A HYBRID GENETIC ALGORITHM FOR MULTI MODE RESOURCE CONSTRAINED SCHEDULING PROBLEM FOR LARGE SIZE PROJECTS

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

BY

MUSTAFA GÜREL

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

SEPTEMBER 2015

Approval of the thesis:

A HYBRID GENETIC ALGORITHM FOR MULTI MODE RESOURCE CONSTRAINED SCHEDULING PROBLEM FOR LARGE SIZE PROJECTS

Submitted by **MUSTAFA GÜREL** in partial fulfillment of the requirements for the degree of **Master of Science in Civil Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver

Dean, Graduate School of **Natural and Applied Sciences** _______________

Prof. Dr. Ahmet Cevdet Yalçıner

Head of Department, **Civil Engineering** _______________

Assoc. Prof. Dr. Rifat Sönmez

Supervisor, **Civil Engineering Dept., METU** _______________

Examining Committee Members:

Prof. Dr. Mustafa Talat Birgönül

Civil Engineering Dept., METU _______________

Assoc. Prof. Dr. Rifat Sönmez

Civil Engineering Dept., METU _______________

Asst. Prof. Dr. Aslı Akçamete Güngör

Civil Engineering Dept., METU _______________

Asst. Prof. Dr. Güzide Atasoy Özcan

Civil Engineering Dept., METU _______________

Asst. Prof. Dr. Önder Halis Bettemir

Civil Engineering Dept., İnönü University _______________

Date: 03.09.2015

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name	**:**
Signature	:

ABSTRACT

A HYBRID GENETIC ALGORITHM FOR MULTI MODE RESOURCE CONSTRAINED SCHEDULING PROBLEM FOR LARGE SIZE PROJECTS

Gürel, Mustafa
M.S., Department of Civil Engineering
Supervisor: Assoc. Prof. Dr. Rifat Sönmez

September 2015, 96 Pages

Just like in all industries, some of the available resources, in order to finish a project on time, are constrained in construction industry. To be able to finish the project on time has high importance both for the contractor and for the owner. Project scheduling in which resources are limited for a particular time are called as resource constrained project scheduling problems (RCPSP) and occupies a significant place in construction management. Especially for large scale projects, little success has been achieved for solving multi-mode RCPSP. In the context of this thesis, RCPSPs with multiple execution modes for activities, are aimed to be solved. With Hybrid Genetic Algorithm (HGA) proposed in this thesis, finding the optimal solutions for large size construction projects with multiple execution modes and resource constraints is aimed. The proposed hybrid algorithm consists of two parts, first part is a heuristic method and second part is a hybrid genetic algorithm. In the first part, some of the population is generated with a heuristic method so that search domain of the algorithm can be concentrated on better results. In second part, Hybrid Genetic Algorithm, solutions are improved with each generation. The performance of the algorithm is verified with the examples available in the literature. The main contribution of this algorithm is that it enables the large sized real life projects to be solved with resource constraints in a fast and efficient way.

Keywords: Project Management and Scheduling, Multiple Modes, Resource Constrained, Genetic Algorithms, Hybrid Genetic Algorithm

ÖZ

BÜYÜK ÖLÇEKLİ PROJELERDE ÇOK MODLU KAYNAK KISITLI İŞ PROGRAMLAMA PROBLEMİ İÇİN GELİŞTİRİLMİŞ BİR HİBRİT GENETİK ALGORİTMA

Gürel, Mustafa

Yüksek Lisans, İnşaat Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Rifat Sönmez

Eylül 2015, 96 Sayfa

Tüm sektörlerde olduğu gibi, inşaat sektöründe de, projelerin bitirilmesi için gereken bazı kaynaklar sınırlıdır. Projelerin zamanında bitirilebilmesi hem müteahhit hem de işveren açısından yüksek önem taşımaktadır. İnşaat yönetiminde önemli bir yere sahip olan, kısıtlı kaynakların bulunduğu proje planlama problemlerine, kaynak kısıtlı proje planlama problemi (KKPPP) denir. Özellikle büyük ölçekli projeler için, çok modlu KKPPP'nin çözümünde ufak başarılara ulaşılmıştır. Bu tezin içeriğinde, çok uygulama modlu aktiviteleri içeren KKPPP'ler çözülmektedir. Bu tezde önerilen melez genetik algoritma (MGA) ile büyük ölçülü inşaat projelerinde, kaynak kısıtları ve farklı uygulama modları dikkate alınarak, optimal bitiş sürelerinin bulunması amaçlanmıştır. Bahsedilen melez algoritma, sezgisel yöntem ve melez genetik algoritma olarak iki bölümden oluşmaktadır. İlk bölümde, algoritmanın popülasyonun bir kısmı sezgisel yöntem ile oluşturulmaktadır, bu sayede, algoritanın arama alanı daha iyi sonuçlar üzerinde yoğunlaştırılmıştır. İkinci bölüm olan Melez Genetik Algoritma bölümünde, sonuçlar her jenerasyonda daha da iyileştirilmektedir. Literatürde bulunan örnekler ile algoritmanın performansı test edilmiş ve algoritmanın etkinliği gösterilmiştir. Bu algoritmanın ana katkısı, kaynak kısıtlı, büyük ölçekli, gerçek projelerin hızlı ve etkili bir şekilde çözülmesine olanak sağlamaktır.

Anahtar Kelimeler: Proje Yönetimi ve Planlaması, Çoklu Modlar, Kaynak Kısıtlı, Genetik Algoritmalar, Melez Genetik Algoritma

Dedicated to my family and my beloved friend, İzel Akdeniz

ACKNOWLEDGEMENTS

During the journey of two years in my Master's degree, I have felt many different emotions. I have faced many challenges during my courses and during the thesis work in Construction Engineering and Management Division of Civil Engineering Department of Middle East Technical University. During this period, I enjoyed learning new concepts and making great effort in order to be successful. I would not be able to overcome the difficulties I have faced without the suggestions and support of the academic members and my friends.

First of all, I would like to show my gratitude to my supervisor, Assoc. Prof. Dr. Rifat Sönmez, who has always supported me and gave his valuable advice throughout the Master studies. Without his support and help, I would not be able to finish this thesis and reach good results.

I specially thank my dear family. During this journey, they have never doubted me and made me smile. Both my parents; Ayşe Gül Gürel and Faruk Gürel, and my brother, Kaan Gürel, has supported me and kept my morale high.

I want to show my appreciation and gratitude to my dear friend, İzel Akdeniz. Without her, it would be very difficult for me to be able to complete this thesis work. She have always kept a smile on her face even when I was stressful and desperate.

Finally, I thank all of my friends for their continuous support for their friendship.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
AIA	Artificial Immune Algorithm
AoA	Activity on Arrow
AoN	Activity on Node
APD	Average Percent Deviation
CP	Constrained Programming
CPM	Critical Path Method
CPU	Central Processing Unit
DE	Differential Evolution
EFT	Early Finish Time
EST	Early Start Time
GA	Genetic Algorithm
GB	Gigabyte
GHz	Gigahertz
HGA	Hybrid Genetic Algorithm
LFT	Late Finish Time
LST	Late Start Time
Np-Hard	Non-Polynomial Hard
PSO	Particle Swarm Optimization
RAM	Random Access Memory
RCPSP	Resource Constrained Project Scheduling Problem
RLP	Resource Levelling Problem
SA	Simulated Annealing
TCT	Time-Cost Tradeoff
TF	Total Float
USD	United States Dollar

CHAPTER 1

INTRODUCTION

In today's world, construction projects with different varieties take one of the biggest share in terms of investment. With developing technology and increase in the number of qualified engineers, competitiveness between the contractors from all around the world has risen. This lead contractors to give bids with less profit so that they could secure the project. Since the profits earned in construction works are not as high as it used to be, contractors searched new ways so that they could decrease the overall costs and increase the total profit of the project. This search lead contractors to work and emphasize on planning and scheduling.

The most common method in terms of planning and scheduling is the Critical Path Method (CPM). In nature, CPM calculates the total project time with the given time input of the activities of the schedule. The logical relationships between activities, lag times, working hours in a day, working calendars and resource availabilities are some of the main parts of preparing an appropriate schedule. The schedule of the works should be done with utmost care and detail, otherwise, the project may not be finished on the planned time which may lead to dissatisfied owner, financial losses, disputes between shareholders or even a lawsuit which may affect the company greatly and may lead the company to bankruptcy.

Although CPM is the most commonly used scheduling technique in the construction industry, it is not proficient at finding the optimal solution in project when resource constraints are available. Due to this reason, resource constrained project scheduling problem (RCPSP) has been given great importance. The main objective of the RCPSP is to reach the minimum project duration while satisfying both resource and precedence constraints. In real life situations, all of the construction projects may have resource constraints since no construction company has infinite resource access. Even if the construction company has great number of resources, directing all of the resources to a single project will increase the cost of that project and will place a burden to other works that the company undertakes. This phenomenon makes RCPSP very significant in terms of management and planning of the construction projects. Although RCPSP is important, popular commercial project management software has very limited capabilities for solving the RCPSP (Mellentien and Trautmann 2001; Hekimoglu 2007; Lu et al. 2008; Bettemir and Sonmez 2014).

Many construction projects have more than 300 activities in real life examples. As the project gets larger, costs increase proportionally. In order to maximize the profit obtained at the end of projects, they should be finished as early as possible. This means that daily overhead costs of the project will be decreased. Many of the RCPSP studies used problem examples up to 60 activities which does not reflect the nature of construction projects. Hegazy and Menesi (2014) has worked with a Constraint Programming (CP) method for large size examples up to 2000 activities.

Many of the designed heuristics and meta-heuristics for RCPSP, cannot be performed with real life construction projects which have more than 300 activities. Moreover, methods that are able to solve the RCPSP for large scale problems, require too much computation time in order to reach quality solutions. The CP model that is designed by Hegazy and Menesi (2014), has been applied to instances that have, 10, 100, 500, 1000, 1500 and 2000 activities. CP model has reached a solution with %9.61 deviation from upper bound (best known solution of the instance) in 3 hours in the largest project instance. As a consequence, in terms of RCPSP, the gap between literature and real life examples is remarkable.

This thesis aims to develop a Hybrid Genetic Algorithm (HGA) so that high quality solutions in RCPSP can be obtained with shorter computational time. To be able to reach high quality solutions in shorter computational time is significant for real life cases so that with better solutions, minimizing the overhead costs and resources can be done efficiently and the project may be finished earlier as efficiently as possible.

In ordinary Genetic Algorithms, the initial population is usually generated randomly. In the proposed HGA, some of the initial population is generated randomly and some is generated with a heuristic method, in which backward scheduling is implemented with respect to resource constraints. In other words, in heuristic algorithm part, improved solutions are generated. After generating the initial population, the algorithm implements forward scheduling to all solutions and calculates makespan values. The solutions are implemented with crossover, mutation, elitism and renewal processes. The algorithm ends when the stopping criterion, schedule number in this case, is met.

The order of thesis is as follows; after the introduction part, literature review continues as chapter 2. Chapter 3 is about genetic algorithms and their content, then comes the Chapter 4 in which the HGA is explained and analysis results are given. After the conclusion part in Chapter 5, the references are given.

CHAPTER 2

LITERATURE REVIEW

In this chapter, first Critical Path Method (CPM) is explained to give understanding of most widely used scheduling technique. Resource constrained project scheduling problem (RCPSP) is defined with some of the proposed optimization techniques.

2.1. Critical Path Method

Throughout the paper, CPM is defined as the most widely used scheduling technique. The first step to understand what CPM does is to understand what scheduling is. Scheduling is consideration of the timing and order of the activities available in a project in order to decide the total time when the project is completed (Mubarek 2010).

In a CPM network, just like any scheduling method, there are some steps to reach a successful schedule. First of all, the work activities that will generate the network, are determined. After the determination of the activities, their corresponding durations are assigned respectively and the logical relationships (precedence, successor relationships) are determined. After these steps are successfully performed, schedule network is drawn.

CPM is the most commonly used network analysis method and the main objective is to find the longest path throughout the schedule network. The illustration of the network can be drawn by two different procedures, which are; activity on arrow (AoA) and activity on node (AoN). Although both procedures reach the same results, AoN method is preferred by many. Some of the reasons include the widespread adaptation of AoN method to computer systems. AoN network is also clearer to understand, early start time (EST), early finish time (EFT), late start time (LST), late finish time (LFT) and total float (TF) of the activities can be seen on the network. Furthermore, activity on node process has become the most common and outstanding procedure for scheduling (Lock 2007).

As their name imply, EST and LST are the earliest and the latest times that an activity can start. The same logic is valid for EFT and LFT, which are the earliest and latest times that an activity can finish. TF can be explained as the amount of time an activity can be delayed without hindering the completion date of the network. TF is calculated by implementing forward pass and backward pass calculations. In forward pass by keeping activity relationships and their durations, EST and EFT values are determined. The calculation begins from the first activity and ends with the last one. In backward pass, LST and LFT values are determined, the process starts from the last activity and finishes at the first one. The backward pass can only be commenced only if the forward pass is finished.

Critical path is the longest path in terms of completion time in a project, thus, it determines the total project completion time. Each of the activities in the critical path have TF of zero. This means that none of the activities can be delayed since it affects the total project and results with a delay. The activities located in critical path are called the critical activities.

Each project has a definite finish time decided by the client. Planning and scheduling is done so that the project may be finished on time. The completion of the project may end up later than the given finish date if the resource constraints and implementation of activities are not taken into consideration. This problem can be overcome by RCPSP methods.

2.2. Resource Constrained Project Scheduling Problem

There is no project in world that the resources and budget are infinite. A construction company determines the budget and the amount of resources for each project so that every project can be completed without any problems.

In RCPSP, the main objective is to find the optimum duration by satisfying the resource constraints. In other words, with definite resources, the project is completed as soon as possible.

RCPSP falls into nondeterministic polynomial-time hard (NP-hard) (Blazewicz et al. 1983; De et al. 1997) computational complexity class. Exact methods for these RCPSP's can solve only for small size or medium to small size projects. In the light of these reasons, numerous heuristic and meta-heuristic methods are designed and proposed in order to reach optimal solution in RCPSP. These methods include priority rule based scheduling heuristics (Özdamar and Ulusoy 1994; Hegazy et al. 2000; Tormos and Lova 2001), and meta-heuristics, including genetic algorithms (Chan et al. 1996; Hartmann 1998; Hegazy 1999; Kim and Ellis 2008; Chen and Shahandashti 2009; Kim and Ellis 2010; Sonmez and Uysal 2014), simulated annealing (Lee and Kim 1996; Bouleimen and Lecocq 2003; Valls et al. 2005), tabu search (Deblaere et. al. 2011), and particle swarm optimization (Lu et al. 2008; Wang and Qi 2009; Chen 2011).

Throughout the project, contractor spends money from the budget of the project. The costs of the contractor is divided into two categories. These categories are named as direct costs and overhead costs. Direct costs are the material costs, labor costs, equipment, and machinery costs. Overhead costs are expenses that cannot be grouped into any specific work item of direct costs. They include mobilization costs, salaries of indirect personnel such as cook and security, general security costs, cost of bonds taken from banks, site office expenses, insurance expenses and many more. As the project duration increases overhead costs increase and direct costs decrease. Overhead costs, in other words, indirect costs are assumed to have linear relationship with the duration of project. To define the overhead costs as linear, costs that are paid in a single time such as bonds and insurance expenses are excluded from daily overhead cost assumptions.

With the help of RCPSP, even if the direct costs cannot be decreased, overhead costs are aimed to be lowered by completing project earlier while satisfying resource constraints. With each day a project is completed earlier, the contractor may increase the profit by not spending any indirect cost.

2.3. Exact, Heuristic and Meta-Heuristic Methods

As mentioned in chapter 2.2, RCPSP examples are categorized as Non-Polynomial hard (NP-hard) problems in which reaching the optimal solution takes great time and long computation is needed. Due to these reasons, many algorithms are proposed in the literature to overcome the long computational time of this problem. As the technology develops and coding language becomes more familiar and user friendly, many optimization methods are implemented into various researches. RCPSP can be solved with exact, heuristic and meta-heuristic algorithms.

Exact methods aim to find the global optimal solution while searching the solution space of the problem. Moreover, enormous amounts of computations are needed in order to reach the optimal solution for large problems. Consequently, as the amount of computation gets higher, the time required to reach the optimal solution gets higher. Exact solution methods also need technologically developed computer systems since they require big amounts of computations. Some of the most widely used exact methods include linear programming, dynamic programming, branch-and-bound method and mixed integer programming. Although exact methods are the most reliable optimization method and has the guarantee of reaching the optimal solution, they are not fully practical to be used in construction projects. Since many construction project schedules include more than 300 activities, trying to solve RCPSP with exact methods will take great time.

Different than the exact methods, heuristic algorithms requires much less amount of computations and proficient at reaching solutions in much small times when compared with exact methods. Heuristic methods use simple calculations and procedures to solve problems. The main drawback of heuristic methods when it is compared with exact methods is that finding the optimal solution is not guaranteed. Near-optimal solutions are reached with the help of heuristic algorithms. Even though the solutions may not be the optimal, they are branded as satisfactory solutions. The reason is that, heuristic methods are able to reach near-optimal solutions, which do not differ greatly from the optimal, in very small time.

Lastly, meta-heuristic algorithms are those which are inspired by stochastic (probabilistic) incidents seen in nature. Meta-heuristic algorithms are different from heuristic algorithms in terms of problem dependency. Heuristic problems are designed as problem dependent, algorithm is coded specifically to solve a certain problem. In meta-heuristic algorithms, the process of finding the solution is independent of the nature of the problem. In this type of algorithms, evolution process is copied and the problem is solved throughout an iterative random search technique. With the help of this iterative search, prevention of being stuck around a local optima is aimed. Just like heuristic methods, however, meta-heuristic methods does not guarantee reaching the optimum solution. They also reach near-optimum solutions in short time with simple calculations. Some of the most widely used meta-heuristic methods are; ant colony optimization (ACO), simulated annealing (SA), particle swarm optimization (PSO) and genetic algorithms (GA).

2.4. Heuristic Methods for RCPSP

In literature, many heuristic methods are implemented to overcome RCPSP. Some examples are priority rule based scheduling heuristics of Hegazy et al. (2000), Tormos and Lova (2001) and methods that works on time-cost tradeoffs (TCT) with RCPSP, such as; Ahn and Erenguc (1998) and Hegazy and Menesi (2012).

