.241
22	1	3	2	3	4	3	3	2	1	1	1	1	1	2	1	2	1	1	114	115375	78.75	2.239
23	1	4	2	3	4	3	3	2	1	1	1	1	3	1	1	2	1	1	114	113475	77.83	2.238
24	1	4	1	3	4	3	3	2	1	1	1	1	3	2	1	2	1	1	114	113375	77.79	2.238
25	1	1	1	3	4	3	2	1	1	1	1	1	2	1	1	5	1	1	114	117880	80.04	2.237
26	1	1	2	3	4	3	2	2	1	1	1	1	1	1	1	2	1	1	114	119175	80.71	2.236
27	1	4	3	3	4	3	2	1	1	1	1	1	1	1	1	2	1	1	114	116880	79.58	2.236
28	1	3	2	3	4	3	3	3	1	1	1	1	1	1	1	1	1	1	114	116960	79.62	2.236
29	1	4	1	3	4	3	3	3	1	1	1	1	2	1	1	2	1	1	114	115360	78.88	2.235
30	1	1	2	3	4	3	3	1	1	1	1	1	3	1	1	4	1	1	114	114480	78.5	2.235

The table 4.6 represents the results achieved by running the program without implementing the simulated annealing function and considering the total budget of the project. The top thirty best Compromise solutions obtained by the genetic algorithm among the 300 solutions are shown in a following table. As can be seen from the Table (4.6), the maximum value of pareto achieved is 2.245. It represents a relatively equal trade-off between the time, cost, and quality. The project quality has reached to 78% while the project has completed within the project due date and it costs around $113320.

Table 4.6: The best solution obtained by the Genetic Algorithm without considering the total budget constraint.

Optimal subcontracting alternative plan

Solutions																			Project Performance			
																			Time days	**Cost** ($)	**Quality** (%)	**Pareto**
1	2	4	3	3	4	3	3	1	1	1	3	1	3	3	1	2	1	1	115	111650	74.17	2.141
2	3	2	2	3	4	3	3	4	1	1	3	1	3	2	1	3	1	1	116	113208	74.38	2.109
3	**1**	**5**	**3**	**3**	**4**	**3**	**3**	**2**	**1**	**1**	**1**	**2**	**3**	**2**	**1**	**3**	**1**	**1**	**116**	**111415**	**74.75**	**2.146**
4	1	4	3	3	4	3	3	1	1	1	2	1	3	3	1	2	1	1	114	111780	75.04	2.180
5	1	4	1	3	4	3	3	5	1	1	3	1	3	2	1	2	1	1	114	113130	75.42	2.171
6	1	3	3	3	4	2	3	3	1	1	1	1	3	2	1	5	1	1	108	124340	75.83	2.113
7	1	3	1	3	4	3	3	5	1	1	1	1	3	2	1	5	1	1	114	112580	76.13	2.200
8	1	4	1	3	4	3	3	1	1	1	3	1	3	3	1	2	1	1	114	113030	76.17	2.194
9	1	3	2	3	4	3	3	5	1	1	1	1	3	2	1	2	1	1	114	113080	76.21	2.195
10	1	4	1	3	4	3	3	1	1	1	2	1	2	3	1	4	1	1	114	113980	76.46	2.189
11	3	4	2	3	4	2	2	1	1	1	1	2	3	2	1	3	1	1	112	127540	77.21	2.045
12	2	4	1	3	4	3	3	1	1	1	1	1	3	2	1	2	1	1	115	113300	77.88	2.225
13	**1**	**4**	**2**	**3**	**4**	**3**	**3**	**1**	**1**	**1**	**1**	**1**	**3**	**1**	**1**	**3**	**1**	**1**	**114**	**113230**	**78.00**	**2.245**
14	1	3	1	3	4	2	3	1	1	1	1	2	3	1	1	5	1	1	110	126350	78.71	2.137
15	1	3	1	3	2	2	3	1	2	1	1	2	3	2	1	5	1	1	113	133700	79.04	1.994
16	1	3	2	3	4	1	2	1	1	1	1	1	1	3	1	2	2	1	109	137530	79.42	2.011
17	1	2	1	3	4	1	3	3	1	1	1	2	3	1	1	3	1	1	107	135170	79.50	2.078
18	1	3	2	3	4	1	3	1	1	1	1	1	3	1	1	2	1	1	105	134250	80.00	2.137
19	1	1	1	3	4	2	2	1	2	1	1	1	3	1	1	4	1	1	111	130410	80.29	2.109
20	**1**	**1**	**1**	**3**	**3**	**1**	**3**	**1**	**1**	**1**	**2**	**1**	**3**	**1**	**1**	**2**	**1**	**1**	**104**	**140680**	**81.21**	**2.094**
21	1	1	1	3	4	2	3	1	1	1	1	1	1	2	1	2	1	1	108	130060	81.50	2.195
22	3	3	1	3	1	3	3	1	1	1	1	1	1	1	1	3	1	1	116	126070	81.67	2.135
23	1	2	1	3	2	2	3	1	1	1	1	2	2	1	1	2	1	1	110	136850	81.75	2.073
24	1	4	1	2	3	2	3	2	1	1	1	1	1	1	1	5	1	1	108	138155	81.88	2.088
25	1	2	1	3	1	1	2	1	1	1	1	1	2	1	1	2	3	1	114	148080	82.13	1.859
26	1	1	2	3	4	1	3	1	2	1	1	1	1	1	1	1	1	1	107	138930	82.54	2.112
27	1	4	1	3	1	1	3	1	2	1	2	2	1	1	1	1	1	1	109	147920	82.71	1.956
28	1	4	1	2	1	3	3	1	1	1	1	1	1	1	1	3	1	1	114	130930	82.71	2.125
29	1	1	1	1	4	1	3	1	1	1	1	1	2	2	1	3	1	1	105	151500	83.04	1.975
30	**1**	**4**	**1**	**1**	**2**	**3**	**2**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**3**	**1**	**1**	**114**	**140430**	**83.83**	**2.020**

The table (4.7) represents the results achieved by running the genetic algorithm without implementing the simulated annealing function. The budget constraint has been considered for the total budget of the project. The top thirty best Compromise solutions obtained by the genetic algorithm among the 300 solutions are shown in the Table (4.7).

Table 4.7: The best solution obtained by the Genetic Algorithm with subject to the total budget constraint

Optimal subcontracting alternative plan

Solutions																			Project Performance			
																			Time days	Cost ($)	Quality (%)	Pareto
1	1	2	3	3	4	3	3	1	1	1	2	4	3	2	1	3	3	1	132	113370	72.75	1.813
2	2	3	2	3	4	3	3	4	4	1	1	1	3	1	1	5	2	1	127	114258	73.79	1.907
3	1	4	2	3	4	2	3	4	1	1	3	1	3	2	1	4	1	1	108	125198	74.88	2.073
4	1	4	1	3	4	2	3	4	1	1	3	2	3	2	1	5	1	1	110	125288	75.08	2.047
5	1	4	3	3	4	3	3	3	1	1	1	2	2	1	1	4	2	1	120	113530	75.13	2.065
6	1	5	1	3	4	3	3	2	1	1	1	1	2	3	1	2	2	1	118	113955	75.92	2.112
7	**1**	**3**	**3**	**3**	**4**	**3**	**3**	**1**	**1**	**1**	**1**	**1**	**3**	**1**	**1**	**4**	**1**	**2**	**120**	**112900**	**76.08**	**2.102**
8	1	4	2	3	4	3	3	2	1	2	1	1	3	1	1	5	1	1	121	113615	76.29	2.082
9	1	2	1	3	4	3	3	4	1	1	2	1	3	1	1	5	1	1	114	113768	76.54	2.195
10	1	2	1	3	4	3	3	4	1	1	2	1	3	1	1	5	1	1	114	113768	76.54	2.195
11	1	4	3	3	4	1	3	1	1	1	2	3	3	1	1	2	1	1	111	133570	76.88	1.963
12	1	3	1	3	4	3	3	4	1	1	2	1	3	1	1	3	1	1	114	113918	76.88	2.202
13	4	3	1	2	4	2	2	2	1	1	2	1	3	2	1	4	1	1	115	133345	77.17	1.913
14	1	2	1	3	4	3	3	2	1	1	2	1	3	1	1	4	1	1	114	114275	77.38	2.211
15	2	2	1	3	4	2	3	4	1	1	1	1	2	2	1	5	1	1	109	127568	77.50	2.099
16	1	1	3	3	4	3	3	1	1	1	1	2	3	1	1	3	1	1	116	114020	77.71	2.194
17	1	2	1	3	4	2	3	1	1	1	1	1	3	2	1	3	2	1	112	126890	78.04	2.079
18	**1**	**2**	**2**	**3**	**4**	**1**	**3**	**3**	**1**	**1**	**1**	**1**	**3**	**1**	**1**	**3**	**1**	**1**	**105**	**134580**	**79.08**	**2.105**
19	1	2	1	3	4	2	3	1	3	1	1	1	1	2	1	4	1	1	113	129690	79.54	2.066
20	1	4	1	3	4	1	3	1	2	1	1	1	1	2	1	4	1	1	107	135830	80.58	2.100
21	1	5	1	3	3	2	2	1	1	1	1	1	2	1	1	1	1	1	108	135860	81.08	2.099
22	1	3	1	3	3	2	2	5	2	1	1	1	1	1	1	2	1	1	111	136810	81.17	2.041
23	1	2	1	3	4	3	3	1	1	1	1	1	1	1	1	2	1	1	114	117080	81.21	2.282
24	1	3	1	3	4	3	2	1	1	1	1	1	1	1	1	2	1	1	114	118480	81.33	2.266
25	**1**	**4**	**1**	**3**	**3**	**3**	**3**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**2**	**1**	**1**	**114**	**121180**	**81.46**	**2.230**
26	1	2	1	2	2	3	3	1	1	1	1	1	3	1	1	2	1	1	114	127380	81.54	2.143
27	1	3	1	3	1	3	2	2	1	1	1	1	1	2	1	2	1	1	114	127875	81.58	2.137
28	1	3	1	2	4	2	2	2	1	1	2	1	1	1	1	2	1	1	108	136355	81.67	2.108
29	1	2	1	2	4	3	1	2	1	1	1	2	1	1	1	4	1	1	116	129665	81.71	2.084
30	**1**	**3**	**1**	**3**	**4**	**1**	**3**	**2**	**2**	**1**	**1**	**1**	**1**	**1**	**1**	**2**	**1**	**1**	**107**	**137225**	**81.83**	**2.116**

As can be seen from the Table (4.7), four solutions with the optimum value for the three objectives are highlighted. the maximum value of pareto achieved is 2.230. The results indicate that the 18th solution exhibit the minimum project time, the 7th solution exhibit minimum project cost and 30th solution exhibit the maximum project quality, and the 25th solution is the best compromise one.

As shown in Fig. (4.1), there is a slight downward trend between which implies the negative correlation between time and cost of the project. However, a very long project's duration may result in huge cost.

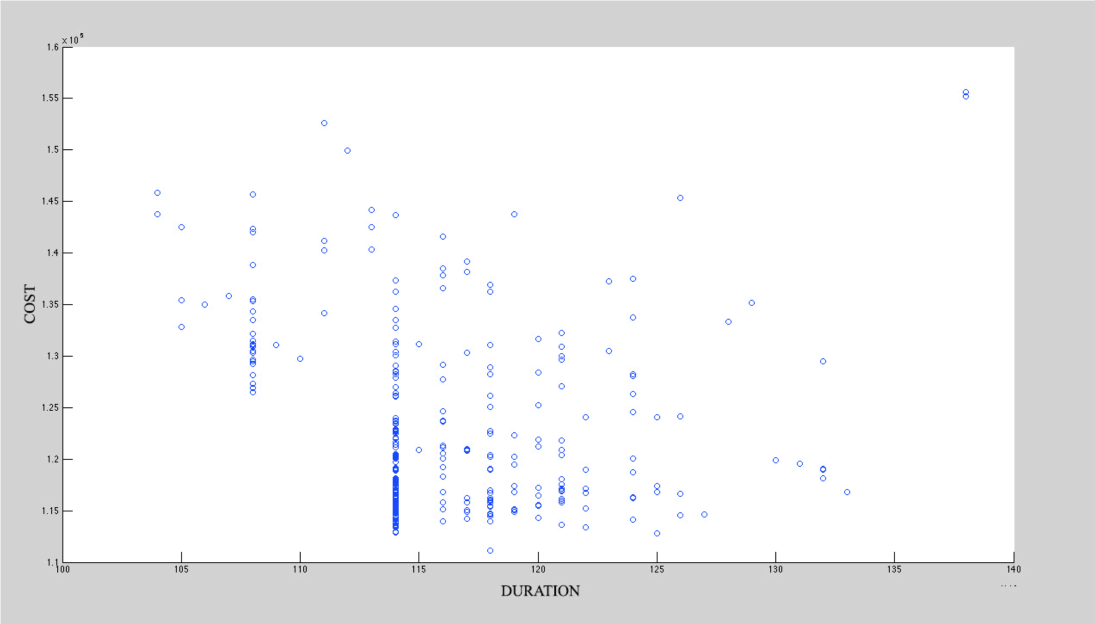

Figure 4.1 : Time-Cost trade-off analysis for 18 activity network.

A similar comparison is made between the projects duration and quality, which is depicted in Fig. (4.2), with quality on the Y-axis and duration on the X-axis. The below graph has a slightly negative declination which indicates that generally as a project get longer to be completed, there is a chance that in its quality declines.

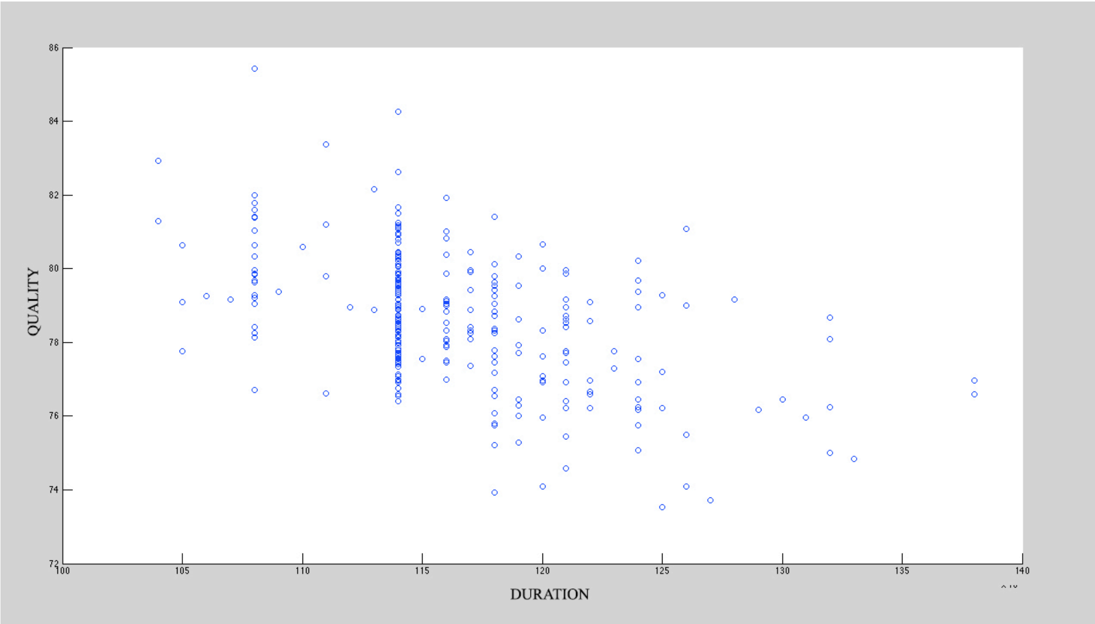

Figure 4.2 : Time-quality trade-off analysis for 18 activity network.

Scatter plot illustrating the general relation between the cost of project and its quality. As can bee seen in Fig. (4.3), high quality projects require higher investments in general.

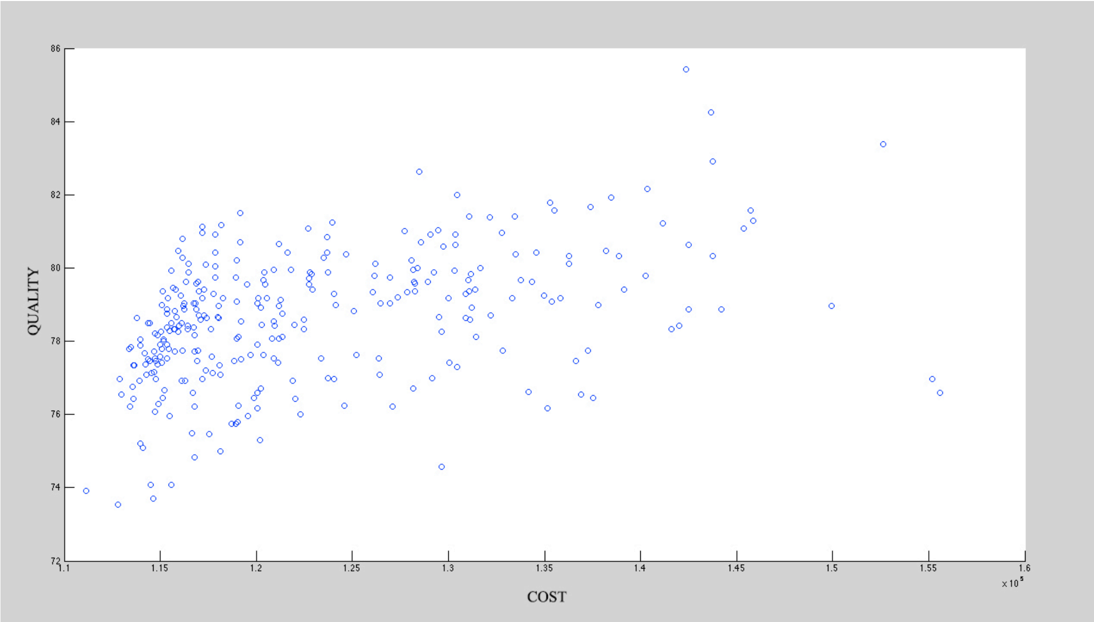

Figure 4.3 : Cost-quality trade-off analysis for 18 activity network.