In the work of Ahn and Erenguc (1998), a heuristic method is applied to solve RCPSP with multimode resource alternatives, in other words, crashable modes. A multi pass algorithm is designed for nonpreemptive resource constrained projects. The objective function is to find a feasible project schedule that has the minimum project cost. In the method, delay penalty cost is also taken into consideration. Two main stages are available in the proposed heuristic method. In the first stage, a feasible schedule is generated and in the following stage, the aim is to improve the feasible schedule. In the first stage, modes that have the cheapest normal costs are selected for each activity. The normal duration is assigned as the duration of the selected modes. Cost and duration of the activities are fixed and problem is approached as a single mode RCPSP. Makespan optimization is applied with minimum total float (slack) rule. In second stage, six different procedures are followed to improve the feasible schedule. In first improvement procedure, an activity is aimed to be rescheduled with changing mode, duration and finish time of that activity if it can be rescheduled with less cost, without hindering the feasibility of the project schedule. In procedure two, two adjacent activities are aimed to be rescheduled. In this process, completion time of the project is not affected. Two activities are rescheduled in order to have less cost than previous mode selection. In procedure 3, crashing is implemented only if the cost of crashing of the selected activity is less than the penalty cost that is caused by delay of the project. Opposite to procedure 3, uncrashing is implemented in procedure 4. The uncrashing is implemented to an activity if the completion time of the project is not affected. In procedure 5, again uncrashing of activities is implemented. However, procedure 5 deals with uncrashing that changes the completion time of the project. If the reduction in activity costs due to uncrashing of that activitiy is greater than penalty costs due to delays, uncrashing is accepted. In the last procedure, two different activities are taken into consideration, one is applied with crashing and the other with uncrashing. If the costs caused by crashing is less than reduction of costs caused by uncrashing, the procedure is accepted. Different project cases with activity numbers of 10, 12, 14, 16, 18, 20, 25, 30, 40 and 50 are studied.

In the heuristic algorithm of Hegazy et al. (2000), a case study obtained from Talbot and Patterson (1979) has been studied. The case study consists of a schedule with 20 activities and six different resources with their corresponding availabilities. In other words, the case study is a RCPSP with single modes having different resource requirements. In the algorithm, resource substitution rules are applied to overcome RCPSP. It is assumed that each different resource has some substitution rule with other resources, in other words, a single resource of R1 is equal to 2 resources of R2. Carpenter and steel worker example is illustrated to further increase the understanding. According to Hegazy et al. (2000), a carpentry work may be carried out with a single carpenter with good efficiency, moreover, steel workers may also carry out the carpentry work. However, their productivity will be much lower than the carpenter. Consequently, a carpentry work can be finished at the same time with a single carpenter or several steel workers. With the help of resource substitution rules, under allocation of one resource settles the problem of over allocation of another.

In the heuristic work of Tormos and Lova (2001), a hybrid multi pass method in which random sampling procedures are combined with backward-forward method. The heuristic method is implemented into three sets of projects. First set is named as j30 set. This set consists of 480 projects each project having 30 activities with 4 resource types. The second set is the j60 set, in which 480 projects, each having 60 activities with 4 resource type, are located. Last set is the j120 set, in which 600 projects with 120 activities with 4 resource types, are located. In other words, heuristic method is implemented to different projects with activity numbers of 30, 60 and 120.

Hegazy and Menesi (2012) have designed a heuristic method for multimode RCPSP as an add-in tool to Microsoft Project software. First step is to check whether the activity crashing is promising or not. If the found duration is less than the project completion deadline the schedule is found as promising, if the case is the opposite, more crashing options are tried. After the schedule is found as promising, the method basically follows three steps, in first step, all of the start delays, lag times, are taken as zero, in second step, activity that can be crashed in the cheapest way is selected and in the third step, resource constraints are applied to current schedule. The methodology continues until the project duration with the lowest cost is found or no more crashing of activities is possible. After the initial check of promising schedules, the modes of all activities are arranged so that cheapest mode of each activity would be selected. Beyond this point, the method continues differently with changing conditions. If all of activities in the critical path are crashed, the least expensive non-critical activity is selected and crashed. If all of the critical activities are not crashed, least expensive critical activity is selected and crashed. With current crashing options, resource constraints are applied and total cost of the project is calculated and if applicable, delay penalties are added. If the current solution is better than previous one in terms of cost, it is labeled as the best solution. If all of the activities are not crashed, the method returns to start point and checks for crashing options, if all of the activities are crashed, best solution is worked on. In the best solution, possible uncrashing of modes are searched in order to decrease the total cost. If the uncrashed mode of a chosen activity gives lower or the same total cost, the uncrashed mode is selected. The best solution is presented after the algorithm stops. The proposed heuristic is applied to projects with activities of 9, 25, 90, 180 and 360 with multi-mode selections, which represent the crashing modes. The bigger projects are created by adding the project with 9 activities several times.

2.5. Meta-Heuristic Methods for RCPSP

Chan et al. (1996) have developed a genetic algorithm (GA) to solve RCPSP. The main advantage of meta-heuristics such as GA over heuristic methods is that heuristic methods proved that they are problem dependent and unable to solve bigger problems. In GA method, algorithm creates a random set of initial population. Each schedule is represented as chromosomes in the population. After the population is created, fitness of the solutions are evaluated. The better solutions of the population are selected and paired randomly so that new generation of solutions can be created by using Crossover operator. Besides crossover, another operator called mutation is implemented and a randomly selected gene in a randomly selected chromosome is applied to mutation, in other words, altered. This process continues until the stopping criterion is met. The analysis and experiments are made with projects that have activity numbers ranging from 11 to 51.

In the work of Mori and Tseng (1997), the ability of GA and a stochastic scheduling method (Drexl and Gruenewald, 1993) is compared with respect to finding solutions in multi-mode RCPSP. The proposed GA method starts initialization of the population. The activities are arranged with respect to their activity numbers in ascending order. The mode of each activity is selected in a random manner and the scheduling priority of the activities are determined randomly. Makespan for each solution is calculated.

Solutions are, then, arranged in the order of increasing project duration. This way population starts with the best solutions and ends with the worst ones. In the applied one point crossover, two parents are selected. The first parent is the best solution in the population and the second parent is chosen randomly from the other solutions. A junction activity is selected and mode assignments and scheduling priorities of that activity and its predecessors are taken from the first parent. The unselected activity priorities and mode assignments are taken from the other parent. A solution is also selected in a random manner in mutation. Random number of activities are selected and mode assignment of the selected activities are changed. Generation of the new population is as follows; first the solution with the minimum makespan is preserved just like in elitism, then, offspring from mutation and crossover are taken into consideration. Lastly, randomly generated new offspring are taken into consideration. Comparisons of the GA and stochastic method are done with projects of 20, 30, 40, 50, 60 and 70 activities. The work of Mori and Tseng (1997) showed that GA has reached good results especially in larger project instances.

Hartmann (1998) also proposed a GA method for RCPSP. The used GA is similar with ordinary ones. In this GA, an initial population is created, solutions, that is, each member of population, is evaluated and sent to crossover operator, after crossover, mutation is implemented. Again, this process takes on until the stopping criterion is reached. The main difference of the GA proposed by Hartmann (1998) is that GA follows a permutation based genetic encoding which contains problem-specific knowledge instead of following a priority rule based representation. In permutation based genetic encoding, each individual, solution, is represented by an activity sequence. The sequence is assumed as a precedence feasible permutation of the set of activities. Then, as decided by the work sequence each activity is scheduled to their earliest feasible time. Whereas, in priority based encoding, each individual is represented by priority values, each value of activity having a number between 1 and total number of activities. The activities are scheduled one by one by following precedence relationship and priority values. GA in this study is applied to problem instances with 30 and 60 activities with single-modes having four different resources.

Hegazy (1999) introduced improved resource allocation and levelling heuristics within a GA model. In heuristic part of algorithm, each activity is given a random priority and the impact of these priorities are controlled. In iterative procedure of improved resource allocation heuristic, first a schedule is obtained by giving all activities the lowest priority. The algorithm changes the priority of the first activity to highest priority and the new project duration is calculated. If the duration is not improved, the priority of the selected activity is returned to lowest priority value and the next activity is taken into consideration. This process is done until all the activities in the schedule are worked with. After all activities are considered, the iterative procedure ends and algorithm continues with GA method. Experiments are conducted on networks with 20, 40, 100 and 200 activities. The larger networks are obtained by adding 20 activity of the smallest network several times. Each activity has 6 different resources. The algorithm is coded using the macro language of Microsoft Project.

Hartmann (2001) has developed a GA for multi-mode RCPSP. In his method, he first enhances the initial population as described by Sprecher et al. (1997). The algorithm will spend time only in promising areas in the search space. In this improvement method, the main idea is that in an optimum schedule, no activity is completed with inefficient modes. In other words, schedules with activities that has inefficient modes are omitted from the search space. After improving the solution space, GA starts with initialization of population. The initialization procedure has three steps. At first step, a mode is selected and assigned to all activities in the schedule. At second stage, mode selections are controlled in terms of non-renewable resource feasibility. If non-renewable resource constraint would be exceeded in the schedule, an activity is selected randomly and its mode assignment is changed so that its mode assignment will differ from previous choice. This procedure is done until all the activity mode assignments are changed or until the schedule becomes feasible with respect to resource constraints. At third stage, mode selection of activities are temporarily fixed. Then by taking the precedence feasibility into consideration, activities are ordered into activity list. After initialization, each individual undergoes fitness evaluation process with respect to their makespan. Then, crossover and mutation process starts. In crossover, from two selected parent individuals, two offspring are created. Parents are named as mother and father, the offspring are called as daughter and son. In crossover method two numbers are selected as q_1 and q_2 which are smaller than total activity number of the project. The crossover both handles the activity priority and mode selection. For instance, in the case of daughter, activities starting from the first activity to number q_1 are taken from the mother, and activities starting from q_1+1 to the last activity are taken from the father. Similar to this procedure, in terms of crossover of mode assignments, number q_2 is used. This time, mode assignments from the first activity to q_2 are taken from the mother, and mode assignments from q_2+1 to the last are taken from the father. In the case of son, the procedure is the same, this time information until point q_1 and q2 is taken from the father and information beyond q_1 and q_2 is taken from the mother. Mutation operator in this method has 2 modifications. In first modification, the location of a selected activity in the activity list is swapped with the next one if it satisfies the precedence constraint. The second modification deals with the mode selection. A mode assignment is randomly changed. This way a mode selection which is not present in the current population may occur. The individuals in the algorithm are improved with various methods. In the proposed single pass method, mode selection of each activity is checked. The main aim is to select a mode for an activity so that it will be completed in a less time. If applicable, this procedure is applied to all of the activities, each activity is considered only once. After single pass improvement, a second improvement method named as multi pass improvement is implemented. The single pass procedure is applied repeatedly in this procedure since improvement, left shift, in an activity may end up with a decrease in non-renewable resource constraints which creates a possibility of an improvement on another activity. The proposed GA is coded in ANSI C and tested under LINUX operating system. Project instances of 10, 12, 14, 16, 18, 20 and 30 non-dummy activities are used in their experiments.

Alcaraz et al. (2003) have also developed a GA for RCPSP. Same as Hartmann they have implemented enhancement designed by Sprecher et al. (1997). At initialization part, not only feasible schedules but also infeasible schedules in terms of non-renewable resources are taken into the population. To create population, first, for each activity a mode is selected randomly. If the mode selection is infeasible, just as in the work of Hartmann (2001), a random activity is selected and assigned a new random mode selection. This procedure is done until a feasible schedule is achieved. In crossover operation, two point crossover is applied and a two-phase mutation is used. Similar to Hartmann (2001), at first stage an activity in the activity priority list is relocated between the locations of its predecessors and successors. In second stage, each activity changes its mutation selection with a predetermined mutation probability. In each generation, a random replacement method is applied. With a predetermined probability, each individual in a population is replaced with a randomly generated new individual. This contributes to variability of the population and prevention of premature convergence. The GA is coded in Microsoft Visual C++ language. The experiments are done with projects of 10, 12, 14, 16, 18 and 20 number of activities, each activity having 3 modes, two non-renewable resources and two renewable resources. To test the performance of the algorithm in large problems, project instances of 50 and 100 activities are also used.

Chen and Shahandashti (2009) have developed a hybrid algorithm in which GA and Simulated Annealing (SA) processes are combined. By combining two generic search methods such as GA and SA, it is aimed that GA-SA Hybrid to be applicable to all optimization problems. The flow of the algorithm is as follows; initial population is generated and the temperature, a used factor in SA, is initialized. Two solutions are selected with a decreasing probability of selecting less-fitted solutions in crossover, and later, mutation is performed with a decreasing mutation rate. If the produced offspring from crossover and mutation operators is better than the worst solution in the population, it is replaced with the worst solution. However, if the produced offspring is not better-fitted than the worst solution the procedure changes. The produced offspring is replaced with worst solution only if it the Metrepolis's criterion (Metrepolis et al. 1953) is satisfied. In the proposed hybrid algorithm, multi-project scheduling is undertaken, 3 real life projects with 45, 46 and 39 number of activities respectively.

In literature, many different GA models are available to overcome RCPSP. Chen and Weng (2009) have designed a two phase GA model for RCPSP. In first phase, the resource constraint problem is overcome and in the second phase, TCT problem is dealt with. At the first step, initial population is created. In first population, genes represent the mode selection of activities. After initialization part, resource scheduling starts. For each activity, selected modes are marked. And with respect to selected modes, a new population is created where priority of the activities are represented with genes until the termination condition is reached, crossover and mutation procedures are implemented. Best fitted solutions are evaluated and with roulette wheel selection parent individuals are selected for crossover operation. The crossover operation in this method is one-point crossover. After crossover, uniform mutation is applied so that randomness in the population will not be lost. At the end of this process, shortest duration and the corresponding total cost, that is, the best solution for the chromosome representing activity modes, is stored. This process is applied to all of the chromosomes representing the chosen activity modes. After the priority solutions are decided, the GA procedure is applied to chromosomes with mode selection. This way, two different GA procedures are implemented in order to overcome RCPSP with TCT. At first GA procedure, activity priorities are determined and the best solutions are stored. In the second GA procedure, activity mode selections are determined. The method is implemented into two different projects, one having 10 activities and the other having 37 activities.

Lova et al. (2009) have proposed a hybrid GA for the purpose of finding optimal solutions in Multi-mode RCPSP. At the start of their algorithm they implement an improvement process to individuals of the population so that search space of the algorithm can be limited. The improvement procedure used was first introduced by Sprecher et al. (1997) which was also implemented in the work of Hartmann (2001). When initial improvements are made, the GA part starts. GA follows an ordinary procedure. The initial population is generated with respect to improvement method and randomness. After creating the initial population, fitness of each solution is calculated. In crossover operation, two point crossover is implemented. In the crossover operation, parent chromosomes are called as the father and the mother. From parent individuals, two different offspring is generated, these are named as son and daughter chromosomes. Two points are selected in the chromosomes, for the son chromosome, the part between the selected two points are taken from the mother individual and the parts beyond chosen points are taken from the father. Similar to this procedure, daughter chromosome is generated. The part between the selected points are taken from the father chromosome and, the parts beyond chosen points are taken from the mother. This way, two offspring is created from a pair of parents. When crossover is finished, mutation follows. In the process of mutation, both activity priority and mode selection of activities are mutated. In terms of activity priority, an activity is inserted into a new position in the activity priority list between its latest predecessor and earliest successor with a given probability of P_{mut}. In terms of mode, mutation procedure changes if that activity has feasible non-renewable mode assignment or not. If the selected activity has non-renewable feasible mode assignment, each activity randomly changes its mode assignment with the probability of P_{mut}. If the mode selection of the activity is not feasible in terms of non-renewable resources, the mutation procedure changes. In this case, the mutation is called as massive mutation. Until the non-renewable mode selection in the schedule becomes feasible, the mutation operator randomly selects activities from the activity list and assigns a random mode to selected activities. This process is applied to increase the diversity and variation of mode selection. At the end of mutation, with a tournament selection procedure, new individuals are selected with respect to their fitness to create a new population. Elitism is also applied to ensure the best solution survives in the next population. The solutions are also improved with multi-mode forward-backward method so that feasible project completion time can be reduced. This method is found especially effective when renewable or non-renewable resource constraints exist. The computational experiments are applied to project instances of 10, 12, 14, 16, 18, 20 and 30.

Although the flow of different GAs are similar, in the work of Mendes, Gonçalves and Resende (2009) the process is a bit different. In the proposed GA, in order to tighten the solution space, concept of parameterized activity schedules is used. In this concept, the schedules are divided into three groups; semi-active schedules, active schedules and non-delay schedules. Semi-active schedules are those in which activities are sequenced as early as possible and none of the activities can be scheduled earlier without changing the sequence of activities. In active schedules, activities are again schedules to the earliest possible times, however, this time, they cannot be scheduled earlier without delaying some other activity or without violating precedence constraints. Active schedules are also a type of semi-active schedules. In non-delay schedules, all resources are used if they can trigger the processing of any activity. In other words, in non-delay schedule, all resources are kept idle. Non-delay schedules are also a type of active schedules, therefore, semi-active schedules. To summarize, activity schedules can be defined as sets where non-delay schedules are subset of active schedules and active schedules are subset of semi-active schedules. The main aim of the parameterized activity schedules method is to work on schedules which are either active or non-delay schedules. This way processing time is decreased greatly. The GA part of the algorithm also shows some differences. First, the initial population is created with random number vectors. Chromosome representation includes the information about activity priorities and delay times of the each activity. In fitness evaluation, the main factor is again the makespan, however, another factor is also taken into consideration. If two schedules with the same makespan is evaluated, the one which has less activities ending close to the completion time, is branded more promising in terms of improvement. This process of evaluation is called modified makespan. After evaluation of fitness of each chromosome, crossover starts. Parameterized uniform crossover is implemented. In this crossover type, each gene is taken from the parents with respect to a decision. A random number is given between 0 and 1 to each chromosome for decision. If the given number is smaller than a predetermined value, that gene is taken from the first parent, if the given number is higher than the predetermined value, that gene is taken from the second parent. The main difference comes from the mutation part. In this GA, common mutation techniques are not applied, instead, some of the worst fitted individuals are terminated and randomly created new individuals are inserted. This procedure also prevents premature convergence. The GA ends when termination conditions are met. The experiments are done with test problems having 30, 60 and 120 number of activities. The algorithm is designed with Visual Basic version 6.0. The activities have single-mode resource representation with four resource types.

Many different algorithms are created to solve RCPSP. Differential evolution (DE) is the method proposed by Damak, Jarboui, Siarry and Loukil (2009). DE is an algorithm which is created with the influence of GA. GA changes the characteristics of individuals with crossover and mutation operators. In DE those evolutionary methods are combined with geometric search mechanisms. In DE, the possible solutions are represented with vectors. Each solution has two vectors, the first one representing the activity priorities and the second one representing mode assignments. Different from GA, DE first implements mutation process. In this operation, three individuals are selected. An offspring named as mutant vector is created. Mutant vector is created by combining priority vectors and mode assignment vectors from all 3 parent individuals. After mutation process, crossover operation starts. In this procedure, the mutant offspring created with mutation and a randomly selected individual from the population are implemented with crossover. A randomly selected part of the mutated vector is implemented into the selected individual. The newly created solution is called as trial vector. At the end of mutation and crossover, if the fitness of the trial vector is better than the selected solution from the population, trial vector is inserted and worse solution is terminated. The fitness comparison is done at the end of the crossover so that the waste of time on the less fitted individuals will be prevented. The algorithm is applied to projects of 10, 12, 14, 16, 18 and 20 activities with multi-mode resource assignments.