5. CONCLUSIONS AND FUTURE WORK

This study aimed to solve Time-Cost-Quality Trade-off Problem in construction projects by using the integration of GA and SA optimization techniques via MATLAB R2013b software program. The Time Cost Quality Trade-off (TCQT) problem is an essential research area in project management. Because of its obvious practical importance, the TCQT problems have attracted the attention of many researchers. The review of literature indicated that the TCQT problems can be categorized under three main groups, which include: the deadline problem, the budget, and the quality problem. The objective of the proposed model is to find solutions under the due date of the project beside maximum qualities for each activity with respect to the several model constraints and limitations. There may be many alternative optimal solutions for the proposed model. In practice, the aim should be the minimization of total completion time, maximization of quality while allocating the least possible money, at most the available budget to the project.

Since more efficient optimization methods are needed, this study proposes a hybrid GA-SA algorithm to solve an illustrative example, which consists of 18-activity network problem. Simulated annealing (SA) and genetic algorithms (GA) represent effective and strong combinatorial optimization algorithms among the modern heuristic approaches with complementary strengths and weaknesses. Implementing the respective advantages of the two paradigms, and to improve the local search ability of genetic algorithm, an effective combination of GA and SA, called hybrid GA-SA, is developed to solve the time-cost-quality trade-off problem (TCQT).

Since there are high level of complexities involved in construction projects and difficulty of a subcontractor selection for construction managers, an example of a highway construction network used in this study. Subsequently, the selection of subcontractors and their ability is complex, comprising high levels of ambiguity and uncertainty. This study aims to help for subcontractor selection, which continues to be an area of importance and interest to project managers, government agencies, and organizations responsible for delivering project outcomes with the total budget limitation.

The current study attempts to assign sub-contractors to an 18-network activity of highway construction project. The main objective of this research emphasizes on minimizing the cost and duration and maximizing the project’s quality. Since Pareto optimality is a set of best optimal trade-offs, choosing the best compromise can be always difficult. The integer-programming model represents the sub-contractor assignment aiming to optimize the time–cost–quality trade-off problem in highway construction project. Despite its obvious practical importance, only little effort has been spent to emphasize the budget aspect. In this study, in order to prevent non-dominated solutions from being lost, the best solution is always kept in each iteration, and the successive solutions of these problems are presented in this study.

All the activities' duration, cost and quality are considered to be deterministic while, in a real world application, most of the project's sub-elements are quit probabilistic and the current model fails to consider that. More efficient algorithms and methods are required to be developed to generate more efficient pareto-optimal solutions and prevent hybrid-GA-SA getting stuck in local optimas. The developed Matlab code for the problem under study could be extended in order to solve different networks and it is capable of taking new inputs and values for a new time-cost-quality problems.

The Matlab code for the applied problem could be extended in order to solve different networks and being able to work as an input data program, which can accept new time-cost-quality values for activities.

REFERENCES

Aarts, E. H., & Laarhoven, P. J. (1987). *Simulated Annealing: Theory and Applications.* Eindhoven, Netherlands: Springer Science+Business Media Dordrecht.

Abolfazl Gholipour Mehdi Abadi, M. R. (2013). Extend time - cost trade off model with Weighted Fuzzy Goal Additive Programming. *Technical Journal of Engineering and Applied Sciences* , 336-343.

Alizamir, S., Rebennack, S., & Pardalos, P. M. (2008). *Improving the Neighborhood Selection Strategy in Simulated Annealing using the Optimal Stopping Problem.* INTECH Open Access Publisher.

Aminbakhs, S. (2013). *Hybrid Particle Swarm Optimization Algorithm for Obtaining Pareto Front Of Discrete Time-Cost Trade-Off Problem.*

Anderson, A. R. (2001). *Guidelines for Warranty, Multi-parameter, and Best Value Contracting.* National Cooperative Highway Research Program,. Washington, DC.

Armstrong, W., & MICE, A. T. (1969). *Critical Path Method: Introduction and Practice.* London: Longman Group LTD.

Arsham, D. H. (2014, 07 10). *Deterministic Modeling: Linear Optimization with Applications*. Retrieved 08 20, 2014 from The University of Baltimore: < *http://home.ubalt.edu/ntsbarsh/opre640a/partviii.htm*>

Bäck, T. F., & Michalewicz. (1997). *Handbook of evolutionary computation.* New York, USA: Taylor and Francis group.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys , 35* (3), 268–308.

Caramia, M., & Dell'Olmo, P. (2008). *Multi-objective Management in Freight Logistics.* Springer.

Cernera, V. A. (2013). Profiling the Distance Characteristics of Mutation Operators for Permutation-Based Genetic Algorithms. *Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference.* Computer Science and Information Systems.

Champion, D. J. (2009, Dec). Elitism Between Populations for the Improvement of the Fitness of a Genetic Algorithm Solution. *Novel Algorithms and Techniques in Telecommunications and Networking* , 229-233.

Chen, Z., Jiang, X., Li, J., & Shushen, L. (2013). A sphere-cut-splice crossover for the evolution of cluster structures. *The Journal of Chemical Physics , 138*.

Coello, C. A., Lamont, G. B., & Veldhuizen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems* (Second Edition ed.). (J. R. David E. Goldberg, Ed.) New York, NY, USA: Springer.

Colburn, K., & James, C. (2013, March). Integrated, Multi-pollutant Planning for Energy and Air Quality (IMPEAQ). *RAP* , 14.

Cooper, W. W. (2007). *Data Envelopment Analysis* (Vol. 2nd). USA.

Daly, M., & Wilson, M. (1988). Evolutionary social psychology and family homicide. *242* (4878), 519-524.

Diaz-Gomez, P. A., & Hougen, D. F. (2007). Initial Population for Genetic Algorithms: A Metric Approach. *Gem*, (pp. 43-49). Oklahoma.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms PhD thesis.* Milan, Italy: Politecnico di Milano.

Durillo, J. J., Nebro, A. J., Luna, F., & Alba, E. (2009). *On the Effect of the Steady-State Selection Scheme in Multi-Objective Genetic Algorithms* (Vol. 5467).

Ed Klotz, A. M. (2013, October). Practical Guidelines for Solving Difficult Linear Programs. *Operations Research and Management Science* , 1–17.

Eiben, A. E., & Smith, J. E. (2003). Introduction to Evolutionary Computing. Springer.

Elmaghraby, S. E. (1993). Resource allocation via dynamic programming in activity networks . *European Journal of Operational Research , 64*, 199-215.

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

Fogarty, T. C. (1996). *Evolutionary computing.* Brightonn, UK: Springer.

G K Joshi, S. M. (2008). Optimum Scheduling of Generators Using Genetic Algorithm. *IJCSNS International Journal of Computer Science and Network Security , 8* (6).

Gary B Gehrig, D. Brinegar, F. Gehrig, M. Mix, S. Thompson, and R. Weiland. (1988). *Concepts and methods of schedule compression.* Austin, TX: University of Texas at Austin. Bureau of Engineering Research.

Geem, Z. W. (2005). Harmony search in water pump switching problem. *Lecture Notes in Computer Science (LNCS) , 3612* , 751–760 .

Geem, Z. W. (2010). Multiobjective Optimization of Time-Cost Trade-Off Using Harmony Search. *Construction Engineering and Management , 136*, 711-716.

Genaro, G., & Anand, P. (2000). Analysis of the effects of uncertainty, risk-pooling, and subcontracting mechanism on project performance. *Operations Research , 48* (6), 927–938.

Goldberg, D. (1990). A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. . *Complex Systems. , 4* , 445-460.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning: Addison-Wesley* . Boston,, MA, USA: Longman Publishing Co. Inc.

Grefenstette, J. (1986). Optimization of control parameters for genetic algorithms. *16* (1), 122-128.

Halpin, D. W. (2010). *Construction Management.* John Wiley & Sons. Harshbarger, R. J. (2014). *Just The Facts 101 - e-Study Guide for: College Algebra in Contex.* Content Technologies Inc.

Haupt, R., & Haupt, S. (1998). *Practical genetic algorithms.* Hoboken, New Jersy, USA: A John Wiley & Sons,Inc.

Horst, R., & Pardalos, P. M. (2002). *Handbook of Global optimization* (Panos M. Pardalos and H. Edvwin Romeijn ed., Vol. 2). Kluwer Academic Publishers.

James Kennedy, R. C. (2001). *Swarm Intelligence.* Morgan Kaufmann.

Jensen, P. A., & Bard, J. F. (2002). *Nonlinear Programming Methods.* John Wiley & Sons.

Jensen, P. A., & Bard, J. F. (2003). *Operations Research Models and Methods.* John Wiley & Sons, Inc.

Jun, Y. (2005). Detailed Analysis of Uphill Moves in Temperature Parallel Simulated Annealing and Enhancement of Exchange Probabilities. *Complex Systems , 15*, 349–358.

Kalai, A. T. (2006). Simulated Annealing for Convex Optimization. *Mathematics of Operations Research , 31* (2), 253-266.

Kalouptisdis, N. (1997). *Signal Processing Systems, Theory and Design.* New York , USA: Wiley.

Kangas, A., Kangas, J., & Kurttila, M. (2008). *Heuristic Optimization* (Vol. 16). Springer Netherlands.

Kaoru Tone, B. K. (2003, 5 11). Scale, indivisibilities and production function in data envelopment analysis. *International Journal of Production Economics* , 165-192.

Kelley, J. E. (1961). Critical path planning and scheduling: Mathematical basis. *Operations Research , 9* (3), 296–320.

Kelley, J. E. (1961). Critical Path Planning and Scheduling: Mathematical Basis. , , . *9* (3), 296-320.

Kerzner, H. (2013). *Project Management: A Systems Approach to Planning, Scheduling, and Controlling* . John Wiley & Sons.

Klotz, E., & Newman, A. M. (2013). *Practical Guidelines for Solving Difficult Linear Programs.* Illinois: Division of Economics and Business, Colorado School of Mines.

Kumar, M., Husian, M., Upreti, N., & Gupta, D. (2010). Genetic Algorithm: Review And Application. *International Journal of Information Technology and Knowledge Management , 2* (2), 451-454.

Kusum Deep, H. M. (2011). Combined Mutation Operators of Genetic Algorithm for the Travelling Salesman problem. *International Journal of Combinatorial Optimization Problems and Informatics , 2* (3), 1-23.

Lee, D.-K., Uwe, S., & Shin, S.-M. (2009). Evaluating Optimum Shape of Thin Steel Plates with Dynamic Problem by Using Topology Optimization Design. *5th International Symposium on STEEL STRUCTURES.* Korea.

Leonora, B., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing: an international journal , 8* (2), 239–287.

Leung, Y., Chen, Z.-P., Xu, Z.-B., & Leung, K.-S. (1998). Convergence rate for non-binary genetic algorithms with different crossover operators. The Chinese University of Hong Kong.

Lingo Software (2010). In *LINGO user's guid* (p. 1). Chicago: LINDO Systems Inc. 1415 North Dayton Street CHICAGO IL 60642 U.S.A.

Liu, L. B.-W. (1995). Construction time-cost trade-off analysis using LP/IP hybrid method. *Journal of Construction Engineering and Management , 121* (4), 446–454.

Magalhães-Mendes. (2013). A Comparative Study of Crossover Operators for Genetic Algorithms to Solve the Job Shop Scheduling Problem. *12* (4).

Mahalakshmi, M., Kalaivani, P., & Nesamalar, E. (2013). A Review on Genetic Algorithm and its Applications. *International Journal of Computing , 02*, 415-423.

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs.* Charlotte, USA: Springer.

Misola, M. G., & Navarro, B. B. (2002). A Multi-point Swapped Crossover and Swap Mutation of Genetic Algorithm to Obtain Optimal Facility Layout Design. *Asia Pacific Industrial Engineering and Management System* .

Morse, J. (1980). Reducing the size of non-dominated set: pruning by clustering. *Computers & Operations Research , 7* ((1–2)), 55–66.

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

Mungle, S., Benyoucef, L., Son, Y.-J., & Tiwari, M. (2013). A fuzzy clustering-based genetic algorithm approach for time–cost–quality trade-off problems: A case study of highway construction project. *Engineering Applications of Artificial Intelligence , 26* (8), 1953–1966.

Murty, K. G. (2003). Optimization Models For Decision Making. Dept. Industrial & Operations Engineering University of Michigan.

Myung, H., & Kim, J.-H. (1996). Hybrid evolutionary programming for heavily constrained problems. *BioSystems , 38* (1), 29-43.

Shahsavari Pour N., Modarres M. and Tavakkoli-Moghaddam R. (2012). Time-Cost-Quality Trade-off in Project Scheduling with Linguistic Variables. *World Applied Sciences Journal* , 404-413.

Negnevitsky, M. (2005). *Artificial intelligence: a guide to intelligent systems: Addison-Wesley.* Pearson Education.

Norozi, A., Ariffin, M. K., Ismail, N., & Mustapha, F. (2011, April). An optimization technique using hybrid GA-SA algorithm for multi-objective scheduling problem. *Scientific Research and Essays* , 1720-1731.

Olarenwaju, O., Daniel, A., & Segun, F. (2014). Implementing A Decision-Making Model for a small scale production company. *International Journal of Basic & Applied Sciences , 13* (2), 1-6.

Osman, I. H., & Kelly, J. P. (1996). *Meta-heuristics: an overview.* USA: Springer US.

Pells, D. (1993). *The AMA Handbook of Project Management.* New York, USA: AMA-COM.

Prebys, E. K. (2007). The Genetic Algorithm in Computer Science .

Princeton, S. A. (1998). The approximability of NP-hard problems. 337-348.

R.Sivaraj. (2011). A Review Of Selection Methods In Genetic Algorithm . *International Journal of Engineering Science & Technology* .

S. Boyd, S.-J. K. (2007, 4 10). A tutorial on geometric programming. *Optimization and Engineering* , 67-127,.

Saaty TL, V. L. (1993). A model of neural impulse firing and synthesis. *J Math Psychol , 37*, 200-219.

Saaty, T. (1989). *Conflict Resolution: The Analytic Hierarchy Process.* New York: Praeger.

Saaty, T. (2008). *Group Decision Making: Drawing out and Reconciling Differences.* Pittsburgh, Pennsylvania, USA: RWS Publications.

Samuel L. Baker. (2004). *"Critical Path Method (CPM)".* University of South Carolina, Health Services Policy and Management Courses.

Santiago, J., & Magallon, D. (2009). *CRITICAL PATH METHOD.* Stanford university. Stanford University.

Santosh Mungle, Lyes Benyoucef, Young-Jun Son, M.K. Tiwari (2013 , September). A fuzzy clustering-based genetic algorithm approach for time–cost–quality trade-off problems: A case study of highway construction project . *Engineering Applications of Artificial Intelligence* , 1953–1966.

Saul I. Gass, M. C. (2013). *Analytic Hierarchy Process.* New York : Springer Science+Business Media.

Shih, H.-S., Shyur, H.-J., & Lee, E. S. (2007). An extension of TOPSIS for group decision making. *Mathematical and Computer Modelling , 45*, 801–813.

Siemens, N. (1971). A Simple CPM Time-Cost Tradeoff Algorithm. *Management Science , 17* (6), B354-B363.

Smith, A. E. (2003). *Introduction to Evolutionary Computing.* Springer.

Solnon, C., & Bridge, D. (2005). *An Ant Colony Optimization Meta-Heuristic for Subset Selection Problems.* Rapport de recherche.

Sorokin, A., Rebennack, S., Pardalos, P. M., Iliadis, N. A., & Pereira, M. V. (2012). *Handbook of Networks in Power Systems II.* Springer Science & Business Media.

Spears, W. M., Jong, K. A., Bäck, T., Fogel, D. B., & Garis, H. d. (1993). An overview of evolutionary computation. *667*, 442-459.

Syswerda, G. (1989). Uniform Crossover in Genetic Algorithms. *In Proceedings of the Third International Conference on Genetic Algorithms*, (pp. 2-9).

Upreti, S. (2009). *Mechanical System Design.* PHI Learning Pvt. Ltd.

Url-1 <*http://www.economyprofessor.com/pareto-optimality-1906*>, date retrieved 08.05.2014.

Url-2 <*http://www.obitko.com/tutorials/genetic-algorithms/index.php*>, date retrieved 21.01.2014.

Url-3 < *http://www.friartuck.net/resources/optimization/what-is-optimization.htm* >, date retrieved 04.01.2014.

Wang, L., & Zheng, D.-Z. (2001). An effective hybrid optimization strategy for job-shop scheduling problems. *28* (6), 585-596.

Winston, W. (2003). *Introduction to mathematical programming: applications and algorithms.* USA: Duxbury Resource Center.

Woeginger, G. J. (2003). Exact Algorithms for NP-Hard Problems: A Survey. (M. J. al, Ed.) *LNCS 2570* , 185–207.

WW Cooper, L. S. (2011). Data Envelopment Analysis History, Models and Interpretations. In L. M. William W. Cooper. USA: Springer.

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

Yang, I. T. (2007). Using elitist particle swarm optimization to facilitate bicriterion time-cost trade-off analysis . *Journal of Construction Engineering and Management , 133* (7), 498–505.

Yang, X. S. (2013). Optimization and Metaheuristic Algorithms in Engineering, in: Metaheursitics in Water, Geotechnical and Transport Engineering. (A. H. X. S. Yang, Ed.) 1-23.

Yin , P. Y. (1999). A fast scheme for optimal thresholding using genetic algorithms. *Signal Processing , 72* (2), 85–95.

Zheng, D. X. (2004). Applying a genetic algorithm based multiobjective approach for time-cost optimization. *Journal of Construction Engineering and Management , 130* (2), 168–176.