Although GA is the most commonly used optimization technique in project scheduling, various methods are designed for RCPSP. Artificial immune system, designed by Peteghem and Vanhoucke (2009), is one of the methods. In the proposed algorithm, inspiration is taken from the vertebrate immune system of the body. The immune cells have receptors, by which the disease causing elements are recognized. The immune cells are implemented with clonal proliferation (reproduction) with respect to their ability to fight efficiently with harmful elements. Different mechanisms such as hyper-mutation are applied to the immune cells with lower efficiency in order to reach efficient cells. In the proposed algorithm, the process starts with initialization. After that, clonal selection and affinity maturation starts. The algorithm stops seeking for optimal solution when the stopping criterion is reached. Since the experimented problem has multi-mode resource representations, initial population is created in a controlled manner instead of creating it by assigning random values. The main reason behind this is to emphasize on the promising solution space and decrease the computational time required to find the optimal solution. In controlled generation, mode assignment of activities are done first. The main idea is to select modes so that each of them will be feasible in terms of non-renewable resource usage. With different experiments, mode assignments with lower profit values are found to be more feasible than other selections. After modes are selected, corresponding resource requirements and duration of the activities are detected. After mode assignment is done, activity scheduling order, in other words, activity priority list, is decided. Various techniques are applied to decide the activity list. These include latest finish time, latest start time, minimum slack and maximum remaining work priority rules. The main reason for applying various priority rules is that two different solutions may have the same mode assignment. By applying different rules, having the exact two solution in the population is avoided. When the population generation is finished, each individual is given an affinity value. If the makespan of the solution is high, the affinity value will be low. Proliferation of the solutions are done with respect to affinity value. If a solution has higher affinity value, or lower makespan, it will appear more frequently in the cloned population. When cloning process ends, affinity maturation starts in which hyper-mutation and receptor editing is done. In hyper-mutation, a solution is mutated with respect to its makespan and the best makespan in the population. The solution with a higher makespan will be performed under high mutation rates. Likewise, lower the makespan of the solution, lower the number of mutations it will face. The mutation is both applied to activity list and mode selections. In terms of activity list, an activity is selected randomly and moved into a new position between its successors and predecessors. In mode selection phase, again, a random activity is selected then a random element is selected from the population. The mode assignment of the selected solution is copied to the cloned solution that is facing mutation. In the last part of the algorithm, receptor editing is done. The best fitted solutions are preserved and the unfitted elements are terminated. The newly generated population becomes the start population of the next iteration. The process ends with the stopping condition is met. Experiments are done with projects with 7 different size. The projects have activity numbers of 10, 12, 14, 16, 18, 20 and 30.

A permutation based elitist genetic algorithm with two schemes is designed by Kim and Ellis (2010) to solve RCPSP and determine whether parallel or serial scheme provides better. The genetic algorithm procedure is same as any ordinary GA. The main difference is during the evaluation of fitness values, before deciding the elitist solution and solutions that will be selected by crossover and mutation, are done both with serial scheduling generation and parallel scheduling generation scheme. In serial scheduling scheme, the scheduling of activities are done by following the priority. The activity having higher priority is scheduled if the resource constraints are satisfied. In parallel scheduling scheme, first, the activities that have 0 number of predecessors are decided, then the one with having higher priority is scheduled. The predecessors of the scheduled activity will be taken into consideration in next step and again between the activities that have 0 number of predecessors, one having the highest priority is scheduled. In this meta-heuristic algorithm, mainly the performance difference between parallel and serial scheduling scheme is analyzed. The experiments are done on a real life project which consists of 29 activities.

In Artificial Immune Algorithm of Mobini et al. (2010), RCPSP problem is aimed to be solved with optimization of makespan. In their meta-heuristic algorithm, the process is imitated from the natural immune system of human beings. In the algorithm, biological theories of clonal selection and affinity maturation is imitated. The body reacts to foreign invading substances, which are named as pathogens. The reaction of the immune system to invading pathogens are divided into two concepts, namely; clonal selection and affinity maturation. In clonal selection, it is stated that immune cells that can recognize the invading pathogens will be reproduced in great numbers. Some of the reproduced cells will be effecter cells so that antibodies that can defeat the pathogens can be created in great amounts and the others will become memory cells so that the same pathogen can be dealt with convenience in case of future expositions. In reproduction stage, cells are implemented with mutation process with a selective force with respect to each cells ability to deal with pathogen. Cells that can deal with pathogens in a more efficient way will have high affinity values. The cells with high affinity values will undergo lower mutation rates, whereas, cells with lower affinity will undergo higher mutation rate. The algorithm is designed with respect to this phenomenon. In the algorithm, possible solutions to the problem are called the antibodies. In antibodies, priority of the activities are represented with an order, the activities are listed with respect to their priorities. Just as GA, in AIA initial set of population is generated randomly. To improve the results of the algorithm, initial population is improved with backward-forward method. Fitness of the individuals, antibodies, are evaluated with affinity evaluation process. The affinity of each antibody is inversely proportional with the makespan of that antibody. An antibody having a higher affinity value is more likely to be reproduced. After each antibody is reproduced with respect to their affinities, a two-phase mutation process is implemented to improve the antibodies. In first phase, a single point mutation is implemented. In single point mutation, an activity is selected randomly in the list of antibody. The priority of the selected activity is changed randomly between the priority of its predecessor and successor activities. In two point mutation process, priority each activity is swapped with the one without violating resource constraints. At the end of the mutation process, worst individuals are replaced with randomly generated individuals. The process continues with the new population, and the loop ends when the stopping criterion is met. In this method, projects of 30, 60 and 120 activities are worked.

Peteghem and Vanhoucke (2010) have designed a GA, for RCPSP. In this GA, possible solutions are not represented with chromosomes, instead, they are represented with vectors. Each vector contains information about the activity priority list. The corresponding mode selection of each activity is represented by a mode list. For the sake of allowing activity preemption, the networks is converted into a new one. In this process, if the mode selection of the activity is assigned, in other words, the resource requirements and the duration is known, that activity is split into sub-activities which have a unit duration of one day. Moreover, splitting is applied to single activity at a certain time. After the splitting is done, sub-activity starting times are determined. The generation of schedules from the priority vectors and mode lists is accomplished with

2 methods. First method applies backward-forward scheduling, which is commonly used for Multi-mode RCPSP. The second method includes a procedure in order to improve mode selection. For each activity, the current mode selection is aimed to be improved. The main aim is to have improvement on the non-renewable resource usage. A mode that uses less resources has higher completion time. However, due to precedence constraints, this mode may fit into a place which ends up an earlier finish time of that activity. This procedure checks the modes of all activities to search for a mode that leads to an earlier finish time in that activity. This procedure is implemented to activities for which all sub-activities are scheduled. After these improvement techniques, GA starts. First, individual vectors and mode lists are generated randomly, then, each is implemented into improvement methods. Each individual is evaluated in terms of fitness which is related with the makespan and the non-renewable resource constraints. If a schedule is feasible, has no excessive non-renewable resource usage, the fitness value is taken as the makespan. If the schedule is not feasible, the fitness value is taken as the makespan plus the amount of excessive non-renewable resource usage. The lower the fitness value, the better the schedule in terms of fitness. In crossover, tournament selection is used to determine the parents and one point crossover is implemented where a single point is selected randomly, all the information regarding the activity priority and mode selection, left of that point is taken from the first parent and the information beyond the selected point is taken from the second parent. In mutation, two sub procedures are implemented. In first, a random point in activity priority vector is altered with a predetermined mutation probability, in the second, a random point in mode list is selected and assigned a new value with a predetermined mutation rate. Same as many GAs, the algorithm stops when stopping criterion is fulfilled. The algorithm is designed and compiled with Visual C++ version 6.0. Set of projects of 10, 15, 20, 25 and 30 activities are experimented on.

A hybrid rank-based evolutionary algorithm is developed by Elloumi and Fortemps (2010). The RCPSP has multi-mode resource assignments. In this method, a preprocessing method developed by Sprecher et al. (1997) and used by Hartmann (2001) is used to improve initial solutions. In this method, as mentioned before, all inefficient modes in terms of non-renewable resources are eliminated. After this process, initialization starts with assigning random priorities and random modes with respect to preprocessing procedure to each activity. The fitness evaluation of activities follows a complex procedure. At first penalty values, which are calculated from excess usage of non-renewable resources, and makespan of each individual is determined. Based on penalty values and makespan, a rank-based fitness evaluation is performed. At the last phase, a clustering procedure is implemented to find density values. The fitness of the individuals is calculated with rank and density values. After fitness evaluation is done crossover starts. In crossover, two randomly selected parents, namely, mother and father, are determined. In the crossover operation, a parent is used only once, so, the selected parents are marked. One point crossover is implemented in which, two numbers, q_1 and q_2, are randomly generated between 0 and total number of activities. In terms of activity priority, the points until q_1 are taken from the first parent, and beyond the selected point, priority of activities, which are not selected from the first parent, are taken from the second parent. In mode selection crossover, the procedure is the same. A number q_2 is randomly selected and until the selected number, mode assignments are taken from the first parent and again, beyond that point, mode selections of activities from the second parent are taken into consideration. Mutation operator is applied as two stages, at first stage, a point in activity priority list is selected and location of the selected activity is swapped with the following activity if the precedence rules are not violated. This procedure is done until two activities are permuted or attempts equal to the total number of activities are made. At the second phase of mutation, a randomly selected point in mode list is assigned a new value so that the current mode assignment changes. Roulette wheel selection is used to determine new generation at the end of each iteration. The stopping criteria is determined as number of schedules or a certain processing time. The algorithm is compiled with Microsoft Visual C++ 6.0. Project instances of 10, 12, 14, 16, 18, 20 and 30 activities are used.

Ant colony optimization is a meta-heuristic method firstly proposed by Dorigo (1992). In ACO proposed by Zhang (2012), multi-mode RCPSP is aimed to be solved. Theory of ACO lies in the behavior of ant colonies that search for food. In ACO, solutions are represented with paths. When an ant founds a path leading to food, it gives off a special chemical substance called phermone to form phermone trails. Ants tend to follow the path which has higher phermone deposition. The probability of choosing a path increases with its phermone level. After some time, ants find the shortest path that leads to the food. In each cycle in the algorithm, an ant creates a RCPSP solution by selecting activities in order in a single activity list. According to the probabilities of each activity selection, a phermone value is given to the selection which shows how well the selection is. After each cycle, phermone levels are decreased with evaporation rates to overcome early convergence and stagnation. The process is repeated until the stopping criterion is met. In the work of Zhang (2012), since multimode resource representation is observed, two types of phermones are used, one representing the activity priority and the other representing the selected resource mode. In the algorithm, firstly, phermones of activity priority is worked then the mode selection of the selected activity is studied. The case projects used in experiments have activity numbers of 10 and 18.

Backward-Forward hybrid genetic algorithm created by Sonmez and Uysal (2014) is another meta-heuristic method for RCPSP. In this method, cooling features of SA is also included. The algorithm starts with generating initial population, then, crossover is performed and a new child is obtained from parent individuals. In this algorithm, two types of mutation operator are applied. If there is not sufficient diversification between the parents and the child, the first mutation process is implemented. Backward-forward scheduling is done on mutated child. If the predefined diversification value between the parents and child is met, the mutation is accepted. The second mutation is performed to randomly chosen individuals, again, the diversification between the original and new solution is compared and accepted if the criterion is met. Elitist selection is done to save the best solutions and the process proceeds till the stopping criterion is met. The temperature that is used in diversification calculations is decreased with each loop of iteration. The experiments are carried out with instances having 30, 60 and 120 number of activities.

The last meta-heuristic method that will be covered in this paper is designed by Menesi and Hegazy (2014) and called as constraint programming (CP). CP is used as an advanced mathematical optimization method and CP model is developed with the IBM ILOG modelling software and its CPLEX-CP solver engine. In this method, projects with 10, 100, 500, 1000, 1500 and 2000 number of activities are used. In terms of project size, this method exceeds the other available ones. Another distinction between CP and other methods is that CP implements multimode resource-constraints.

Although, numerous researchers aimed to solve multi-mode RCPSP as shown in Table 2.1., majority of them achieved solutions limited to small size problems. This shows that there is a need for a method that can solve very large problems with multi-mode resource assignments. All of the methods except from CP, emphasized on projects smaller than 500 activities and many of them preferred single-mode RCPSP. However, in real life, an activity can be completed with many different alternatives, of course, the resource requirements and the duration of the activity will change with changing modes. A contractor may complete an activity with more resources if it will end up shortening the total completion date.

Many of the proposed methods also solve RCPSP up to certain point which does not reflect the true nature of construction projects. Majority of the construction projects have schedule networks having more than 300 activities. Although CP model is applied to large problems, the computational time is high and the results of the large problems have high deviation from the best known solutions.

Within this context, there is a need for a method that can solve large networks having more than 300 activities with multimode resource constraints while reaching high quality solutions in short computational time.

Table 2.1. Heuristic and Meta-Heuristic algorithms for RCPSP

Year of Publication	Authors	Method	Problem	# of Activities	Representation of Resources
1997	Mori and Tseng	GA	RCPSP	20, 30, 40, 50, 60 and 70	Multimode Representation
1998	Ahn and Erenguc	Heuristic Method	RCPSP with TCT	10, 12, 14, 16, 18, 20, 25, 30, 40 and 50	Multimode Representation
2001	Hartmann	GA	RCPSP	10, 12, 14, 16, 18, 20 and 30	Multimode Representation
2003	Alcaraz, Maroto and Ruiz	GA	RCPSP	10, 12, 14, 16, 18 , 20, 50 and 100	Multimode Representation
2009	Chen and Weng	Two-phase GA	RCPSP with TCT	10 and 37	Multimode Representation

Table 2.1. Heuristic and Meta-Heuristic algorithms for RCPSP (Continued)

Year of Publication	Authors	Method	Problem	# of Activities	Representation of Resources
2009	Lova, Tormos, Cervantes and Barber	Hybrid Genetic Algorithm (HGA)	RCPSP	10, 12, 14, 16, 18, 20 and 30	Multimode Representation
2009	Damak, Jarboui, Siarry and Loukil	Differential Evolution (DE)	RCPSP	10, 12, 14, 16, 18 and 20	Multimode Representation
2009	Peteghem and Vanhoucke	Artificial Immune Algorithm	RCPSP	10, 12, 14, 16, 18, 20 and 30	Multimode Representation
2010	Peteghem and Vanhoucke	GA	RCPSP	10, 15, 20, 25 and 30	Multimode Representation
2010	Elloumi and Fortemps	Hybrid Evolutionary Algorithm	RCPSP	10, 12, 14, 16, 18, 20 and 30	Multimode Representation

Table 2.1. Heuristic and Meta-Heuristic algorithms for RCPSP (Continued)

Year of Publication	Authors	Method	Problem	# of Activities	Representation of Resources
2012	Hegazy and Menesi	Heuristic Method	RCPSP with TCT	9, 25, 90, 180 and 360	Multimode Representation
2012	Zhang	Ant Colony Optimization (ACO)	RCPSP	10 and 18	Multimode Representation
2014	Menesi and Hegazy	Constraint Programming	RCPSP	10, 100, 500, 1000, 1500 and 2000	Multimode Representation

CHAPTER 3

GENETIC ALGORITHM OPTIMIZATION

In the following chapter, mainly, genetic algorithm optimization technique will be introduced since the proposed method is mainly based on genetic algorithms. The properties of the method, differences to other optimization problems and the procedure will be described in detail.

3.1. Introduction to Genetic Algorithms

Genetic algorithms are stochastic search algorithms which imitate the process of natural selection and genetics. They have been developed by John Holland in the University of Michigan, USA (Goldberg 1989). The main objective of their research was to explain the adaptations observed in nature and to design a software that can use the mechanics of such adaptations. The process of GA is as follows; at the beginning, initial set of random solutions, which are called as the population, are created. Each solution in the population is defined as chromosome or individual. A single chromosome is represented by a string of numbers, usually by binaries. These set of numbers represent different characteristics of the each solution. Throughout the process of GA, chromosomes are evolved through consecutive iterations. Each iteration process is called a generation. In each generation, chromosomes are evaluated and implemented into different operators, namely; crossover and mutation. The next generation of chromosomes are generated with the help of crossover and mutation. At the end of each generation, a fitness evaluation is done to decide which individuals will be terminated and which will be taken to next generation. This termination is done to keep the population size constant and to provide the algorithm with the ability to converge into better individuals.

3.2. Structure of GA

As stated in the introduction, GA follows a procedure to reach the best solution. This procedure is as follows;

- An initial set of solutions is created.
- An evaluation of the fitness of the chromosomes are made.
- Chromosomes are applied with genetic operators named as; crossover and mutation.
- With a selection procedure, best fitted solutions are pointed and used in the next generation.

The general structure of GA can be seen in Figure 3.1.

3.3. Initialization

In GA, mainly two different procedures are available for generating the initial population. First way is to create initial population with randomness. This way, the individuals will be created randomly while satisfying the constraints and parameters of the algorithm. By creating the individuals randomly, the algorithm covers a huge search space, the fitness of created individuals have the probability of the best and the worst. Second way to create individuals is by implementing a heuristic initialization procedure. By applying heuristic method at initialization, the algorithm starts the search around the better solutions obtained with heuristic method. This contributes the GA to find the optimum or aimed solutions in a much faster time. This contribution is great in large scale networks, the computational time may be decreased greatly compared to random initialization. The main disadvantage of heuristic initialization is that, since only a certain part of the solution space is explored, the algorithm may be stuck at local maximum or local minimum points, in other words, the algorithm may

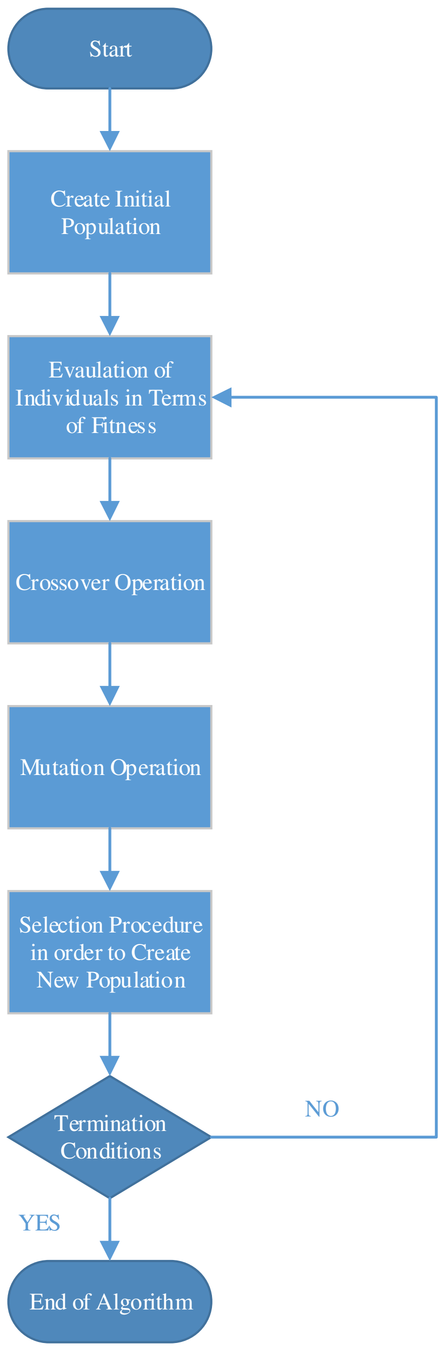

Start
Create Initial Population
Evaulation of Individuals in Terms of Fitness
Crossover Operation
Mutation Operation
Selection Procedure in order to Create New Population
Termination Conditions
NO
YES
End of Algorithm

Figure 3.1. General Structure of Genetic Algorithm

not converge to the optimum solution. To overcome this problem and let the algorithm benefit from random and heuristic initialization, an encoding procedure is applied.