APPENDICES

APPENDIX A: The first part

APPENDIX B: The second part

APPENDIX A

```
clc;
clear all;
format bank;
display('Program to find the minimum duration for 18 Activity
network using a hybrid GA-SA');
maxit=1500;
npar=18;
popsize=300
mutrate=.055;
selection=0.5;
Activities=18;
keep=floor(selection*popsize);
IC=50;
DueDate=121;
Beta=200;
I=120;
itNumberCrossover=150;
itNumberTournament =150;
genNumber=4;
MinDur= 104;
MaxDur= 169;
MinCost= 109965;
MaxCost= 178920;
MaxQuality= 97.04;
MinQuality= 62.67;
work140=2;
Paretofirst=1;
Paretosecond=0;
Paretodifference=1;
global globalvar
global globalpareto
globalpareto=0;
globalvar=zeros(1,18);
countf=0;
iga=1;
pop=round(rand(popsize,Activities));
[numRows, numCols] = size(pop);


ActivitiesPossibility_1=1
ActivitiesPossibility_2=3
ActivitiesPossibility_3=4;
ActivitiesPossibility_4=5;

for iz=1:popsize
pop(iz,:)=randperm(npar);
end

for i=1:popsize
for j=1:npar

pop(i,1)=round(1+(ActivitiesPossibility_4-1)*rand);
pop(i,2)=round(1+(ActivitiesPossibility_4-1)*rand);
pop(i,3)=round(1+(ActivitiesPossibility_2-1)*rand);
pop(i,4)=round(1+(ActivitiesPossibility_2-1)*rand);
pop(i,5)=round(1+(ActivitiesPossibility_3-1)*rand);
pop(i,6)=round(1+(ActivitiesPossibility_2-1)*rand);
pop(i,7)=round(1+(ActivitiesPossibility_2-1)*rand);
pop(i,8)=round(1+(ActivitiesPossibility_4-1)*rand);
pop(i,9)=round(1+(ActivitiesPossibility_4-1)*rand);
```

```
pop(i,10)=round(1+(ActivitiesPossibility_2-1)*rand);
pop(i,11)=round(1+(ActivitiesPossibility_2-1)*rand);
pop(i,12)=round(1+(ActivitiesPossibility_3-1)*rand);
pop(i,13)=round(1+(ActivitiesPossibility_2-1)*rand);
pop(i,14)=round(1+(ActivitiesPossibility_2-1)*rand);
pop(i,15)=round(1+(ActivitiesPossibility_1-1)*rand);
pop(i,16)=round(1+(ActivitiesPossibility_4-1)*rand);
pop(i,17)=round(1+(ActivitiesPossibility_2-1)*rand);
pop(i,18)=round(1+(ActivitiesPossibility_2-1)*rand);

end
end

Dur_options=     [14 15 16 21 24;
                  15 18 20 23 25;
                  15 22 33 0  0 ;
                  12 16 20 0  0 ;
                  22 24 28 30 0 ;
                  14 18 24 0  0 ;
                  9  15 18 0  0 ;
                  14 15 16 21 24;
                  15 18 20 23 25;
                  15 22 33 0  0 ;
                  12 16 20 0  0 ;
                  22 24 28 30 0 ;
                  14 18 24 0  0 ;
                  9  15 18 0  0 ;
                  16 0  0  0  0 ;
                  20 22 24 28 30;
                  14 18 24 0  0 ;
                  9  15 18 0  0 ];


Cost_options=      [2400  2150  1900  1500  1200 ;
                    3000  2400  1800  1500  1000 ;
                    4500  4000  3200  0     0    ;
                    45000 35000 30000 0     0    ;
                    20000 17500 15000 10000 0    ;
                    40000 32000 18000 0     0    ;
                    30000 24000 22000 0     0    ;
                    220   215   200   208   120  ;
                    300   240   180   150   100  ;
                    450   400   320   0     0    ;
                    450   350   300   0     0    ;
                    2000  1750  1500  1000  0    ;
                    4000  3200  1800  0     0    ;
                    3000  2400  2200  0     0    ;
                    3500  0     0     0     0    ;
                    3000  2000  1750  1500  1000 ;
                    4000  3200  1800  0     0    ;
                    3000  2400  2200  0     0    ];




Quality_options=      [ 100   90    86    75    63;
                        98    87    81    77    60;
                        100   80    62     0     0;
                        99    74    59     0     0;
                        100   93    77    61     0;
                        95    76    59     0     0;
                        97    70    61     0     0;
```

```
                         95   83   75   68   61;
                         100  97   81   71   63;
                         94   79   63    0    0;
                         96   72   61    0    0;
                         99   89   70   62    0;
                         99   73   60    0    0;
                         100  79   63    0    0;
                         100   0    0    0    0;
                         97   89   81   72   67;
                         98   73   62    0    0;
                         98   75   63    0    0;];

DBD=zeros(popsize,Activities);

for i=1:numRows
for j=1:numCols

DBD(i,j)=Dur_options(j,pop(i,j));

end
end

CPM=zeros(popsize,Activities);
S=0;
for i=1:numRows
CPM(i,1)=DBD(i,1)+S;
CPM(i,2)=DBD(i,2)+S;
CPM(i,3)=DBD(i,3)+S;
CPM(i,4)=DBD(i,4)+S;
CPM(i,5)=CPM(i,1)+DBD(i,5);
CPM(i,6)=CPM(i,1)+DBD(i,6);
CPM(i,7)=CPM(i,5)+DBD(i,7);
CPM(i,8)=CPM(i,6)+DBD(i,8);
CPM(i,9)=CPM(i,6)+DBD(i,9);
CPM(i,10)=DBD(i,10)+max(CPM(i,6),CPM(i,2));
CPM(i,11)=DBD(i,11)+max(CPM(i,7),CPM(i,8));
CPM(i,12)=DBD(i,12)+max(CPM(i,5),max(CPM(i,9),CPM(i,10)));
CPM(i,13)=DBD(i,13)+CPM(i,3);
CPM(i,14)=DBD(i,14)+max(CPM(i,10),CPM(i,4));
CPM(i,15)=DBD(i,15)+CPM(i,12);
CPM(i,16)=DBD(i,16)+max(CPM(i,14),CPM(i,13));
CPM(i,17)=DBD(i,17)+max(CPM(i,14),max(CPM(i,11),CPM(i,15)));
CPM(i,18)=DBD(i,18)+max(CPM(i,16),CPM(i,17));
end

while iga<maxit
DBC=zeros(popsize,Activities);
for i=1:numRows
for j=1:numCols
DBC(i,j)=Cost_options(j,pop(i,j));
end
end

RawCost=zeros(numRows,1);
for i=1 : numRows
for j=1 : numCols
RawCost (i,1)= RawCost (i,1)+DBC(i,j) ;
end
end

LastPop=pop;
LastPop(:,19)=0;
LastPop(:,20)=0;
```

```
LastPop(:,19)=CPM(:,18);

for i=1:numRows
Beta=200;
I=120;
if  CPM(i,18) <121
Beta=0;
end
if  CPM(i,18) >121
I=0;
end
LastPop(i,20)= RawCost (i,1)+ (IC* LastPop (i,19))+Beta*( CPM(i,18)
-DueDate)- I*(DueDate- CPM(i,18));
end