3.4. Encoding

As stated before, the individuals, chromosomes, in the population represent the probable solutions of a problem in GAs. The properties and parameters of the network are implemented into chromosomes in a such way that, in later steps of the algorithm, just by checking the implemented data in the chromosome, the network properties and information of that solution can be managed. The process of implementation of network properties into chromosomes is called encoding. The process of reading the chromosome and converting the encoded data into network characteristics and properties is called decoding. In GA encoded solutions are usually called as Genotype, likewise, the decoded solutions are called Phenotype. Since GA has roots both in natural sciences and computational sciences, the terminologies of both science fields are used.

There are several methods used for encoding. These are; binary encoding, real number encoding, integer or literal permutation encoding and lastly, a general data structure encoding. For instance, in the work of Holland, the encoding is applied with binary encoding. In the research of Holland, it is found that using binary encoding may not be appropriate for function optimization problems. The reason is the existence of Hamming cliffs. Hamming cliffs can be explained in the following way. A two solution that are close in terms of network characteristics are close to each other in phenotype, decoded solution, space. However, when these networks are encoded into chromosomes, they may have great distance in terms of binary representation just as in Figure 3.2. The distance between the chromosomes in genotype, encoded solution, space is called the Hamming distance. The Hamming cliff is the phenomenon, where the distance of between two points is large in genotype space while it is minimal in phenotype space.

0	1	1	1	1	1	1

1	0	0	0	0	0	0

Figure 3.2. Pair of Individuals with Hamming Cliff

Findings of researches show that, real number encoding is the best method in order to solve optimization problems.

When structure of the encoding is taken into consideration, the classification of encoding can be made with two types, namely; one-dimensional encoding and multi-dimensional encoding. Also, classification can be made according to the contents of encoded data. This classification is, again, divided into two types, namely; solution only encoding and solution+parameters encoding. The first way is generally used in optimization problems to fully design an appropriate encoding to the problem. The second way is used in evolution strategies (Rechenberg 1973, Schwefel 1995), where the first part of encoding holds the properties of solution and second part holds strategy parameters which represents the variances of normal distribution for mutation process.

3.5. Evaluation of Fitness

In GA, in each generation, individuals are compared with each other to decide which individuals are better fitted in terms of objective function. To decide the fitness of individuals, fitness evaluation is done. To form a better population with every generation, a selection procedure is applied. This selection is based on the fitness of activities, the better fitted activities have more probability of being chosen to be in the next generation of population. In other words, fitness evaluation is a significant part of GA so that the chromosomes that will be terminated or the chromosomes that will be reproduced in next generation are decided.

3.6. Genetic Algorithm Operators

GA imitates the natural mechanism of adaptation. Operators are used to imitate the creation of new individuals so that the population in GA can improve with newly generated child chromosomes. In GA, three operators are used; these are crossover, mutation and selection.

3.6.1. Crossover

Crossover is the main operator in GA. The performance of GA is greatly affected by the performance of crossover. In crossover operation, two individuals, chromosomes are selected. While imitating the reproduction of new individuals in nature, crossover creates new offspring by combining both of the parent chromosomes' characteristics.

In each GA, crossover operation has some probability. The crossover probability is the amount of offspring that will be placed into the population. As the probability of crossover increases, the performance of GA increases since by increased crossover rate, algorithm will explore the wider space in solution domain. However, if the crossover rate is too high, too much computational time will be needed, also, many less fitted solution space will be searched. All in all, an optimum crossover rate should be found so that it will contribute greatly to the ability of reaching the optimum solution.

Three types of crossover is mainly used in GA. These are called, one-point crossover, two-point crossover and uniform crossover.

In one-point crossover, a single point is marked in the chromosome. The part until the marked point is taken from one parent chromosome and the part coming after the marked point is taken from the other. Representation of one-point crossover is shown in Figure 3.3.

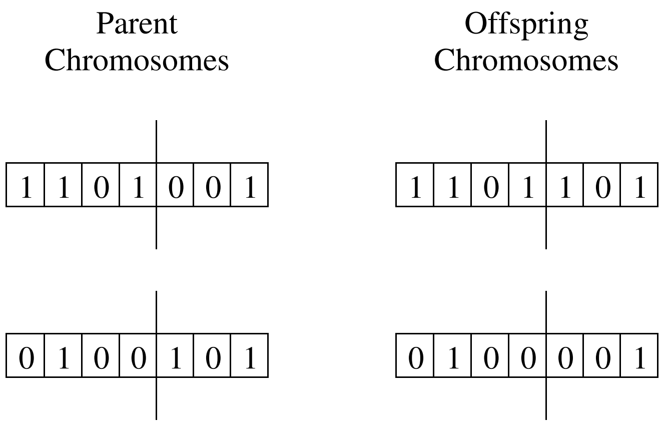


Figure 3.3 One-Point Crossover

In two-point crossover, the procedure is similar. Instead of marking a single point, two points are marked on parent chromosomes. The part between the marked points are taken from a parent, and parts beyond the marked points, which are not between the marked points, are taken prom the other parent. Representation of two-point crossover is shown in Figure 3.4.

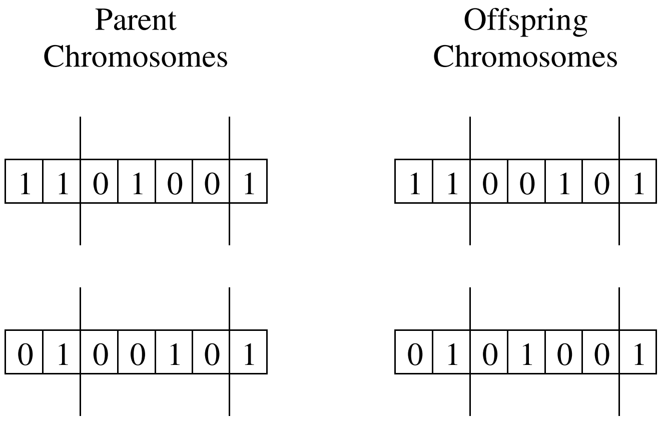


Figure 3.4. Two-Point Crossover

In the last most widely used crossover method, uniform crossover, genes of parent chromosomes are taken randomly and used to form the offspring chromosomes. In general, a rate of 50% is used in uniform crossover. With 50% rate, half of the chromosomes are taken from the first parent and other half is taken from the second parent chromosome. The genes are chosen randomly while satisfying the ratio.

3.6.2. Mutation

Mutation process is the second operator that creates new chromosomes. In mutation, spontaneous random changes are aimed so that genes that are not present in the initial population or genes that are hard to obtain from crossover operator can be created. In other words, role of mutation is to reach genes that are not present in population with random alterations.

Alternatively, in mutation, a mutation rate (mutation probability) is implemented. Just like in crossover, this probability decides what percent of the population will be mutated. However, the probability of mutation should not be as high as crossover so that the population can cover the resemblance to the parent chromosomes. A high mutation rate will also lead to high randomness which will end up with too long computational time to reach the optimum solution.

There are several different mutation operators. Inversion mutation, insertion mutation and displacement mutation are the most commonly used methods.

In inversion mutation, two genes in a chromosome is selected randomly and genes are swapped with each other.

In insertion mutation, a single gene is selected and it is inserted into a random point in the chromosome.

Lastly, in displacement mutation, instead of a single gene, a string of genes are selected. This string is inserted into a random point in the mutation. As it can be understood, insertion mutation is the convenient version of displacement mutation.

3.6.3. Selection

The last part before working with the new generation of solutions, selection procedure is applied. The selection is done so that GA can be directed into promising regions in the search domain. Commonly used selection procedures include roulette wheel selection, $(\mu+\lambda)$-selection, tournament selection and ranking and scaling selection.

Roulette wheel selection is proposed by Holland. The main idea is the probability of selecting each chromosome proportional to their fitness value. In other words, if the fitness of a single chromosome is high, selection of that chromosome is also high. If f_i is the fitness of individual "i" in the population, the probability of that individual to be selected is found by the following equation; $P_i = f_i / (\sum_{j=1}^{N} f_j)$, where N is the total number of individuals in the population.

In $(\mu+\lambda)$-selection method, the best individuals both from the parents and their offspring are selected.

In tournament selection, just as its name implies, a tournament between the individuals are done. The individuals competing in the tournament is selected randomly and the solution with the best fitness is selected as the winner. Tournament size can be changed and if the size is set as a large value, weaker chromosomes have less probability of being chosen.

In ranking and scaling method, procedure is a bit different. First, chromosomes are ranked with respect to their fitness values, then each chromosome is given a scale number with respect to their ranking. With best solution having the best scale, each chromosome has a probability of being chosen with respect to their scale. The main advantage of ranking and scaling method is that rapid takeover of chromosomes having much better fitness than others is eliminated.

With the completion of selection, a new population is created with individuals from initial population, offspring from crossover and mutated individuals from mutation. The process of this iteration continues until the termination condition of the algorithm is reached.

3.7. Advantages of GA

There are several reasons why GA is the most widely used optimization technique. One of the reasons is that the ability of adaptation. Due to nature of the algorithm, no complex mathematical and coding information is required. GA can handle any objective function and will search for solutions. GA is more efficient than most conventional heuristics which perform local searches and are usually designed for a case problem. On the other hand, GAs are designed as generic algorithms which can be implemented to any problem. These reasons contributes to wide usage of GAs.

CHAPTER 4

HYBRID GENETIC ALGORITHM

Throughout the literature, many different heuristic and meta-heuristic methods are suggested in order to solve resource levelling problems (RLP) in Resource Constraint Project Scheduling Problem (RCPSP), however, many of the methods are not appropriate to solve real life construction problems, which have more than 300 activities. Many of the RCPSP solvers available in the literature are practical for short project schedules which are less than 150 activities. Majority of the methods require long computational time when they are applied to large project schedules, moreover, the found solutions are not considered to be high quality solutions. In real life situations, completing a construction project on time or as short as possible is one of the main objectives of the contractors, since by completing the project in shorter durations, overhead costs may be decreased significantly.

For this purpose, in the context of this thesis, a hybrid genetic algorithm (HGA) is developed for multi-mode RCPSP. The proposed algorithm is composed of a heuristic method and genetic algorithm. In ordinary GA methods, search space is enormous and this limits the capability of the computation, and results in longer computational time. HGA starts with better solutions with the help of embedded Heuristic method and helps GA to limit the search space where better solutions are available. The main objective of the HGA is to reach high quality solutions with short computational time for large scale construction project of RCPSPs. This chapter further explains the details of HGA.

4.1. Project Characteristics

In the literature, there is a lack of algorithms that can solve large projects with multimode resource availabilities. Constraint Programming (CP) is suggested as an advanced mathematical optimization technique, to solve and optimize large scheduling problems from Menesi and Hegazy (2014). In CP, projects having activity numbers of 10, 100, 500, 1000, 1500 and 2000 are analyzed on. The projects are created by adding the core project having 10 activities several times. The CP required 3 hours for reaching an optimized solution at a RLP with 2000 activities with 9.61% deviation. The experiments with CP has been carried out on a laptop with 2.4 GHz CPU and 4 GB RAM. Hegazy (2014), has compared the CP method with ordinary GA, PSO and ACO methods and reached the solution that CP is faster and better in terms of finding high quality solutions. Despite this, in real life construction problems, shorter project duration is always favorable from the contractors' point of view since the overhead costs will be much lower. Due to this reason, HGA is designed to solve multimode RCPSP in less time with high quality results.

Although HGA is a generic algorithm, can work with any project instance, case study is taken same as Menesi and Hegazy (2014), who used the 10 activity project of Zhang (2012), so that the performance can be compared. In HGA, only a single resource type is implemented in the project. The project network can be seen in Figure 4.1.

The case project consists of 10 activities. Each activity can be completed with different modes each having different durations and different resource requirements. The multimode representation mirrors real life resource requirements successfully. An activity can be completed with different allocations of resources with different time. As the method of implementation changes, the duration and resource requirements change. Activities and resource requirements are summarized in Table 4.1.

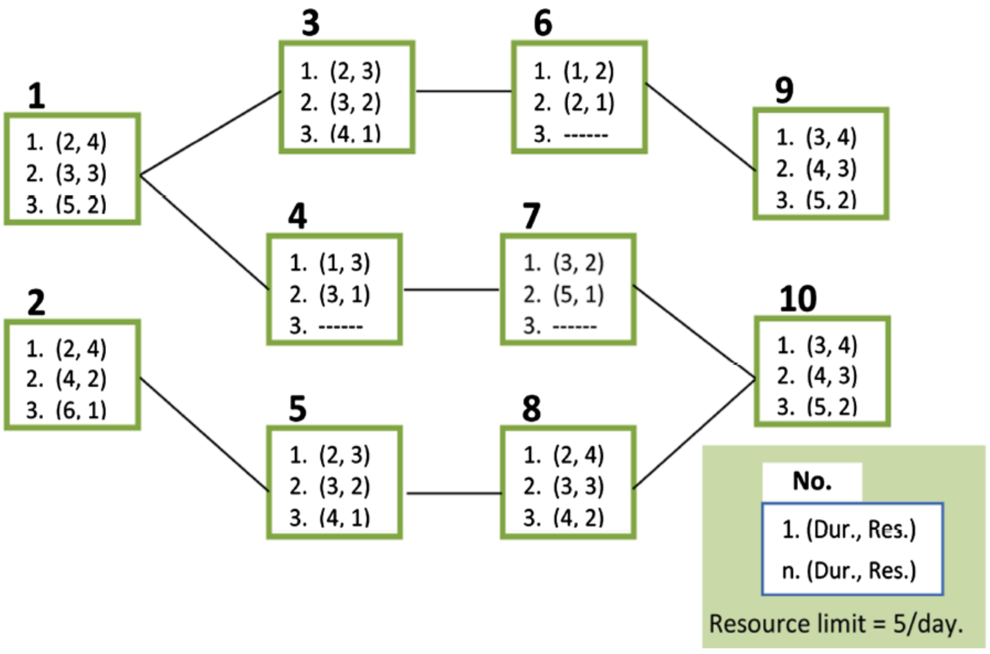


Figure 4.1. Project Network for the Case Project

Table 4.1. Resource Requirements with respect to Activities and Modes

	Mode 1		Mode 2		Mode 3	
Activity Number	Duration	Resource Requirement	Duration	Resource Requirement	Duration	Resource Requirement
1	2	4	3	3	5	2
2	2	4	4	2	6	1
3	2	3	3	2	4	1
4	1	3	3	1	-	-
5	2	3	3	2	4	1
6	1	2	2	1	-	-
7	3	2	5	1	-	-
8	2	4	3	3	4	2
9	3	4	4	3	5	2
10	3	4	4	3	5	2

To have convenient calculations and combining of 10 activity project severally to obtain bigger projects, two dummy activities are created, one as the starting point and the other as the end point of the network.

4.2. Chromosome Representation

In Genetic Algorithm methods, an initial population of individuals, each corresponding to a solution, is created. The individuals, probable solutions for the problem, are represented by chromosomes. In HGA, chromosomes are designed as a single main string which represent two different characteristics of the network. The first part of the chromosome, which is from the first gene to total number of activities, represents the activity priority. Between two genes, the one having the higher number is scheduled first. The second part, which starts from the gene after the total number of activities to last gene, represents the corresponding modes of each activity and the chosen mode. For instance, if an activity has three modes, the gene with the highest number is the chosen mode. In a chromosome, the priority of the activities and their corresponding modes are stored. Total number of genes in a chromosome is found by adding the total number of activities with corresponding mode number of each activity. For instance, the case project of 10 activities is represented by a chromosome of 41 genes. A dummy activity is assigned at the start and at the end of the network, this way creating bigger projects from the 10 activity network will be convenient. First 12 genes in the chromosome represents the priority of activities, including the dummy activities, after the first 12 genes, multimode alternatives of each activity is represented. Chromosome representation is shown in Figure 4.2.

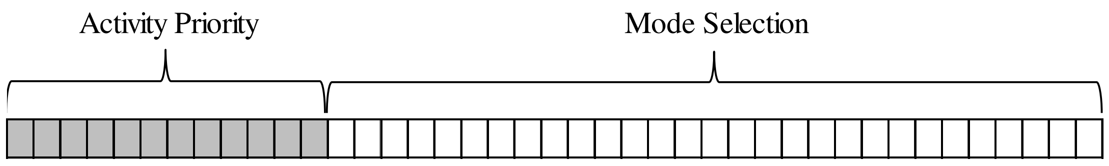


Figure 4.2. Chromosome Representation of HGA

4.3. Heuristic Method

The main difference of the proposed HGA from other methods is that it improves the initial population with the help of a newly developed Heuristic Algorithm. HGA begins with creating the initial population of 100 chromosomes. 30% of the chromosomes are created by assigning random real numbers between 0 and 1 to all genes in a chromosome. Remaining 70% of the chromosomes are created with the help of Heuristic Method. This process is implemented so that the initial population will have good starting point with good solutions which helps HGA to search promising areas in search domain. The reason for not creating all of the chromosomes with the heuristic method is so that randomness will not be lost. 30% of the population ensures that randomness is preserved throughout the algorithm.

The heuristic algorithm implements backward scheduling method to decide which activities will be scheduled and what will be their corresponding modes. In the algorithm, firstly, the activities which have zero number of successors are taken into consideration. If there are more than 1 activity that satisfies the successor number condition, one of the activities is chosen randomly. After choosing an activity, the start time of the schedule of the activities, possible finish times with respect to backward scheduling, are decided. After this process, mode selection process starts. Corresponding modes of the activities are decided randomly. Resource requirement of the selected mode is checked with the available resource at the possible finish time of the activity. If that mode does not satisfy the resource constraint, the activity is scheduled to earliest date where resources are available. This process is the general procedure that Heuristic Method follows.