QDB=zeros(popsize,Activities);
for i=1:numRows
for j=1:numCols
QDB(i,j)=Quality_options(j,pop(i,j));
end
end

MinQualityOfRows=zeros(numRows,1);
AverageQualityOfRows=zeros(numRows,1);
for  i=1: numRows
j=1;
while MinQualityOfRows(i)==0
MinQualityOfRows(i)=QDB(i,j);
j=j+1;
end

for j=1: numCols
AverageQualityOfRows(i)=AverageQualityOfRows(i)+QDB(i,j)/numCols;
if QDB(i,j)<MinQualityOfRows(i)
if QDB(i,j)~=0
MinQualityOfRows(i)=QDB(i,j);
end
end
end
end

LastPop(:,21)=0;
LastPop(:,21)=MinQualityOfRows(:,1)/4+
(3/4)*AverageQualityOfRows(:,1);
LastPop(:,22)=0;
Pfuzzy= zeros(1, numRows);
for i=1 : numRows
Pfuzzy(1,i)=( (MaxDur - LastPop(i,19))/( MaxDur - MinDur))+((MaxCost
- LastPop(i,20))/ (MaxCost - MinCost))+( (LastPop(i,21)-
MinQuality)/( MaxQuality- MinQuality));
LastPop(i,22)=Pfuzzy(1,i);
end

Poptemp=LastPop;
Poptemp=sortrows(Poptemp,22);
minimum=Poptemp(1, 22);
for i=1:popsize
if minimum== LastPop (i,22)
RecordPopRow=i;
end
end

ElitRow=pop(RecordPopRow,:);
```

```
if iga == 1
Paretofirst= LastPop(RecordPopRow,22);
end
if iga > 1
Paretosecond= LastPop(RecordPopRow,22);
end
output(iga)= LastPop(RecordPopRow,22);
Paretodifference=abs(Paretosecond-Paretofirst);
if Paretodifference < 0.01 && iga > 1
countf=countf+1;
fonk(ElitRow);
for i=1: 18
ElitRow(i)= globalvar(1,i);
end
Paretofirst=globalpareto;
else
Paretofirst=Paretosecond;
end
R1_2_compare=zeros(itNumberTournament,2);
for i=1: itNumberTournament
True=0;
while True==0
True=1;
RNG1=0;
RNG2=0;
while RNG1==RNG2
RNG1=(1+round((popsize-1)*rand));
RNG2=(1+round((popsize-1)*rand));
end

for i=1: itNumberTournament
if RNG1==R1_2_compare(i,1);
if RNG2==R1_2_compare(i,2);
True=0;
end
end
if RNG2==R1_2_compare(i,1);
if RNG1==R1_2_compare(i,2);
True=0;

end
end
end
end

R1_2_compare(i,1)=RNG1;
R1_2_compare(i,2)=RNG2;
x=LastPop(RNG1,22);
y=LastPop(RNG2,22);

if x>y
pop(RNG2,:)=pop(RNG1,:);
end
if y>x
pop(RNG1,:)=pop(RNG2,:);
end
end

DBD=zeros(popsize,Activities);
for i=1:numRows
for j=1:numCols
DBD(i,j)=Dur_options(j,pop(i,j));
end
```

```
end

CPM=zeros(popsize,Activities);

S=0;
for i=1:numRows
CPM(i,1)=DBD(i,1)+S;
CPM(i,2)=DBD(i,2)+S;
CPM(i,3)=DBD(i,3)+S;
CPM(i,4)=DBD(i,4)+S;
CPM(i,5)=CPM(i,1)+DBD(i,5);
CPM(i,6)=CPM(i,1)+DBD(i,6);
CPM(i,7)=CPM(i,5)+DBD(i,7);
CPM(i,8)=CPM(i,6)+DBD(i,8);
CPM(i,9)=CPM(i,6)+DBD(i,9);
CPM(i,10)=DBD(i,10)+max(CPM(i,6),CPM(i,2));
CPM(i,11)=DBD(i,11)+max(CPM(i,7),CPM(i,8));
CPM(i,12)=DBD(i,12)+max(CPM(i,5),max(CPM(i,9),CPM(i,10)));
CPM(i,13)=DBD(i,13)+CPM(i,3);
CPM(i,14)=DBD(i,14)+max(CPM(i,10),CPM(i,4));
CPM(i,15)=DBD(i,15)+CPM(i,12);
CPM(i,16)=DBD(i,16)+max(CPM(i,14),CPM(i,13));
CPM(i,17)=DBD(i,17)+max(CPM(i,14),max(CPM(i,11),CPM(i,15)));
CPM(i,18)=DBD(i,18)+max(CPM(i,16),CPM(i,17));
end

R_compare=zeros(itNumberCrossover,genNumber);
gen_box=zeros(1,genNumber);
gen_temp=zeros(1,genNumber);
R1_2_compare=zeros(itNumberCrossover,2);

for etap=1: itNumberCrossover
for k=1:genNumber
Truegen=0;
while Truegen==0
Truegen=1;
RNGen=(1+round((numCols-2)*rand));
for i=1:genNumber
if RNGen==gen_box(1,i)
Truegen=0;
end
end
end
gen_box(1,k)=RNGen;
end

True=0;
while True==0;
True=1;
RNG1=0;
RNG2=0;
while RNG1==RNG2
RNG1=(1+round((popsize-1)*rand));
RNG2=(1+round((popsize-1)*rand));
end

for i=1: itNumberCrossover
if RNG1==R1_2_compare(i,1);
if RNG2==R1_2_compare(i,2);
True=0;
end
end
if RNG2==R1_2_compare(i,1);
```

```
if RNG1==R1_2_compare(i,2);
True=0;
end
end
end
end

R1_2_compare(etap,1)=RNG1;
R1_2_compare(etap,2)=RNG2;

for i=1:genNumber
gen_temp(1,i)=pop(RNG1,gen_box(1,i));
end



for i=1:genNumber
pop(RNG1,gen_box(1,i))=pop(RNG2,gen_box(1,i));
end

for i=1:genNumber
pop(RNG2,gen_box(1,i))=gen_temp(1,i);
end
end

DBD=zeros(popsize,Activities);
for i=1:numRows
for j=1:numCols
DBD(i,j)=Dur_options(j,pop(i,j));
end
end

CPM=zeros(popsize,Activities);
S=0;
for i=1:numRows
CPM(i,1)=DBD(i,1)+S;
CPM(i,2)=DBD(i,2)+S;
CPM(i,3)=DBD(i,3)+S;
CPM(i,4)=DBD(i,4)+S;
CPM(i,5)=CPM(i,1)+DBD(i,5);
CPM(i,6)=CPM(i,1)+DBD(i,6);
CPM(i,7)=CPM(i,5)+DBD(i,7);
CPM(i,8)=CPM(i,6)+DBD(i,8);
CPM(i,9)=CPM(i,6)+DBD(i,9);
CPM(i,10)=DBD(i,10)+max(CPM(i,6),CPM(i,2));
CPM(i,11)=DBD(i,11)+max(CPM(i,7),CPM(i,8));
CPM(i,12)=DBD(i,12)+max(CPM(i,5),max(CPM(i,9),CPM(i,10)));
CPM(i,13)=DBD(i,13)+CPM(i,3);
CPM(i,14)=DBD(i,14)+max(CPM(i,10),CPM(i,4));
CPM(i,15)=DBD(i,15)+CPM(i,12);
CPM(i,16)=DBD(i,16)+max(CPM(i,14),CPM(i,13));
CPM(i,17)=DBD(i,17)+max(CPM(i,14),max(CPM(i,11),CPM(i,15)));
CPM(i,18)=DBD(i,18)+max(CPM(i,16),CPM(i,17));
end

Mutation=(popsize*Activities);
NumOfMutations=floor(Mutation* mutrate);
MutationGen=zeros(NumOfMutations,2);
PosibilityOfContractors=0;
for etap=1:NumOfMutations
True=0;
while True==0
True=1;
```

```
RNG1=(1+round((popsize-1)*rand));
RNG2=(1+round((Activities-1)*rand));

if RNG2==15
True=0;
end
for k=1: etap
if RNG1==MutationGen(k,1);
if RNG2==MutationGen(k,2);
True=0;
end
end
end

end

MutationGen(etap,1)=RNG1;
MutationGen(etap,2)=RNG2;
Gen=pop(MutationGen(etap,1),MutationGen(etap,2));
if RNG2 == 1 | RNG2 == 2 | RNG2 == 8 | RNG2 == 9 | RNG2 == 16
PosibilityOfContractors= ActivitiesPossibility_4;
end
if RNG2 == 5 | RNG2 == 12
PosibilityOfContractors= ActivitiesPossibility_3;
end
if RNG2 == 3 | RNG2 == 4 | RNG2 == 6 | RNG2 == 7 | RNG2 == 10  |
RNG2 == 11 | RNG2 == 13 | RNG2 == 14 | RNG2 == 17 | RNG2 == 18
PosibilityOfContractors= ActivitiesPossibility_2;
end

True=0;
while True==0;
True=1;
NewGen=round(1+( PosibilityOfContractors-1)*rand);
if NewGen==Gen
True=0;
end
if NewGen ~= Gen;
Gen=NewGen;
end
end
pop(MutationGen(etap,1),MutationGen(etap,2))=Gen;
end

DBD=zeros(popsize,Activities);
for i=1:numRows
for j=1:numCols
DBD(i,j)=Dur_options(j,pop(i,j));

end
end
CPM=zeros(popsize,Activities);
S=0;
for i=1:numRows
CPM(i,1)=DBD(i,1)+S;
CPM(i,2)=DBD(i,2)+S;
CPM(i,3)=DBD(i,3)+S;
CPM(i,4)=DBD(i,4)+S;
CPM(i,5)=CPM(i,1)+DBD(i,5);
CPM(i,6)=CPM(i,1)+DBD(i,6);
CPM(i,7)=CPM(i,5)+DBD(i,7);
CPM(i,8)=CPM(i,6)+DBD(i,8);
CPM(i,9)=CPM(i,6)+DBD(i,9);
```

```
CPM(i,10)=DBD(i,10)+max(CPM(i,6),CPM(i,2));
CPM(i,11)=DBD(i,11)+max(CPM(i,7),CPM(i,8));
CPM(i,12)=DBD(i,12)+max(CPM(i,5),max(CPM(i,9),CPM(i,10)));
CPM(i,13)=DBD(i,13)+CPM(i,3);
CPM(i,14)=DBD(i,14)+max(CPM(i,10),CPM(i,4));
CPM(i,15)=DBD(i,15)+CPM(i,12);
CPM(i,16)=DBD(i,16)+max(CPM(i,14),CPM(i,13));
CPM(i,17)=DBD(i,17)+max(CPM(i,14),max(CPM(i,11),CPM(i,15)));
CPM(i,18)=DBD(i,18)+max(CPM(i,16),CPM(i,17));
end

CPMtemp2=CPM;
CPMtemp2=sortrows(CPMtemp2,18);
minimum=CPMtemp2(1, Activities);
maximum=CPMtemp2(popsize, Activities);

for i=1:popsize;
if maximum==CPM(i,Activities);
RecordPopRow2=i;
end
end
pop(RecordPopRow2,:)=ElitRow;

if work140==1;
for i=1:numRows
DBCone=zeros(1,numCols);
DBDone=zeros(1,numCols);
DBCsum=0;
S=0;
LastCost=0;
for j=1:numCols
DBCone (j)=Cost_options(j,pop(i,j));
DBCsum= DBCsum+ DBCone (j);
end
for j=1:numCols
DBDone(j)=Dur_options(j,pop(i,j));
end
CPM(i,1)=DBDone (1)+S;
CPM(i,2)=DBDone (2)+S;
CPM(i,3)=DBDone (3)+S;
CPM(i,4)=DBDone (4)+S;
CPM(i,5)=CPM(i,1)+DBDone (5);
CPM(i,6)=CPM(i,1)+DBDone (6);
CPM(i,7)=CPM(i,5)+DBDone (7);
CPM(i,8)=CPM(i,6)+DBDone (8);
CPM(i,9)=CPM(i,6)+DBDone (9);
CPM(i,10)=DBDone (10)+max(CPM(i,6),CPM(i,2));
CPM(i,11)=DBDone (11)+max(CPM(i,7),CPM(i,8));
CPM(i,12)=DBDone (12)+max(CPM(i,5),max(CPM(i,9),CPM(i,10)));
CPM(i,13)=DBDone (13)+CPM(i,3);
CPM(i,14)=DBDone (14)+max(CPM(i,10),CPM(i,4));
CPM(i,15)=DBDone (15)+CPM(i,12);
CPM(i,16)=DBDone (16)+max(CPM(i,14),CPM(i,13));
CPM(i,17)=DBDone (17)+max(CPM(i,14),max(CPM(i,11),CPM(i,15)));
CPM(i,18)=DBDone(18)+max(CPM(i,16),CPM(i,17));

Beta=200;
I=120;
if  CPM(i,18) <121
Beta=0;
end
if  CPM(i,18) >121
I=0;
```

```
end

LastCost= DBCsum+ (IC* CPM(i,18))+Beta*( CPM(i,18)-DueDate)-
I*(DueDate- CPM(i,18));
RandomRow=pop(i,:);
while LastCost >140000
DBCone=zeros(1,numCols);
DBDone=zeros(1,numCols);
DBCsum=0;
S=0;
LastCost=0;
for k=1:numCols
if k == 1 | k == 2 | k == 8 | k == 9 | k == 16
PosibilityOfContractors= ActivitiesPossibility_4;
end
if k == 5 | k == 12
PosibilityOfContractors= ActivitiesPossibility_3;
end
if k == 3 | k == 4 | k == 6 | k == 7 | k == 10  | k == 11 | k == 13
| k == 14 | k == 17 | k == 18
PosibilityOfContractors= ActivitiesPossibility_2;
end

if k==15
RandomRow(k)=1;
else
RandomRow(k)=round(1+( PosibilityOfContractors-1)*rand);
end
end

for j=1:numCols
DBCone (j)=Cost_options(j,RandomRow(j));
DBCsum= DBCsum+ DBCone (j);
end

for j=1:numCols
DBDone(j)=Dur_options(j,RandomRow(j));
end

CPM(i,1)=DBDone (1)+S;
CPM(i,2)=DBDone (2)+S;
CPM(i,3)=DBDone (3)+S;
CPM(i,4)=DBDone (4)+S;
CPM(i,5)=CPM(i,1)+DBDone (5);
CPM(i,6)=CPM(i,1)+DBDone (6);
CPM(i,7)=CPM(i,5)+DBDone (7);
CPM(i,8)=CPM(i,6)+DBDone (8);
CPM(i,9)=CPM(i,6)+DBDone (9);
CPM(i,10)=DBDone (10)+max(CPM(i,6),CPM(i,2));
CPM(i,11)=DBDone (11)+max(CPM(i,7),CPM(i,8));
CPM(i,12)=DBDone (12)+max(CPM(i,5),max(CPM(i,9),CPM(i,10)));
CPM(i,13)=DBDone (13)+CPM(i,3);
CPM(i,14)=DBDone (14)+max(CPM(i,10),CPM(i,4));
CPM(i,15)=DBDone (15)+CPM(i,12);
CPM(i,16)=DBDone (16)+max(CPM(i,14),CPM(i,13));
CPM(i,17)=DBDone (17)+max(CPM(i,14),max(CPM(i,11),CPM(i,15)));
CPM(i,18)=DBDone(18)+max(CPM(i,16),CPM(i,17));

Beta=200;
I=120;
if  CPM(i,18) <121
Beta=0;
end
```

```
if  CPM(i,18) >121
I=0;
end
LastCost= DBCsum+ (IC* CPM(i,18))+Beta*( CPM(i,18)-DueDate)-
I*(DueDate- CPM(i,18));
end
pop(i,:)= RandomRow;
end
end

iga=iga+1;
end

DBD=zeros(popsize,Activities);
for i=1:numRows
for j=1:numCols
DBD(i,j)=Dur_options(j,pop(i,j));
end
end

CPM=zeros(popsize,Activities);

S=0;
for i=1:numRows
CPM(i,1)=DBD(i,1)+S;
CPM(i,2)=DBD(i,2)+S;
CPM(i,3)=DBD(i,3)+S;
CPM(i,4)=DBD(i,4)+S;
CPM(i,5)=CPM(i,1)+DBD(i,5);
CPM(i,6)=CPM(i,1)+DBD(i,6);
CPM(i,7)=CPM(i,5)+DBD(i,7);
CPM(i,8)=CPM(i,6)+DBD(i,8);
CPM(i,9)=CPM(i,6)+DBD(i,9);
CPM(i,10)=DBD(i,10)+max(CPM(i,6),CPM(i,2));
CPM(i,11)=DBD(i,11)+max(CPM(i,7),CPM(i,8));
CPM(i,12)=DBD(i,12)+max(CPM(i,5),max(CPM(i,9),CPM(i,10)));
CPM(i,13)=DBD(i,13)+CPM(i,3);
CPM(i,14)=DBD(i,14)+max(CPM(i,10),CPM(i,4));
CPM(i,15)=DBD(i,15)+CPM(i,12);
CPM(i,16)=DBD(i,16)+max(CPM(i,14),CPM(i,13));
CPM(i,17)=DBD(i,17)+max(CPM(i,14),max(CPM(i,11),CPM(i,15)));
CPM(i,18)=DBD(i,18)+max(CPM(i,16),CPM(i,17));

end

DBC=zeros(popsize,Activities);
for i=1:numRows
for j=1:numCols
DBC(i,j)=Cost_options(j,pop(i,j));
end
end

RawCost=zeros(numRows,1);
for i=1 : numRows
for j=1 : numCols
RawCost (i,1)= RawCost (i,1)+DBC(i,j) ;
end
end
LastPop=pop;
LastPop(:,19)=0;
LastPop(:,20)=0;
LastPop(:,19)=CPM(:,18);
```

```
for i=1:numRows
Beta=200;
I=120;
if  CPM(i,18) <121
Beta=0;
end
if  CPM(i,18) >121
I=0;
end
LastPop(i,20)= RawCost (i,1)+ (IC* LastPop (i,19))+Beta*( CPM(i,18)-
DueDate)- I*(DueDate- CPM(i,18));
end

QDB=zeros(popsize,Activities);
for i=1:numRows
for j=1:numCols
QDB(i,j)=Quality_options(j,pop(i,j));
end
end

MinQualityOfRows=zeros(numRows,1);
AverageQualityOfRows=zeros(numRows,1);
for  i=1: numRows
j=1;
while MinQualityOfRows(i)==0
MinQualityOfRows(i)=QDB(i,j);
j=j+1;
end

for j=1: numCols
AverageQualityOfRows(i)=AverageQualityOfRows(i)+QDB(i,j)/numCols;
if QDB(i,j)<MinQualityOfRows(i)
if QDB(i,j)~=0
MinQualityOfRows(i)=QDB(i,j);
end
end
end

end

LastPop(:,21)=0;
LastPop(:,21)=MinQualityOfRows(:,1)/4+
(3/4)*AverageQualityOfRows(:,1);
LastPop(:,22)=0;
Pfuzzy= zeros(1, numRows);

for i=1 : numRows

Pfuzzy(1,i)=( MaxDur - LastPop(i,19))/( MaxDur - MinDur)+(MaxCost -
LastPop(i,20))/ (MaxCost - MinCost)+( LastPop(i,21)- MinQuality)/(
MaxQuality- MinQuality)

LastPop(i,22)=Pfuzzy(1,i);

end
```