In the calculation of possible finish time of the activities with respect to backward scheduling, the algorithm checks whether more than one activity can be scheduled starting from the same date. In other words, if their possible finish time dates are the same with respect to their successors, the algorithm gives priority to those activities and chooses the corresponding modes so that they can be scheduled together at the same schedule date while trying to use the resource availability to limits. When it comes to activities that can start at the same time, mode selection procedure differs. Between two activities, one of them is selected randomly and assigned a random corresponding mode, then, starting from the mode that has highest resource requirement, shortest duration, other activity is checked if they could be scheduled together. The reason for this selection is to ensure that resource requirement can be filled efficiently and possible makespan of the schedule can be arranged as early as possible. After an activity is scheduled, it is marked so that it will not be checked and will not be tried to be scheduled again. If an activity is the only one that can be scheduled at that point, the mode that can fill up the resource constraint is selected. At the end of the heuristic method, after all the activities are scheduled and the modes are decided, they are transformed into numbers and transferred into genes of a chromosome. Instead of giving random real numbers between 0 and 1 to genes, genes that represent mode selection are given binary numbers of 0 and 1, 0 meaning that mode is not selected and 1 meaning that particular mode is selected. A part of the chromosome is exampled in Figure 4.3. If that specific mode is selected that gene will be given the number of 1 and the others are given 0. The genes that represents priority of the activities are given very high real numbers, so that throughout the code, activities scheduled with heuristic method will always have priority over randomly generated ones. The activity that is scheduled last with respect to backward scheduling, will be given the highest priority and the priority number will be decreased by one until all activities are assigned with priority numbers. The reason for this arrangement is so that, the solution obtained from Heuristic Method can furtherly be improved with the forward scheduling implemented in HGA part.

Modes of Activity 3			Modes of Activity 4		Modes of Activity 5		
0	0	1	1	0	1	0	0

Figure 4.3. Chromosome Representation of modes in Heuristic Method

To further explain the heuristic method, an example will be illustrated with the project with 10 activities. At first, activity 11, which has zero resource requirement, no duration and is a dummy activity, is scheduled. Since the method applies backward schedule, activities that have zero number of successors are activity 10 and activity 9. The algorithm randomly selects activity 10 and selects second mode randomly for it. The method checks if activity 10 and 9 can be scheduled from the same date with the chosen mode of activity 10. Due to their predecessor, both activity 11 in this case, their probable finish time dates are same. The heuristic method checks whether they can be scheduled backwardly together starting at the same date with any possible mode selection combination. The corresponding mode of activity 9 is the third mode since it fills uses all of the remaining resources that are not used by activity 10. The selection of mode is done so that total resource availability can be used fully. Selected modes are labeled together with activity number and scheduled as seen in Figure 4.4.

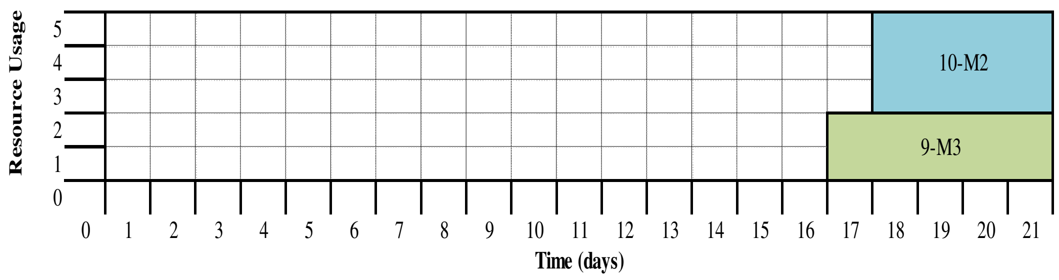


Figure 4.4. Activity 9 and 10 with Selected Modes

After activity 9 and 10 is scheduled, their predecessors, which are activity 6, 7 and 8, are arranged and taken into consideration. The algorithm randomly selects activity 6 and checks whether it can be scheduled from the same date with others. Since predecessor of activities 7 and 8, which is activity 10, has a start time different than the predecessor of activity 6, which is activity 9, no activity can be scheduled at the same time with activity 6. So, the algorithm randomly selects the first mode of activity 6 and schedules the activity as in Figure 4.5. Since activity 6 is now scheduled, activity 3 can now be worked with. At this point, method has 3 alternatives; activity 3, 7 and 8. Activity 7 is selected randomly and first mode of it is selected randomly. It is observed that activity 7 and 8 can be scheduled at the same date with respect to their predecessors, however, there is no mode for activity 8 that can use the remaining resource availability when activity 7 is implemented with the first mode. So, activity 7 is scheduled alone and activity 8 is scheduled to the next possible date in Figure 4.6.

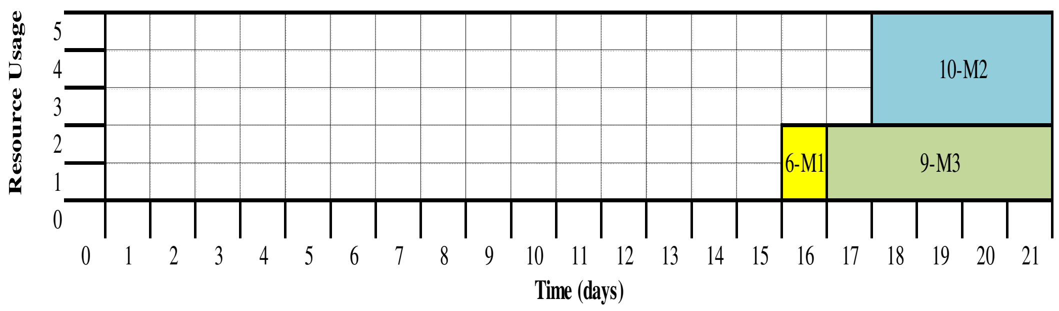


Figure 4.5. Scheduling of Activity 6

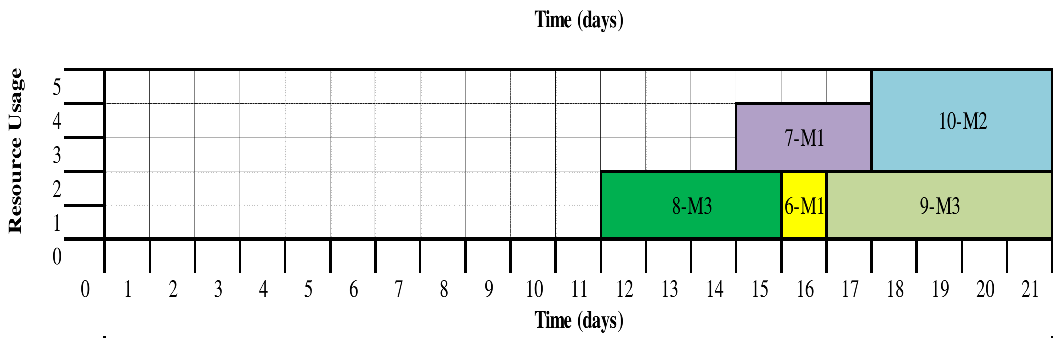


Figure 4.6. Scheduling of Activity 7 and 8

The next available activities for scheduling are; 3, 4 and 5. With respect to their predecessors, none of the activities can be scheduled together. Activity 4 is chosen randomly and scheduled with a randomly chosen mode. From the remaining activities, 3 and 5, activity 3 is selected and scheduled with a corresponding random mode. When activity 3 is scheduled, number of available activities increase. Since activity 1 has no successors except from activity 3 and 4, it can be taken into consideration as one of the available activities. Between activity 1 and 5, 5 is selected randomly and scheduled with a random mode in Figure 4.7. After activity 5 is scheduled, its predecessor, activity 2, can now be branded as an available activity. Activity 1 is chosen randomly together with a random mode and checked if it can be scheduled together with activity 2. Because of the reason that both of their successors has same earliest start times, they can be scheduled together. A mode for activity 2 that fills up the resource constraint is selected with respect to available resources. In the end dummy activity zero is scheduled and the backward scheduling finishes.

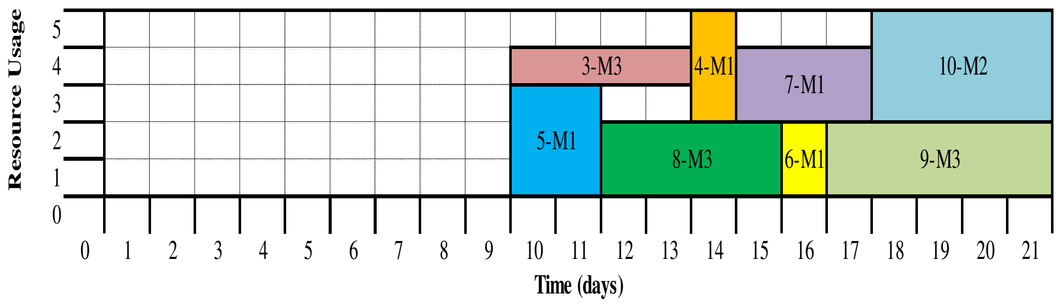


Figure 4.7. Scheduling of Activity 4, 3 and 5 in Order

At the end of the heuristic mode, genes that represent activity priorities are assigned real numbers opposite of their backward schedule order, since the chromosome will be taken into forward scheduling for makespan calculation. The order of the schedule with respect to backward scheduling starts with activity 11 (dummy activity) and continues with 10, 9, 6, 7, 8, 4, 3, 5, 1, 2 and ends with another dummy activity which is activity 0. Since there are 12 activities in total, including the dummy activities, the Heuristic method gives priority numbers opposite to their backward scheduling order. The lastly scheduled activity with respect to backward schedule will be given the highest priority of 11 and the firstly scheduled one will be given 0. This process is implemented so that the schedule obtained from Heuristic Method can be furtherly improved with forward scheduling in HGA. Selected mode of the activities are given binary number of 1 and unselected modes are given 0. Both mode selections and priority numbers can be observed in Figure 4.8.

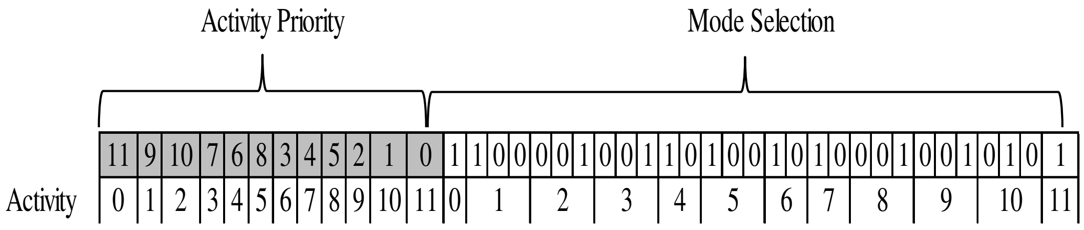


Figure 4.8. Representation of Mode Assignments of Example Problem

The scheduling process of heuristic method continues until all the activities are scheduled. The final form of the schedule is shown in Figure 4.9. The make-span of the backward schedule is found as 18 days. The backward schedule starts from a time in the future so that all activities can be scheduled backwardly without problem. Both mode selections and priorities are transferred into numbers and encoded into a chromosome so that HGA can work with the solutions.

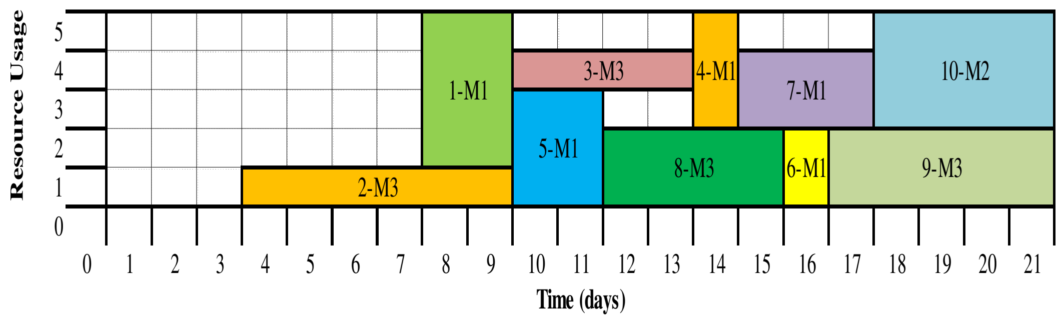


Figure 4.9. Final Form of Schedule with 18 Days Makespan

4.4. Hybrid Genetic Algorithm

When 70% of initial population is created with heuristic method, that 70% and 30%, which are created by giving chromosomes to random numbers, are inserted into forward scheduling. In forward scheduling part, the gene numbers are read for each corresponding chromosome. From chromosomes, by checking assigned numbers, the method decides the priority of the activities and the selected modes for each activity. Schedules obtained from heuristic method are also implemented into forward schedule method, the reason behind this is that some activities that can start at the same time may have different start dates when that schedule is obtained with backward schedule. For instance, in the example of Heuristic Method, activities 1 and 2 can be scheduled together if the resource constraint is satisfied. In Figure 4.9., Activities 1 and 2 satisfy the resource constraint to be scheduled together. By implementing the solution obtained with Heuristic Method into forward scheduling method, activity 1 and 2 can be started together, this way total make span of the schedule can be shortened.

In forward scheduling, the procedure is similar with backward scheduling. Instead of checking successor number, in forward scheduling predecessor number of an activity should be 0 so that it can be scheduled. Opposite from the Heuristic Method, in forwards scheduling part decision of mode selection and priorities are not done. All chromosomes have the required information encoded. The forward method does the scheduling with respect to encoded data.

4.5. Crossover, Mutation and Renewal

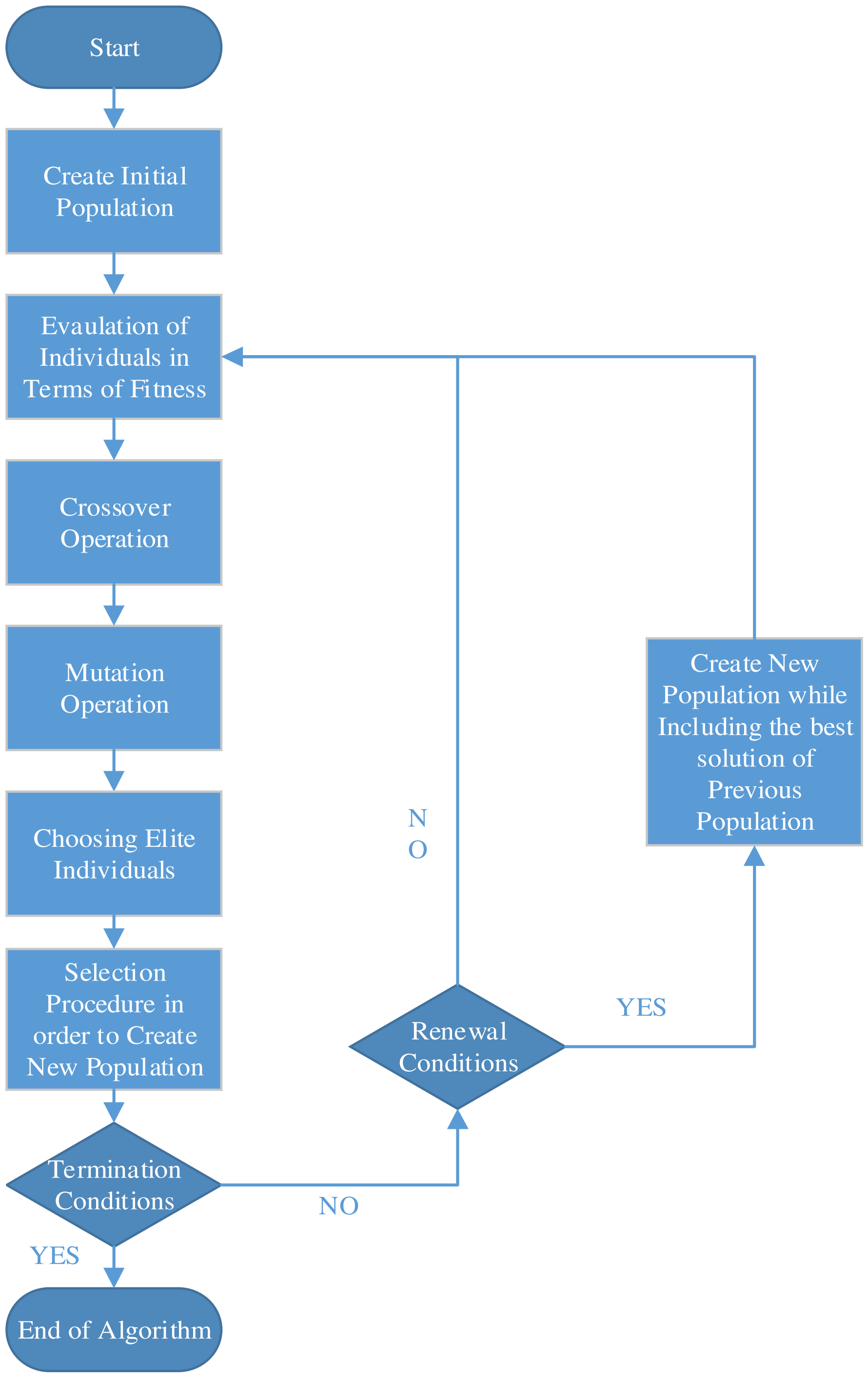
Same as ordinary GAs, HGA uses Crossover and Mutation procedures to improve the population. After initial population is created with Heuristic Method and randomly generated gene numbers, crossover and mutation procedures are implemented so that new individuals are introduced into the population. HGA uses a modified two point crossover. Initially, two chromosomes from the population is selected randomly. In this specific crossover method, at first stage of deciding the genes that will be crossed, two points are selected randomly between zero and total number of activities. By this way, activities between the randomly selected points are labeled. In second phase of the crossover operation, the corresponding modes of the randomly selected activities are selected. When both mode and activity selection is completed, the crossover operator crosses the selected genes between two chromosomes and a new individual is born. Child individuals created with crossover process are saved.

After crossover operation, mutation starts. In mutation a chromosome is selected randomly. At the start of mutation process, a random number is generated between 0 and 10 including both. If the generated number is lower than 2, the mutation is implemented into activity priority part of the chromosome, likewise, if the generated number is greater than 2 or equal to 2, mutation is applied to mode selection part. In the selected part of the chromosome, a random gene point is selected. If the selected part represents activity priority, the activity priority is given a random new number. However, if the selected gene point represents mode selection part, the procedure differs. The selected gene number is assigned a new number so that the current mode selection changes. This way, a controlled randomness is ensured and mutations that will not change the original schedule is avoided.

After crossover and mutation is finished elitist method is implemented. Some of the best individuals are marked so that they will not be terminated when new individuals created with crossover and mutation are inserted into the population. At each generation, 3% of the population is chosen as elite individuals.

Some of the individuals in the population are terminated with a probability with respect to their fitness. Less fitted chromosomes have higher probability of being terminated. New individuals are inserted instead of terminated chromosomes. And the whole procedure starts again from the beginning by implementing crossover, mutation, elitist method and termination.