APPENDIX B

```
function y=fonk(row);
Temperature= 200;
StopTemperature=1;
E=0.01;
npar=18;
Activities=18;
IC=50;
Beta=200;
I=120;
DueDate=121;


Dur_options=[14 15 16 21 24;
             15 18 20 23 25;
             15 22 33 0  0 ;
             12 16 20 0  0 ;
             22 24 28 30 0 ;
             14 18 24 0  0 ;
             9  15 18 0  0 ;
             14 15 16 21 24;
             15 18 20 23 25;
             15 22 33 0  0 ;
             12 16 20 0  0 ;
             22 24 28 30 0 ;
             14 18 24 0  0 ;
             9  15 18 0  0 ;
             16 0  0  0  0 ;
             20 22 24 28 30;
             14 18 24 0  0 ;
             9  15 18 0  0 ];

Cost_options=[2400  2150  1900  1500  1200 ;
              3000  2400  1800  1500  1000 ;
              4500  4000  3200  0     0    ;
              45000 35000 30000 0     0    ;
              20000 17500 15000 10000 0    ;
              40000 32000 18000 0     0    ;
              30000 24000 22000 0     0    ;
              220   215   200   208   120  ;
              300   240   180   150   100  ;
              450   400   320   0     0    ;
              450   350   300   0     0    ;
              2000  1750  1500  1000  0    ;
              4000  3200  1800  0     0    ;
              3000  2400  2200  0     0    ;
              3500  0     0     0     0    ;
              3000  2000  1750  1500  1000 ;
              4000  3200  1800  0     0    ;
              3000  2400  2200  0     0    ];





Quality_options=[ 100  90   86   75   63;
                  98   87   81   77   60;
```

```
                 100   80    62     0     0;
                 99    74    59     0     0;
                 100   93    77    61     0;
                 95    76    59     0     0;
                 97    70    61     0     0;
                 95    83    75    68    61;
                 100   97    81    71    63;
                 94    79    63     0     0;
                 96    72    61     0     0;
                 99    89    70    62     0;
                 99    73    60     0     0;
                 100   79    63     0     0;
                 100    0     0     0     0;
                 97    89    81    72    67;
                 98    73    62     0     0;
                 98    75    63     0     0;];


ActivitiesPossibility_1=1;
ActivitiesPossibility_2=3;
ActivitiesPossibility_3=4;
ActivitiesPossibility_4=5
Possibility_3=[3, 4, 6, 7, 10, 11, 13,14,17,18];
Possibility_4=[5,12];
Possibility_5=[1, 2, 8, 9,16];

numCols=18;
AtomicStructure=zeros(1, numCols);
AtomicStructuretemp=zeros(1, numCols);

for j=1: 18
AtomicStructure(1,j) = row(j);
end

DBCone=zeros(1,numCols);
DBDone=zeros(1,numCols);
QBDone=zeros(1,numCols);
CPM=zeros(1,numCols);

DBCsum=0;
S=0;
LastCost=0;
LastQuality=0;
time=0;
LastCost2=0;
LastQuality2=0;
AverageQualityOfRows=0;
MinQualityOfRows=0;
MinDur= 104;
MaxDur= 169;
MinCost= 109965;
MaxCost= 178920;
MaxQuality= 97.04;
MinQuality= 62.67;

Fitness=0;

DBCsum=0;

for j=1:numCols
DBCone (j)=Cost_options(j,AtomicStructure(j));
```

```
DBCsum= DBCsum+ DBCone (j);
end

for j=1:numCols
DBDone(j)=Dur_options(j,AtomicStructure(j));
end

AverageQualityOfRows=0;
for j=1:numCols
QBDone(j)=Quality_options(j,AtomicStructure(j));
AverageQualityOfRows= AverageQualityOfRows+ QBDone(j)/18;
end

MinQualityOfRows =QBDone(1);
for j=1:numCols
if QBDone(j)< MinQualityOfRows
MinQualityOfRows =QBDone(j);
end
end

LastQuality= MinQualityOfRows/4+ (3/4)*AverageQualityOfRows;

CPM(1)=DBDone (1)+S;
CPM(2)=DBDone (2)+S;
CPM(3)=DBDone (3)+S;
CPM(4)=DBDone (4)+S;
CPM(5)=CPM(1)+DBDone (5);
CPM(6)=CPM(1)+DBDone (6);
CPM(7)=CPM(5)+DBDone (7);
CPM(8)=CPM(6)+DBDone (8);
CPM(9)=CPM(6)+DBDone (9);
CPM(10)=DBDone (10)+max(CPM(6),CPM(2));
CPM(11)=DBDone (11)+max(CPM(7),CPM(8));
CPM(12)=DBDone (12)+max(CPM(5),max(CPM(9),CPM(10)));
CPM(13)=DBDone (13)+CPM(3);
CPM(14)=DBDone (14)+max(CPM(10),CPM(4));
CPM(15)=DBDone (15)+CPM(12);
CPM(16)=DBDone (16)+max(CPM(14),CPM(13));
CPM(17)=DBDone (17)+max(CPM(14),max(CPM(11),CPM(15)));
CPM(18)=DBDone(18)+max(CPM(16),CPM(17));

Beta=200;
I=120;
if  CPM(18) <121
Beta=0;
end
if  CPM(18) >121
I=0;
end

LastCost = DBCsum+ (IC* CPM(18))+Beta*( CPM(18)-DueDate)-
I*(DueDate- CPM(18));
time=CPM(18);

Fitness2=(MaxDur - (CPM(18) )/( MaxDur - MinDur))+(( MaxCost-
LastCost)/ (MaxCost - MinCost))+((LastQuality-MinQuality)/(
MaxQuality- MinQuality));

BestFitness=Fitness;
BestAtomicStructure=AtomicStructure;
BestCost=LastCost;
```

```
BestTime=time;
BestQuality=LastQuality;
i=0;

while  Temperature>StopTemperature
i=i+1;
Fitness2=0;
DBCsum=0;

AtomicStructuretemp= AtomicStructure;

RNG1=15;
Possibility=0;
Possibility2=0;

while RNG1==15
RNG1=(1+round((numCols-1)*rand));
end

for j=1:10
if Possibility_3(j) == RNG1
Possibility=3;
end
end

for j=1:2
if Possibility_4(j) == RNG1
Possibility=4;
end
end

for j=1:5
if Possibility_5(j) == RNG1
Possibility=5;
end
end

True=0;

while True==0

RNG2=15;
while RNG2==15
RNG2=(1+round((numCols-1)*rand));
end

for j=1:10
if Possibility_3(j) == RNG2
Possibility2=3;
end
end



for j=1:2
if Possibility_4(j) == RNG2
Possibility2=4;
end
end
```

```
for j=1:5
if Possibility_5(j) == RNG2
Possibility2=5;
end
end

if Possibility2== Possibility
True=1;
end

if Possibility2> Possibility
if AtomicStructuretemp(RNG2)<= Possibility;
True=1;
end

if Possibility2< Possibility
if AtomicStructuretemp(RNG1)<= Possibility2;
True=1;
end
end
end

end



Gen=0;
Gen=AtomicStructuretemp(RNG1);
AtomicStructuretemp(RNG1)= AtomicStructuretemp(RNG2);
AtomicStructuretemp(RNG2)=Gen;



DBCsum=0;

for j=1:numCols
DBCone (j)=Cost_options(j, AtomicStructuretemp(j));
DBCsum= DBCsum+ DBCone (j);
end

for j=1:numCols
DBDone(j)=Dur_options(j, AtomicStructuretemp(j));
end
AverageQualityOfRows=0;

for j=1:numCols
QBDone(j)=Quality_options(j, AtomicStructuretemp(j));
AverageQualityOfRows= AverageQualityOfRows+ QBDone(j)/18;
end
MinQualityOfRows =QBDone(1);
for j=1:numCols
if QBDone(j)< MinQualityOfRows
MinQualityOfRows =QBDone(j);
end
end

LastQuality= MinQualityOfRows/4+ (3/4)*AverageQualityOfRows;

CPM(1)=DBDone (1)+S;
CPM(2)=DBDone (2)+S;
CPM(3)=DBDone (3)+S;
CPM(4)=DBDone (4)+S;
```

```
CPM(5)=CPM(1)+DBDone (5);
CPM(6)=CPM(1)+DBDone (6);
CPM(7)=CPM(5)+DBDone (7);
CPM(8)=CPM(6)+DBDone (8);
CPM(9)=CPM(6)+DBDone (9);
CPM(10)=DBDone (10)+max(CPM(6),CPM(2));
CPM(11)=DBDone (11)+max(CPM(7),CPM(8));
CPM(12)=DBDone (12)+max(CPM(5),max(CPM(9),CPM(10)));
CPM(13)=DBDone (13)+CPM(3);
CPM(14)=DBDone (14)+max(CPM(10),CPM(4));
CPM(15)=DBDone (15)+CPM(12);
CPM(16)=DBDone (16)+max(CPM(14),CPM(13));
CPM(17)=DBDone (17)+max(CPM(14),max(CPM(11),CPM(15)));
CPM(18)=DBDone(18)+max(CPM(16),CPM(17));

Beta=200;
I=120;
if  CPM(18) <121
Beta=0;
end
if  CPM(18) >121
I=0;
end

LastCost = DBCsum+ (IC* CPM(18))+Beta*( CPM(18)-DueDate)-
I*(DueDate- CPM(18));

Fitness2=(MaxDur - (CPM(18) )/( MaxDur - MinDur))+(( MaxCost-
LastCost)/ (MaxCost - MinCost))+((LastQuality-MinQuality)/(
MaxQuality- MinQuality));

if  Fitness2 < Fitness
Fitness=Fitness2;
AtomicStructure= AtomicStructuretemp;

else
Uphill=0;
Delta=0;
Delta=Fitness-Fitness2;
Uphill=exp(Delta/Temperature);

Rnd=rand;

if  Uphill > Rnd
Fitness=Fitness2;
AtomicStructure= AtomicStructuretemp;

end
end



if BestFitness > Fitness

BestFitness=Fitness;
end

Temperature=Temperature-E*Temperature;
end

global globalvar;
```

```
for i=1: 18
globalvar(1,i) = AtomicStructure(1,i);
end
global globalpareto
globalpareto=BestFitness;
```

CURRICULUM VITAE

Name Surname: Yashar Armaghani

Place and Date of Birth: IRAN – 24 November 1982

E-Mail: edu@armaghani.com

B.Sc.: Civil Engineering, Azad University, Mahabad, Iran, 2007

Associate of Sc.: Computer Software, Comprehensive Scientific & Applied University, Tabriz, Iran (2004).

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS:

G. E. Gürcanlı, A. G. Özışık, A. A. Tatar, B. Özaslan, Y. Armaghani, Ö. Aydın. 2012. *Türkiye İnşaat Sektöründe Yapı Makinaları Kullanımı Üzerine Bir Araştırma*, 2. Proje ve Yapım Yönetimi Kongresi, 13 – 16 Eylül 2012İzmir Yüksek Teknoloji Enstitüsü, Urla-İzmir.