In this HGA method, each makespan calculation, which is done when the initial population is created and new offspring in each generation created, is considered a schedule counter. This counter decides the stopping criteria of the algorithm. At some point of the algorithm, the best solution is saved, all other individuals are terminated, and the algorithm creates a new population by assigning the previously saved best solution as the first chromosome. This process is called as the renewal process. In other words, at some intervals (5000 schedule) of the algorithm, the best solution is inserted into a new population. This procedure is implemented so that early convergence, which is the main disadvantage of ordinary GAs, can be prevented. The flow chart of the HGA is shown in Figure 4.10.



Start
Create Initial Population
Evaulation of Individuals in Terms of Fitness
Crossover Operation
Mutation Operation
Choosing Elite Individuals
Selection Procedure in order to Create New Population
Termination Conditions
YES
End of Algorithm
NO
Renewal Conditions
N O
YES
Create New Population while Including the best solution of Previous Population

Figure 4.10. Flowchart of HGA

In Figure 4.11, the example problem is inserted into forward scheduling method and the make span is shortened by 2 days resulting the make span with 16 days.

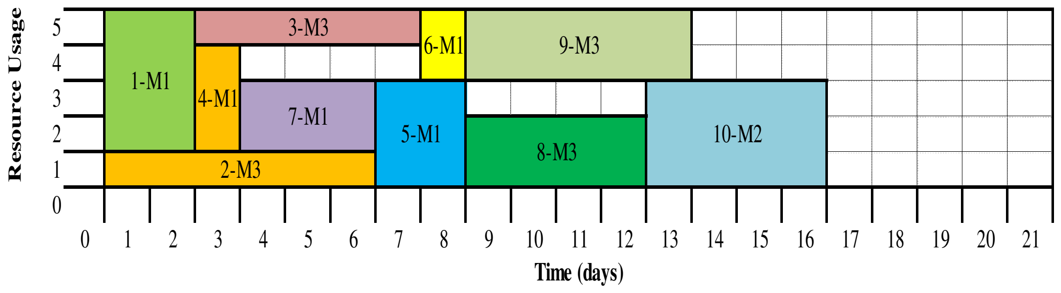


Figure 4.11. Improved Schedule with 16 Days Makespan

In crossover operation, if activities 3 and 5 are selected, at first, priorities of the activities between 3 and 5 are marked, including 3 and 5. Later, Multimode alternatives of the same activities are marked and between the selected parents, both priority and mode alternatives of the selected activities are crossed. The crossed parts of the parent chromosome can be observed in Figure 4.12.

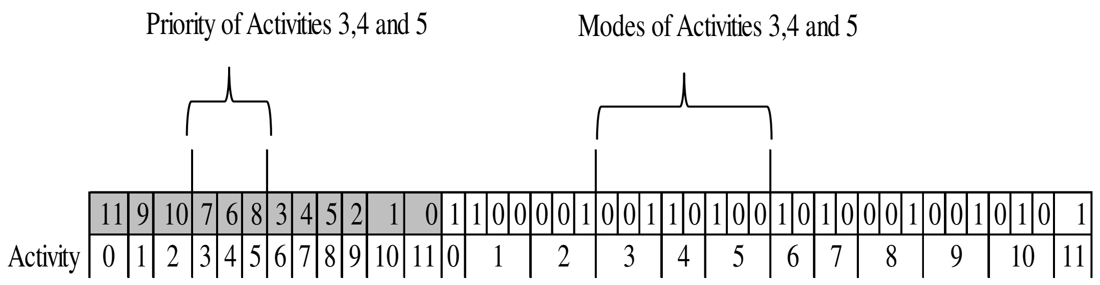


Figure 4.12. Priority and Mode Alternatives in Crossover

4.6. Fine-Tuning

With fine-tuning analysis, appropriate set of parameter values for HGA are determined and summarized in Table 4.2. Fine-tuning analysis is done with 6 different parameters, namely; crossover rate, mutation rate, initial population size, renewal of the population in terms of schedule number and percentage of the population that is implemented into heuristic algorithm. The project with 100 activities is selected for analysis since it takes less time when compared with bigger projects. Each combination is analyzed, 10 results are obtained for each combination and the one with the least average standard deviation is selected. In the result of fine-tuning process, the combination of 40% crossover rate, 4% mutation rate, initial population size of 100, heuristic percentage of 70% and renewal of 5000 schedules is selected with 1.86% standard deviation since it gives the best performance out of all experiments. In total, 48 experiments are done in fine-tuning phase. The detailed information, including the best and worst results and average results, about fine-tuning cases can be found in Appendix A.

Table 4.2. Fine-tuning Parameter Selection

Parameters	Range of Parameters			Selected Value
	Low	Medium	High	
Crossover	0,4	0,6	0,8	0,4
Mutation	0,02	0,04	-	0,04
Initial Population Size	50	100	-	100
Heuristic Percentage	0,5	0,7	-	0,7
Renewal	5000	10000	-	5000

CHAPTER 5

PERFORMANCE ANALYSIS OF HGA

In this chapter, the performance of HGA is compared with the state-of-art methods. Instead of choosing computational time as the stopping criteria, HGA uses number of schedules as stopping criteria. The performance analysis is done with a similar laptop with 2.4 GHz CPU and 8GB RAM just as Hegazy (2014) so that performance and the results would not be affected by the properties of the computer. The algorithm is designed with C# language in Microsoft Visual Studio 2013 Ultimate Edition. Average percentage deviations available in the Table 5.1, are the values from the best solutions found. Optimal solution is the best current solution found by HGA. The APD is calculated with the following equation;

APD (%) = (Solution Duration-Best Known Duration) x 100 / Best Known Duration

Computational results are summarized and presented in Table 5.1 and the relationship between the solution quality and processing time can be observed in Figure 5.1. The complete results, obtained by 40 consecutive experiments, found for each different project schedule with changing schedule numbers can be found in the Appendix B. For each project, different number of schedules are used as stopping criteria, namely; 10000 schedule, 50000 schedule, 100000 schedule, 250000 schedule, 500000 schedule and 1000000 schedule for projects with activities 100 and 500, 1000, 1500 and 2000.

5.1. Small Size Projects

The proposed HGA is compared both with algorithms that concentrated on small size and large size projects. For the schedule with 10 activities both CP and HGA found the optimal solution instantly. The comparison between the algorithms with respect to

Table 5.1. Experiment Results of HGA

Case	Upper Bound	# of Schedules	HGA Solution	Average Percentage Deviation (APD)
10 Activities	14	10000	14	0%
100 Activities	140	10000	145.90	4.21%
		50000	143.15	2.25%
		100000	141.63	1.16%
		250000	140.45	0.32%
		500000	140.20	0.14%
		1000000	140.00	0.00%
500 Activities	700	10000	761.15	8.74%
		50000	739.03	5.58%
		100000	729.05	4.15%
		250000	719.80	2.83%
		500000	713.33	1.90%
		1000000	706.85	0.98%
1000 Activities	1400	10000	1549.83	10.70%
		50000	1503.78	7.41%
		100000	1481.25	5.80%
		250000	1459.48	4.25%
		500000	1445.80	3.27%
		1000000	1432.88	2.35%
1500 Activities	2100	10000	2341.20	11.49%
		50000	2281.48	8.64%
		100000	2252.68	7.27%
		250000	2208.05	5.15%
		500000	2184.53	4.03%
		1000000	2166.73	3.18%

Table 5.1. Experiment Results of HGA (Continued)

Case	Upper Bound	# of Schedules	HGA Solution	Average Percentage Deviation (APD)
2000 Activities	2800	10000	3134.10	11.93%
		50000	3064.23	9.44%
		100000	3020.88	7.89%
		250000	2963.45	5.84%
		500000	2929.95	4.64%
		1000000	2909.50	3.91%

computational time, number of schedules and solution quality can be observed in Table 5.2. It can be observed that HGA outperforms all of the proposed algorithms.

Table 5.2. Comparison between HGA and Other Methods on Small Size Projects

Year of Publication	Authors	Method	# of Activities	APD	# of schedules	Computational Time
2012	Zhang	Ant Colony Optimization (ACO)	10	0.10%	-	18.3
2014	Menesi and Hegazy	Constraint Programming (CP)	10	0.00%	-	<1 sec
2015	This work	HGA	10	0.00%	1000	0.12 sec

5.2. Large Size Projects

When the project gets bigger the results start to differ. For the project with 100 activities, HGA found APD of 2.25%, 1.16%, 0.32%, 0.14% and 0% for 10000, 50000, 100000, 250000, 500000 and 1000000 number of schedules respectively. For the project with 500 activities APD results for the chosen stopping criteria are 8.74%, 5.58%, 4.15%, 2.83%, 1.90% and 0.98%. For the 1000 activity project, APD results are 10.70%, 7.41%, 5.80%, 4.25%, 3.27% and 2.35%. For the 1500 activity project APD results are 11.49%, 8.64%, 7.27%, 5.15%, 4.03% and 3.18%. Lastly for the largest project of 2000 activities, APD results are 11.93%, 9.44%, 7.89%, 5.84%, 4.64% and 3.91%.

As it can be understood from the data, HGA has decreasing APD as the schedule number increases. Especially, in large sized projects, HGA managed to obtain high quality results, much better than CP proposed by Hegazy and Menesi (2014). The same results with CP has been reached by HGA in much less computational time. When HGA is run with the same time as CP, the obtained results are much better than CP. Although, computational time and APD values are close with each other in small sized projects, mainly with 100 activities, HGA has great ability to solve large sized project instances and exceeded the CP both in terms of computational time and quality of the results. The best and worst results of each experiment can be found in Table 5.3.

The comparison between the experiment results in terms of average solution, average percent deviation and processing time of CP and HGA for each project instance can be found in Table 5.4.

Table 5.3. Best and Worst Results of Experiments

Case	Upper Bound	# of Schedules	HGA Average Solution	HGA Best Solution	HGA Worst Solution
10 Activities	14	10000	14.00	14.00	14.00
100 Activities	140	10000	145.90	142.00	148.00
		50000	143.15	140.00	146.00
		100000	141.63	140.00	144.00
		250000	140.45	140.00	142.00
		500000	140.20	140.00	141.00
		1000000	140.00	140.00	140.00
500 Activities	700	10000	761.15	755.00	767.00
		50000	739.03	729.00	744.00
		100000	729.05	721.00	733.00
		250000	719.80	714.00	728.00
		500000	713.33	706.00	718.00
		1000000	706.85	701.00	712.00
1000 Activities	1400	10000	1549.83	1536.00	1572.00
		50000	1503.78	1489.00	1513.00
		100000	1481.25	1470.00	1490.00
		250000	1459.48	1448.00	1469.00
		500000	1445.80	1437.00	1455.00
		1000000	1432.88	1426.00	1441.00
1500 Activities	2100	10000	2341.20	2321.00	2372.00
		50000	2281.48	2260.00	2304.00
		100000	2252.68	2236.00	2267.00
		250000	2208.05	2196.00	2216.00
		500000	2184.53	2179.00	2190.00
		1000000	2166.73	2159.00	2176.00
2000 Activities	2800	10000	3134.10	3113.00	3159.00
		50000	3064.23	3042.00	3080.00
		100000	3020.88	2998.00	3036.00
		250000	2963.45	2946.00	2975.00
		500000	2929.95	2919.00	2943.00
		1000000	2909.50	2895.00	2926.00

Table 5.4. Comparison between HGA and CP

Case	Upper Bound	# of Schedules	HGA Average Solution	APD of HGA	HGA Processing time	CP Average Solution	APD of CP	CP Processing time
10 Activities	14	50000	14	0%	Instant	14	0.00%	
100 Activities	140	10000	145.9	4%	01:35 s	161	15.00%	1 s
		50000	143.15	2.25%	6 s	143	2.14%	1 min
		100000	141.63	1.16%	12 s	140	0.00%	3 min
		250000	140.45	0.32%	30 s			
		500000	140.20	0.14%	1 min			
		1000000	140.00	0.00%	02:05 min			
500 Activities	700	10000	761.15	8.74%	08:75 s	777.00	11.00%	1 min
		50000	739.03	5.58%	45 s	732.00	4.57%	10 min
		100000	729.05	4.15%	01:45 min	725.00	3.57%	20 min
		250000	719.80	2.83%	03:30 min	718.00	2.57%	30 min
		500000	713.33	1.90%	07:10 min	712.00	1.71%	1 h
		1000000	706.85	0.98%	14:05 min	708.00	1.14%	2 h
1000 Activities	1400	10000	1549.83	10.70%	26 s	1664.00	18.86%	1 min
		50000	1503.78	7.41%	2 min	1550.00	10.71%	10 min
		100000	1481.25	5.80%	04:20 min	1512.00	8.00%	20 min
		250000	1459.48	4.25%	10:40 min	1500.00	7.14%	30 min
		500000	1445.80	3.27%	21:30 min	1488.00	6.29%	1 h
		1000000	1432.88	2.35%	43:00 min	1467.00	4.79%	2 h

Table 5.4. Comparison between HGA and CP (Continued)

Case	Upper Bound	# of Schedules	HGA Average Solution	APD of HGA	HGA Processing time	CP Average Solution	APD of CP	CP Processing time
1500 Activities	2100	10000	2341.20	11.49%	54 s	2541.00	21.00%	1 min
		50000	2281.48	8.64%	4:30 min	2330.00	10.95%	10 min
		100000	2252.68	7.27%	09:20 min	2322.00	10.57%	20 min
		250000	2208.05	5.15%	22:50 min	2316.00	10.29%	30 min
		500000	2184.53	4.03%	45:30 min	2296.00	9.33%	1 h
		1000000	2166.73	3.18%	1:30 h	2254.00	7.33%	2 h
2000 Activities	2800	10000	3134.10	11.93%	1:30 min	3406.00	21.64%	1 min
		50000	3064.23	9.44%	7:30 min	3118.00	11.36%	10 min
		100000	3020.88	7.89%	15:45 min	3107.00	10.96%	20 min
		250000	2963.45	5.84%	40 min	3102.00	10.79%	30 min
		500000	2929.95	4.64%	1:16 h	3098.00	10.64%	1 h
		1000000	2909.50	3.91%	2:35 h	3079.00	9.96%	2 h
						3069.00	9.61%	3 h

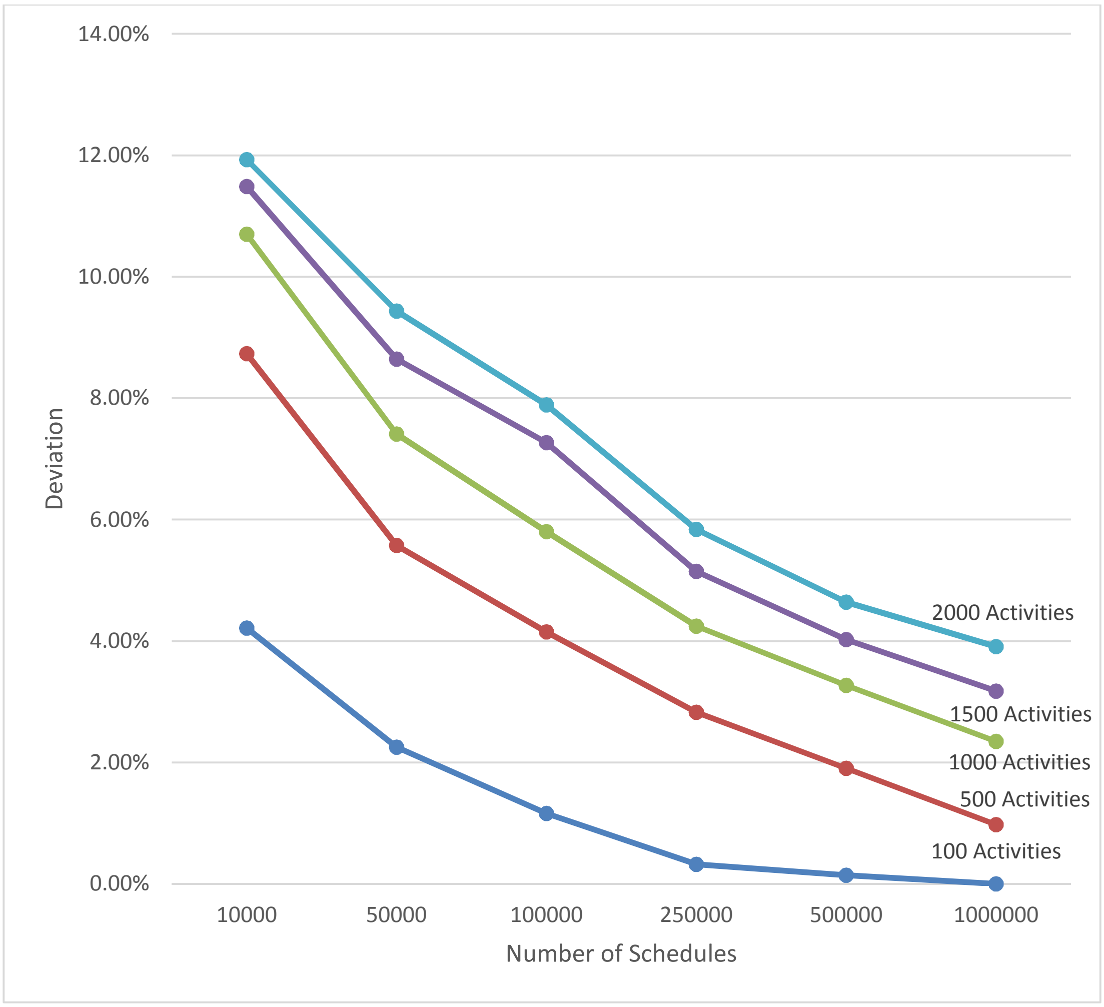


Figure 5.1. Solution Quality versus Number of Schedules for Different Project Sizes

All results of HGA are better in terms of APD when compared with CP. Another advantage HGA has over CP is that HGA reaches much better results with less computational time. Result comparison of CP and HGA can be observed in Table 5.4. The results of HGA have both better quality and less computational time. In real life situations in construction industry, even 1 day has big impact on the cost of the project. Overhead costs such as; security costs, electricity costs, accommodation costs of the workers and the cooking costs increase when the total project time increases. The average best solution found by HGA is 2909.50 days with APD of 3.91% for the project with 2000 activities. The best result found by CP is 3069 days with APD of 9.61% for the same problem. The difference in total project time is approximately 160 days which contributes greatly to the total overhead cost of the project. If 10,000 USD, which is an acceptable value for medium sized projects, is spent for overhead costs per day, 1,600,000 USD in total can be saved throughout the project.

CHAPTER 6

CONCLUSION

In this thesis, the significance of the RCPCP is discussed. To solve problems with resource constraints and multiple modes, a hybrid genetic algorithm with a newly developed Heuristic Method is proposed. In the proposed Heuristic Method, backward scheduling is implemented and successor number of the activities together with the resource requirements are taken into consideration. In HGA, forward scheduling is implemented to all solutions, this way, advantages of backward-forward scheduling is used. Computational experiments show that HGA is much more efficient than other proposed methods.

In literature, there is a lack of study on large problems with multimode alternatives. Due to this reason, a project of 10 activities having multimode resource alternatives is implemented. Projects with larger sizes of 100, 500, 1000, 1500 and 2000 activities are created by adding the project of 10 activities several times. For each project, 40 consecutive experiments are run and average deviation from optimal solution is found. The main contribution of this work is that HGA can reach high quality solutions in large projects in short computational time, especially for large size projects. Experiment results show that, Heuristic Method lets the algorithm concentrate on the promising search areas, this way, computational time required to reach high quality solutions decrease significant

Since many of the real life construction projects have more than 300 activities, main aim of this thesis was to reach high quality solutions in large project instances. In projects of 10 and 100 activities, HGA reached best known solutions. In larger projects, HGA reached solutions with too little deviation from best known solutions. When compared with CP, HGA has reached better results in the same amount of computational time.

Although large project instances reflect the size of real life construction projects, they do not reflect the complexity of them since they are obtained by adding the project of 10 activities several times. This leaves a possible future research on development of methods that can reach high quality solutions on complex large scale projects with resource constraints. Although the proposed method reaches high quality solutions, solutions with the same makespan may have different priority and mode arrangements. In future works, a method can be developed for solutions with same makepsan so that different mode and priority arrangements can be observed. It is also observed from the experimental results that, with enough time HGA can reach the optimal solution for all project instances. In HGA single resource is defined. Another future study may be focused on increased resource number.

In conclusion, there was a need for a method that can provide good solutions for large scale projects with multimode RCPSP. HGA is designed in order to find high quality solutions in large scale projects. Many of the construction projects usually delay due to inefficient use of resources and improper planning. In this work, the main aim is to overcome RCPSP in large scale projects with multimode alternatives. The implementation of multimode resource alternatives shows that an activity in a project can be finished earlier if more resources are transferred. The decision of whether to crash the activity or not should be given carefully so that project could be finished on time without violating the resource constraints. Reaching earlier finish times are important since with each early day, the overhead costs of the construction company will decrease which, in the end, contributes to the overall profit of the project.

REFERENCES

Ahn, T., and Erenguc, S.S. (1998). “The resource constrained project scheduling problem with multiple crashable modes: A heuristic procedure.” European Journal of Operational Research, 107, 250-259.

Alcaraz, J. Maroto, C. and Ruiz, R. (2003). Solving the multi-mode resourceconstrained project scheduling problem with genetic algorithms. Journal of the Operational Research Society, 54, 614–626.

Bettemir, O., and Sonmez, R. (2014). “Hybrid genetic algorithm with simulated annealing for resource-constrained project scheduling.” Journal of Management in Engineering, 10.1061/(ASCE)ME.1943-5479.0000323 , 04014082.

Blazewicz, J., Lenstra, J., and Rinnooy Kan, A.H.G. (1983). “ Scheduling subject to resource constraints: classification and complexity.” Discrete Applied Mathematics, 5, 11–24.

Chan, W. T., Chua, D. K. H., and Kannan, G. (1996). “Construction Resource Scheduling with Genetic Algorithms.” Journal of Construction Engineering and Management, 122(2), 125-132.

Chen, P. H., and Shahandashti, S.M. (2009). “Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints.” Automation inConstruction, 18, 434–443.

Chen, P. H. and Weng, H. (2009). “A two-phase GA model for resource-constrained project scheduling.“ Automation in Construction, 18, 485–498

Damak, N. Jarboui, B. Siarry, P. and Loukil, T. (2009). Differential evolution for solving multi-mode resource constrained project scheduling problems. Computers and Operations Research, 36, 2653-2659.

De, P., Dunne, E. J., Ghosh, J. B., and Wells, C. E. (1997). “Complexity of the Discrete Time-Cost Tradeoff Problem for Project Networks.” Operations Research, 45(2), 302–306.

Dorigo, M. (1992). “Optimization, learning and natural algorithms.” Ph.D. thesis, Politecnico di Milano, Italy.

Drexl, A., and Gruenewald, J. (1993). "Nonpreemptive multi-mode resource-constrained project scheduling", IIE Transactions 25, 74-81.

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

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

Elloumi, S. and Fortemps, P. (2010). A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. European Journal of Operational Research, 205, 31–41.

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

Gen, M., Cheng, R. and Lin, L. (2008) Network Models and Optimization, Multiobjective Genetic Algorithm Approach, MA: Springer-Verlag

Goldberg, D. (1989). Genetic Algorithms in search, Optimization, and Machine Learning, MA: Addison-Wesley., 1-44.

Hartmann, S. (1998). “A competitive genetic algorithm for resource constrained project scheduling.” Naval Research Logistics, 45(7), 733–75.

Hartmann, S. (2001). Project scheduling with multiple modes: A genetic algorithm. Annals of Operations Research, 102, 111–135.

Hegazy, T. (1999). “Optimization of resource allocation and leveling using genetic algorithms.” Journal of Construction Engineering and Management, 125(3), 167–175.

Hegazy, T., and Menesi, W. (2012). “Heuristic Method for Satisfying Both Deadlines and Resource Constraints.” Journal of Construction Engineering and Management, 138(6), 688-696.

Hegazy, T., Shabeeb, A., El-Beltagi, E., and Cheema, T. (2000). “Algorithm for Scheduling with Multi-Skilled Constrained Resources.” Journal of Construction Engineering and Management, 126(4), 414-421.

Hekimoglu, O. (2007). Comparison of the resource allocation capabilities of project management software packages in resource constrained project scheduling problems. M.S. thesis, Middle East Technical Univ., Ankara, Turkey.

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

Holland, J. (1992). Adaptation in Natural and Artificial System, Ann Arbor: University of Michigan Press; 1975, MA: MIT Press.

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

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

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

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

Kim, J. L., and Ellis R. D. (2010). "Comparing Schedule Generation Schemes in Resource-Constrained Project Scheduling Using Elitist Genetic Algorithm." Journal of Construction Engineering and Management, 136(2) 160–169.

Kim, J. L., and Ellis, R. D. (2008). "Permutation-based elitist genetic algorithm for optimization of large-sized resource-constrained project scheduling." Journal of Construction Engineering and Management, 134(11) 904–913.

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

Lova, A. Tormos, P. Cervantes, M. and Barber, F. (2009). An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. International Journal of Production Economics, 117 (2), 302–316.

Lu, M., Lam, H-C., and Dai, F. (2008). “Resource-constrained critical path analysis based on discrete event simulation and particle swarm optimization.” Automation in Construction, 17, 670–681.

Mellentien, C., and Trautmann, N. (2001). “Resource allocation with project management software.” OR Spektrum, 23, 383–394

Mendes, J. Gonçalves, J. and Resende, M. (2009). A random key based genetic algorithm for the resource constrained project scheduling problem. Computers and Operations Research, 36, 92 – 109.

Menesi, W. and Hegazy, T. (2014). "Multimode Resource-Constrained Scheduling and Leveling for Practical-Size Projects." J. Manage. Eng., 10.1061/(ASCE)ME.1943-5479.0000338 , 04014092.

Mobini, M., Mobini, Z., Rabbani, M. (2010). “An Artificial Immune Algorithm for the project scheduling problem under resource constraints.” Journal: Applied Soft Computing - ASC, vol. 11, no. 2, pp. 1975-1982, 2011 DOI: 10.1016/j.asoc.2010.06.013.

Mori, M. and Tseng, C. (1997). A genetic algorithm for multi-mode resource constrained project scheduling problem. European Journal of Operational Research, 100, 134–141.

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

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, (1953). Equation of state calculations by fast computing machines, Journal of Chemical Physics 21 (6) 1087–1092.

Peteghem, V.V. and Vanhoucke, M. (2009). An artificial immune system for the multi-mode resource-constrained project scheduling problem. In: EvoCOP, Springer, Berlin, Tubingen, Germany.

Peteghem, V.V. and Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. European Journal of Operational Research, 201, 409–418.

Rechenberg, I. (1973). Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Stuttgart: Fromman-Holzboog.

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

Schwefel, H. (1995). Evolution and Optimum Seeking, New York: Wiley & Sons.

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

Sonmez, R., and Uysal, F., (2014). "Backward-Forward Hybrid Genetic Algorithm for Resource-Constrained Multiproject Scheduling Problem." Journal of Computing in Civil Engineering, 10.1061/(ASCE)CP.1943-5487.0000382, 04014072.

Sprecher, A., Hartmann, S. and Drexl, A., (1997). An exact algorithm for project scheduling with multiple modes.OR Spektrum 19,195–203.

Talbot, F., and Patterson, J. H. (1979). ‘‘Optimal methods for scheduling projects under resource constraints.’’ Proj. Mgmt. Quarterly, 10(4), 26 – 33.

Tormos, P., and Lova, A. (2001). “A competitive heuristic solution technique for resource constrained project scheduling.” Annals of Operations Research,102(1/4), 65–81.

Zhang, H. (2012). “Ant colony optimization for multimode resource constrained project scheduling.” J. Manage. Eng., 10.1061/(ASCE) ME.1943-5479.0000089, 150–159.

APPENDIX A

FINE-TUNING EXPERIMENT

Table A.1. Fine-Tuning Experiment

Crossover	Mutation	Population Size	Heuristic Percentage	New Population	Best Solution	Worst Solution	Average Solution	APD
0.4	0.02	50	50	5000	143.00	147.00	145.00	3.57%
0.4	0.02	50	50	10000	144.00	150.00	146.60	4.71%
0.4	0.02	100	50	5000	142.00	145.00	143.90	2.79%
0.4	0.02	100	50	10000	142.00	146.00	144.50	3.21%
0.4	0.02	50	70	5000	144.00	147.00	145.20	3.71%
0.4	0.02	50	70	10000	143.00	149.00	145.50	3.93%
0.4	0.02	100	70	5000	140.00	145.00	142.70	1.93%
0.4	0.02	100	70	10000	142.00	145.00	143.90	2.79%
0.4	0.04	50	50	5000	142.00	148.00	145.10	3.64%
0.4	0.04	50	50	10000	144.00	148.00	146.10	4.36%
0.4	0.04	100	50	5000	140.00	145.00	143.30	2.36%
0.4	0.04	100	50	10000	140.00	146.00	143.80	2.71%
0.4	0.04	50	70	5000	143.00	147.00	145.20	3.71%
0.4	0.04	50	70	10000	142.00	148.00	145.30	3.79%
0.4	0.04	100	70	5000	141.00	144.00	142.60	1.86%
0.4	0.04	100	70	10000	142.00	146.00	143.90	2.79%
0.6	0.02	50	50	5000	145.00	148.00	145.90	4.21%
0.6	0.02	50	50	10000	144.00	149.00	146.80	4.86%
0.6	0.02	100	50	5000	142.00	146.00	144.10	2.93%
0.6	0.02	100	50	10000	141.00	146.00	144.20	3.00%
0.6	0.02	50	70	5000	142.00	145.00	143.90	2.79%
0.6	0.02	50	70	10000	144.00	148.00	146.50	4.64%

Table A.1. Fine-Tuning Experiment (Continued)

Crossover	Mutation	Population Size	Heuristic Percentage	New Population	Best Solution	Worst Solution	Average Solution	APD
0.6	0.02	100	70	5000	141.00	145.00	143.00	2.14%
0.6	0.02	100	70	10000	141.00	146.00	144.40	3.14%
0.6	0.04	50	50	5000	143.00	147.00	145.50	3.93%
0.6	0.04	50	50	10000	142.00	150.00	145.20	3.71%
0.6	0.04	100	50	5000	141.00	145.00	143.30	2.36%
0.6	0.04	100	50	10000	141.00	148.00	144.00	2.86%
0.6	0.04	50	70	5000	142.00	147.00	144.40	3.14%
0.6	0.04	50	70	10000	145.00	148.00	146.50	4.64%
0.6	0.04	100	70	5000	142.00	146.00	143.20	2.29%
0.6	0.04	100	70	10000	142.00	146.00	144.20	3.00%
0.8	0.02	50	50	5000	143.00	148.00	145.00	3.57%
0.8	0.02	50	50	10000	144.00	149.00	146.10	4.36%
0.8	0.02	100	50	5000	143.00	145.00	143.70	2.64%
0.8	0.02	100	50	10000	143.00	146.00	144.30	3.07%
0.8	0.02	50	70	5000	144.00	148.00	145.70	4.07%
0.8	0.02	50	70	10000	143.00	148.00	145.70	4.07%
0.8	0.02	100	70	5000	142.00	145.00	143.80	2.71%
0.8	0.02	100	70	10000	141.00	146.00	143.70	2.64%
0.8	0.04	50	50	5000	144.00	148.00	145.40	3.86%
0.8	0.04	50	50	10000	144.00	148.00	145.60	4.00%
0.8	0.04	100	50	5000	143.00	146.00	144.50	3.21%
0.8	0.04	100	50	10000	143.00	148.00	145.20	3.71%

Table A.1. Fine-Tuning Experiment (Continued)

Crossover	Mutation	Population Size	Heuristic Percentage	New Population	Best Solution	Worst Solution	Average Solution	APD
0.8	0.04	50	70	5000	142.00	148.00	144.20	3.00%
0.8	0.04	50	70	10000	144.00	147.00	145.60	4.00%
0.8	0.04	100	70	5000	141.00	146.00	143.90	2.79%
0.8	0.04	100	70	10000	141.00	147.00	143.50	2.50%

APPENDIX B

DATA RELATIVE TO CHAPTER 5

Table B.1. Experiments for All Schedule Numbers

# of Activities	100	# of Schedule	10000			# of Activities	100	# of Schedule	50000		
No	Result	APD	No	Result	APD	No	Result	APD	No	Result	APD
1	143	2.14%	21	148	5.71%	1	143	2.14%	21	142	1.43%
2	145	3.57%	22	145	3.57%	2	143	2.14%	22	144	2.86%
3	147	5.00%	23	147	5.00%	3	145	3.57%	23	143	2.14%
4	148	5.71%	24	147	5.00%	4	141	0.71%	24	144	2.86%
5	148	5.71%	25	144	2.86%	5	145	3.57%	25	142	1.43%
6	143	2.14%	26	145	3.57%	6	142	1.43%	26	144	2.86%
7	147	5.00%	27	146	4.29%	7	140	0.00%	27	145	3.57%
8	144	2.86%	28	146	4.29%	8	142	1.43%	28	143	2.14%
9	145	3.57%	29	146	4.29%	9	140	0.00%	29	146	4.29%
10	147	5.00%	30	147	5.00%	10	143	2.14%	30	144	2.86%
11	143	2.14%	31	147	5.00%	11	143	2.14%	31	144	2.86%
12	148	5.71%	32	145	3.57%	12	144	2.86%	32	143	2.14%
13	146	4.29%	33	148	5.71%	13	145	3.57%	33	141	0.71%
14	142	1.43%	34	147	5.00%	14	145	3.57%	34	143	2.14%
15	145	3.57%	35	145	3.57%	15	143	2.14%	35	144	2.86%
16	146	4.29%	36	145	3.57%	16	144	2.86%	36	143	2.14%
17	147	5.00%	37	146	4.29%	17	142	1.43%	37	143	2.14%
18	148	5.71%	38	147	5.00%	18	144	2.86%	38	142	1.43%
19	147	5.00%	39	146	4.29%	19	142	1.43%	39	143	2.14%
20	145	3.57%	40	145	3.57%	20	144	2.86%	40	143	2.14%
			Average	145.90	4.21%				Average	143.15	2.25%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	100	# of Schedule	100000			# of Activities	100	# of Schedule	250000		
No	Result	APD	No	Result	APD	No	Result	APD	No	Result	APD
1	141	0.71%	21	142	1.43%	1	140	0.00%	21	140	0.00%
2	140	0.00%	22	141	0.71%	2	141	0.71%	22	140	0.00%
3	141	0.71%	23	142	1.43%	3	140	0.00%	23	141	0.71%
4	142	1.43%	24	142	1.43%	4	141	0.71%	24	140	0.00%
5	141	0.71%	25	144	2.86%	5	140	0.00%	25	142	1.43%
6	141	0.71%	26	140	0.00%	6	140	0.00%	26	140	0.00%
7	142	1.43%	27	142	1.43%	7	141	0.71%	27	142	1.43%
8	142	1.43%	28	141	0.71%	8	140	0.00%	28	141	0.71%
9	142	1.43%	29	143	2.14%	9	141	0.71%	29	140	0.00%
10	142	1.43%	30	142	1.43%	10	140	0.00%	30	140	0.00%
11	142	1.43%	31	141	0.71%	11	141	0.71%	31	140	0.00%
12	142	1.43%	32	140	0.00%	12	140	0.00%	32	140	0.00%
13	143	2.14%	33	142	1.43%	13	141	0.71%	33	141	0.71%
14	141	0.71%	34	142	1.43%	14	140	0.00%	34	140	0.00%
15	141	0.71%	35	142	1.43%	15	140	0.00%	35	140	0.00%
16	143	2.14%	36	142	1.43%	16	141	0.71%	36	141	0.71%
17	141	0.71%	37	141	0.71%	17	140	0.00%	37	141	0.71%
18	141	0.71%	38	141	0.71%	18	140	0.00%	38	140	0.00%
19	141	0.71%	39	144	2.86%	19	141	0.71%	39	140	0.00%
20	141	0.71%	40	141	0.71%	20	140	0.00%	40	141	0.71%
			Average	141.63	1.16%				Average	140.45	0.32%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	100	# of Schedule	500000			# of Activities	100	# of Schedule	1000000		
No	Result	APD	No	Result	APD	No	Result	APD	No	Result	APD
1	140	0.00%	21	140	0.00%	1	140	0.00%	21	140	0.00%
2	140	0.00%	22	140	0.00%	2	140	0.00%	22	140	0.00%
3	141	0.71%	23	140	0.00%	3	140	0.00%	23	140	0.00%
4	140	0.00%	24	141	0.71%	4	140	0.00%	24	140	0.00%
5	140	0.00%	25	140	0.00%	5	140	0.00%	25	140	0.00%
6	140	0.00%	26	140	0.00%	6	140	0.00%	26	140	0.00%
7	141	0.71%	27	140	0.00%	7	140	0.00%	27	140	0.00%
8	140	0.00%	28	140	0.00%	8	140	0.00%	28	140	0.00%
9	140	0.00%	29	140	0.00%	9	140	0.00%	29	140	0.00%
10	141	0.71%	30	140	0.00%	10	140	0.00%	30	140	0.00%
11	141	0.71%	31	140	0.00%	11	140	0.00%	31	140	0.00%
12	140	0.00%	32	140	0.00%	12	140	0.00%	32	140	0.00%
13	141	0.71%	33	140	0.00%	13	140	0.00%	33	140	0.00%
14	141	0.71%	34	140	0.00%	14	140	0.00%	34	140	0.00%
15	140	0.00%	35	141	0.71%	15	140	0.00%	35	140	0.00%
16	140	0.00%	36	140	0.00%	16	140	0.00%	36	140	0.00%
17	140	0.00%	37	140	0.00%	17	140	0.00%	37	140	0.00%
18	140	0.00%	38	140	0.00%	18	140	0.00%	38	140	0.00%
19	140	0.00%	39	140	0.00%	19	140	0.00%	39	140	0.00%
20	140	0.00%	40	140	0.00%	20	140	0.00%	40	140	0.00%
			Average	140.20	0.14%				Average	140.00	0.00%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	500	# of Schedule	10000			# of Activities	500	# of Schedule	50000		
No	Result	APD	No	Result	APD	No	Result	APD	No	Result	APD
1	762	8.86%	21	756	8.00%	1	743	6.14%	21	743	6.14%
2	760	8.57%	22	764	9.14%	2	743	6.14%	22	739	5.57%
3	755	7.86%	23	758	8.29%	3	741	5.86%	23	741	5.86%
4	766	9.43%	24	762	8.86%	4	744	6.29%	24	738	5.43%
5	767	9.57%	25	760	8.57%	5	736	5.14%	25	740	5.71%
6	764	9.14%	26	761	8.71%	6	739	5.57%	26	741	5.86%
7	759	8.43%	27	758	8.29%	7	731	4.43%	27	737	5.29%
8	759	8.43%	28	757	8.14%	8	739	5.57%	28	738	5.43%
9	765	9.29%	29	759	8.43%	9	734	4.86%	29	735	5.00%
10	756	8.00%	30	762	8.86%	10	737	5.29%	30	741	5.86%
11	759	8.43%	31	763	9.00%	11	740	5.71%	31	738	5.43%
12	761	8.71%	32	762	8.86%	12	744	6.29%	32	738	5.43%
13	765	9.29%	33	763	9.00%	13	743	6.14%	33	735	5.00%
14	758	8.29%	34	765	9.29%	14	741	5.86%	34	740	5.71%
15	759	8.43%	35	759	8.43%	15	735	5.00%	35	742	6.00%
16	767	9.57%	36	766	9.43%	16	736	5.14%	36	741	5.86%
17	761	8.71%	37	767	9.57%	17	729	4.14%	37	737	5.29%
18	762	8.86%	38	765	9.29%	18	743	6.14%	38	740	5.71%
19	763	9.00%	39	756	8.00%	19	739	5.57%	39	743	6.14%
20	757	8.14%	40	758	8.29%	20	737	5.29%	40	740	5.71%
			Average	761.15	8.74%				Average	739.03	5.58%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	500	# of Schedule	100000			# of Activities	500	# of Schedule	250000		
No	Result	Deviation	No	Result	Deviation	No	Result	Deviation	No	Result	Deviation
1	730	4.29%	21	730	4.29%	1	717	2.43%	21	716	2.29%
2	732	4.57%	22	730	4.29%	2	723	3.29%	22	723	3.29%
3	731	4.43%	23	731	4.43%	3	720	2.86%	23	728	4.00%
4	729	4.14%	24	730	4.29%	4	724	3.43%	24	721	3.00%
5	733	4.71%	25	726	3.71%	5	717	2.43%	25	718	2.57%
6	731	4.43%	26	731	4.43%	6	721	3.00%	26	716	2.29%
7	721	3.00%	27	727	3.86%	7	721	3.00%	27	714	2.00%
8	731	4.43%	28	724	3.43%	8	722	3.14%	28	714	2.00%
9	729	4.14%	29	727	3.86%	9	723	3.29%	29	717	2.43%
10	724	3.43%	30	726	3.71%	10	717	2.43%	30	726	3.71%
11	733	4.71%	31	730	4.29%	11	724	3.43%	31	719	2.71%
12	724	3.43%	32	728	4.00%	12	721	3.00%	32	720	2.86%
13	726	3.71%	33	729	4.14%	13	720	2.86%	33	724	3.43%
14	730	4.29%	34	733	4.71%	14	719	2.71%	34	717	2.43%
15	727	3.86%	35	731	4.43%	15	719	2.71%	35	720	2.86%
16	726	3.71%	36	733	4.71%	16	717	2.43%	36	725	3.57%
17	730	4.29%	37	730	4.29%	17	717	2.43%	37	720	2.86%
18	727	3.86%	38	731	4.43%	18	717	2.43%	38	722	3.14%
19	728	4.00%	39	730	4.29%	19	721	3.00%	39	714	2.00%
20	732	4.57%	40	731	4.43%	20	715	2.14%	40	723	3.29%
			Average	729.05	4.15%				Average	719.80	2.83%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	500	# of Schedule	500000			# of Activities	500	# of Schedule	1000000		
No	Result	Deviation	No	Result	Deviation	No	Result	Deviation	No	Result	Deviation
1	709	1.29%	21	711	1.57%	1	708	1.14%	21	707	1.00%
2	710	1.43%	22	714	2.00%	2	708	1.14%	22	704	0.57%
3	712	1.71%	23	713	1.86%	3	708	1.14%	23	712	1.71%
4	714	2.00%	24	715	2.14%	4	712	1.71%	24	709	1.29%
5	714	2.00%	25	716	2.29%	5	705	0.71%	25	707	1.00%
6	716	2.29%	26	715	2.14%	6	704	0.57%	26	708	1.14%
7	718	2.57%	27	717	2.43%	7	703	0.43%	27	706	0.86%
8	711	1.57%	28	711	1.57%	8	707	1.00%	28	709	1.29%
9	717	2.43%	29	715	2.14%	9	708	1.14%	29	704	0.57%
10	715	2.14%	30	706	0.86%	10	705	0.71%	30	707	1.00%
11	712	1.71%	31	716	2.29%	11	706	0.86%	31	708	1.14%
12	713	1.86%	32	712	1.71%	12	708	1.14%	32	707	1.00%
13	712	1.71%	33	714	2.00%	13	706	0.86%	33	709	1.29%
14	708	1.14%	34	711	1.57%	14	706	0.86%	34	711	1.57%
15	715	2.14%	35	712	1.71%	15	701	0.14%	35	708	1.14%
16	718	2.57%	36	716	2.29%	16	709	1.29%	36	707	1.00%
17	712	1.71%	37	715	2.14%	17	701	0.14%	37	706	0.86%
18	713	1.86%	38	713	1.86%	18	710	1.43%	38	704	0.57%
19	715	2.14%	39	709	1.29%	19	709	1.29%	39	704	0.57%
20	713	1.86%	40	715	2.14%	20	708	1.14%	40	705	0.71%
			Average	713.33	1.90%				Average	706.85	0.98%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	1000	# of Schedule	10000			# of Activities	1000	# of Schedule	50000		
No	Result	Deviation	No	Result	Deviation	No	Result	Deviation	No	Result	Deviation
1	1557	11.21%	21	1536	9.71%	1	1501	7.21%	21	1502	7.29%
2	1546	10.43%	22	1547	10.50%	2	1497	6.93%	22	1504	7.43%
3	1554	11.00%	23	1547	10.50%	3	1510	7.86%	23	1499	7.07%
4	1552	10.86%	24	1553	10.93%	4	1508	7.71%	24	1508	7.71%
5	1549	10.64%	25	1542	10.14%	5	1510	7.86%	25	1504	7.43%
6	1544	10.29%	26	1555	11.07%	6	1499	7.07%	26	1506	7.57%
7	1548	10.57%	27	1544	10.29%	7	1507	7.64%	27	1506	7.57%
8	1553	10.93%	28	1550	10.71%	8	1500	7.14%	28	1503	7.36%
9	1562	11.57%	29	1555	11.07%	9	1507	7.64%	29	1502	7.29%
10	1547	10.50%	30	1545	10.36%	10	1502	7.29%	30	1505	7.50%
11	1553	10.93%	31	1551	10.79%	11	1504	7.43%	31	1502	7.29%
12	1542	10.14%	32	1541	10.07%	12	1500	7.14%	32	1508	7.71%
13	1545	10.36%	33	1562	11.57%	13	1489	6.36%	33	1501	7.21%
14	1540	10.00%	34	1555	11.07%	14	1511	7.93%	34	1502	7.29%
15	1545	10.36%	35	1557	11.21%	15	1498	7.00%	35	1513	8.07%
16	1542	10.14%	36	1546	10.43%	16	1506	7.57%	36	1506	7.57%
17	1555	11.07%	37	1555	11.07%	17	1508	7.71%	37	1502	7.29%
18	1551	10.79%	38	1572	12.29%	18	1510	7.86%	38	1502	7.29%
19	1550	10.71%	39	1547	10.50%	19	1500	7.14%	39	1498	7.00%
20	1548	10.57%	40	1550	10.71%	20	1507	7.64%	40	1504	7.43%
			Average	1549.83	10.70%				Average	1503.78	7.41%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	1000	# of Schedule	100000			# of Activities	1000	# of Schedule	250000		
No	Result	Deviation	No	Result	Deviation	No	Result	Deviation	No	Result	Deviation
1	1483	5.93%	21	1482	5.86%	1	1468	4.86%	21	1463	4.50%
2	1481	5.79%	22	1483	5.93%	2	1466	4.71%	22	1459	4.21%
3	1476	5.43%	23	1484	6.00%	3	1462	4.43%	23	1455	3.93%
4	1477	5.50%	24	1479	5.64%	4	1467	4.79%	24	1461	4.36%
5	1485	6.07%	25	1481	5.79%	5	1452	3.71%	25	1464	4.57%
6	1479	5.64%	26	1477	5.50%	6	1467	4.79%	26	1456	4.00%
7	1483	5.93%	27	1479	5.64%	7	1466	4.71%	27	1460	4.29%
8	1472	5.14%	28	1481	5.79%	8	1451	3.64%	28	1469	4.93%
9	1485	6.07%	29	1480	5.71%	9	1467	4.79%	29	1452	3.71%
10	1486	6.14%	30	1484	6.00%	10	1457	4.07%	30	1458	4.14%
11	1483	5.93%	31	1486	6.14%	11	1462	4.43%	31	1462	4.43%
12	1486	6.14%	32	1489	6.36%	12	1461	4.36%	32	1461	4.36%
13	1490	6.43%	33	1470	5.00%	13	1462	4.43%	33	1457	4.07%
14	1487	6.21%	34	1480	5.71%	14	1449	3.50%	34	1465	4.64%
15	1482	5.86%	35	1482	5.86%	15	1457	4.07%	35	1455	3.93%
16	1478	5.57%	36	1484	6.00%	16	1458	4.14%	36	1451	3.64%
17	1480	5.71%	37	1478	5.57%	17	1469	4.93%	37	1464	4.57%
18	1481	5.79%	38	1484	6.00%	18	1452	3.71%	38	1456	4.00%
19	1478	5.57%	39	1483	5.93%	19	1453	3.79%	39	1457	4.07%
20	1476	5.43%	40	1476	5.43%	20	1460	4.29%	40	1448	3.43%
			Average	1481.25	5.80%				Average	1459.48	4.25%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	1000	# of Schedule	500000			# of Activities	1000	# of Schedule	1000000		
No	Result	Deviation	No	Result	Deviation	No	Result	Deviation	No	Result	Deviation
1	1452	3.71%	21	1444	3.14%	1	1431	2.21%	21	1434	2.43%
2	1449	3.50%	22	1448	3.43%	2	1435	2.50%	22	1433	2.36%
3	1450	3.57%	23	1440	2.86%	3	1429	2.07%	23	1435	2.50%
4	1443	3.07%	24	1446	3.29%	4	1435	2.50%	24	1433	2.36%
5	1447	3.36%	25	1444	3.14%	5	1433	2.36%	25	1430	2.14%
6	1454	3.86%	26	1449	3.50%	6	1435	2.50%	26	1434	2.43%
7	1439	2.79%	27	1441	2.93%	7	1427	1.93%	27	1435	2.50%
8	1440	2.86%	28	1443	3.07%	8	1435	2.50%	28	1434	2.43%
9	1441	2.93%	29	1447	3.36%	9	1433	2.36%	29	1429	2.07%
10	1444	3.14%	30	1452	3.71%	10	1434	2.43%	30	1435	2.50%
11	1447	3.36%	31	1446	3.29%	11	1433	2.36%	31	1441	2.93%
12	1455	3.93%	32	1443	3.07%	12	1426	1.86%	32	1436	2.57%
13	1444	3.14%	33	1438	2.71%	13	1428	2.00%	33	1430	2.14%
14	1441	2.93%	34	1445	3.21%	14	1433	2.36%	34	1433	2.36%
15	1449	3.50%	35	1454	3.86%	15	1431	2.21%	35	1429	2.07%
16	1444	3.14%	36	1445	3.21%	16	1428	2.00%	36	1432	2.29%
17	1446	3.29%	37	1437	2.64%	17	1437	2.64%	37	1434	2.43%
18	1449	3.50%	38	1450	3.57%	18	1432	2.29%	38	1435	2.50%
19	1447	3.36%	39	1451	3.64%	19	1440	2.86%	39	1436	2.57%
20	1441	2.93%	40	1447	3.36%	20	1430	2.14%	40	1432	2.29%
			Average	1445.80	3.27%				Average	1432.88	2.35%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	1500	# of Schedule	10000			# of Activities	1500	# of Schedule	50000		
No	Result	Deviation	No	Result	Deviation	No	Result	Deviation	No	Result	Deviation
1	2342	11.52%	21	2341	11.48%	1	2274	8.29%	21	2285	8.81%
2	2346	11.71%	22	2329	10.90%	2	2270	8.10%	22	2289	9.00%
3	2331	11.00%	23	2321	10.52%	3	2268	8.00%	23	2282	8.67%
4	2340	11.43%	24	2348	11.81%	4	2287	8.90%	24	2284	8.76%
5	2338	11.33%	25	2337	11.29%	5	2260	7.62%	25	2284	8.76%
6	2321	10.52%	26	2325	10.71%	6	2280	8.57%	26	2288	8.95%
7	2350	11.90%	27	2343	11.57%	7	2286	8.86%	27	2274	8.29%
8	2354	12.10%	28	2335	11.19%	8	2287	8.90%	28	2282	8.67%
9	2353	12.05%	29	2340	11.43%	9	2285	8.81%	29	2276	8.38%
10	2333	11.10%	30	2353	12.05%	10	2293	9.19%	30	2283	8.71%
11	2344	11.62%	31	2327	10.81%	11	2284	8.76%	31	2304	9.71%
12	2372	12.95%	32	2358	12.29%	12	2278	8.48%	32	2272	8.19%
13	2346	11.71%	33	2346	11.71%	13	2275	8.33%	33	2276	8.38%
14	2329	10.90%	34	2338	11.33%	14	2280	8.57%	34	2275	8.33%
15	2337	11.29%	35	2337	11.29%	15	2284	8.76%	35	2277	8.43%
16	2356	12.19%	36	2351	11.95%	16	2284	8.76%	36	2283	8.71%
17	2346	11.71%	37	2346	11.71%	17	2287	8.90%	37	2284	8.76%
18	2326	10.76%	38	2341	11.48%	18	2278	8.48%	38	2280	8.57%
19	2356	12.19%	39	2329	10.90%	19	2294	9.24%	39	2285	8.81%
20	2343	11.57%	40	2340	11.43%	20	2290	9.05%	40	2272	8.19%
			Average	2341.20	11.49%				Average	2281.48	8.64%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	1500	# of Schedule	100000			# of Activities	1500	# of Schedule	250000		
No	Result	Deviation	No	Result	Deviation	No	Result	Deviation	No	Result	Deviation
1	2250	7.14%	21	2258	7.52%	1	2212	5.33%	21	2206	5.05%
2	2267	7.95%	22	2256	7.43%	2	2197	4.62%	22	2196	4.57%
3	2248	7.05%	23	2265	7.86%	3	2216	5.52%	23	2216	5.52%
4	2241	6.71%	24	2248	7.05%	4	2214	5.43%	24	2216	5.52%
5	2245	6.90%	25	2251	7.19%	5	2198	4.67%	25	2209	5.19%
6	2246	6.95%	26	2253	7.29%	6	2207	5.10%	26	2209	5.19%
7	2264	7.81%	27	2253	7.29%	7	2209	5.19%	27	2207	5.10%
8	2252	7.24%	28	2252	7.24%	8	2204	4.95%	28	2212	5.33%
9	2259	7.57%	29	2258	7.52%	9	2209	5.19%	29	2206	5.05%
10	2247	7.00%	30	2267	7.95%	10	2208	5.14%	30	2208	5.14%
11	2256	7.43%	31	2256	7.43%	11	2215	5.48%	31	2209	5.19%
12	2258	7.52%	32	2242	6.76%	12	2205	5.00%	32	2210	5.24%
13	2262	7.71%	33	2246	6.95%	13	2200	4.76%	33	2206	5.05%
14	2252	7.24%	34	2259	7.57%	14	2210	5.24%	34	2215	5.48%
15	2259	7.57%	35	2248	7.05%	15	2213	5.38%	35	2198	4.67%
16	2251	7.19%	36	2249	7.10%	16	2203	4.90%	36	2214	5.43%
17	2244	6.86%	37	2258	7.52%	17	2197	4.62%	37	2210	5.24%
18	2258	7.52%	38	2246	6.95%	18	2214	5.43%	38	2208	5.14%
19	2256	7.43%	39	2236	6.48%	19	2210	5.24%	39	2209	5.19%
20	2254	7.33%	40	2237	6.52%	20	2213	5.38%	40	2204	4.95%
			Average	2252.68	7.27%				Average	2208.05	5.15%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	1500	# of Schedule	500000			# of Activities	1500	# of Schedule	# of Schedule	100000	
No	Result	Deviation	No	Result	Deviation	No	Result	Deviation	No	Result	Deviation
1	2184	4.00%	21	2186	4.10%	1	2162	2.95%	21	2171	3.38%
2	2187	4.14%	22	2185	4.05%	2	2167	3.19%	22	2163	3.00%
3	2188	4.19%	23	2185	4.05%	3	2165	3.10%	23	2167	3.19%
4	2183	3.95%	24	2190	4.29%	4	2168	3.24%	24	2169	3.29%
5	2182	3.90%	25	2182	3.90%	5	2169	3.29%	25	2171	3.38%
6	2186	4.10%	26	2180	3.81%	6	2168	3.24%	26	2165	3.10%
7	2182	3.90%	27	2184	4.00%	7	2168	3.24%	27	2163	3.00%
8	2187	4.14%	28	2187	4.14%	8	2165	3.10%	28	2168	3.24%
9	2184	4.00%	29	2186	4.10%	9	2159	2.81%	29	2170	3.33%
10	2180	3.81%	30	2184	4.00%	10	2168	3.24%	30	2168	3.24%
11	2183	3.95%	31	2186	4.10%	11	2169	3.29%	31	2164	3.05%
12	2186	4.10%	32	2179	3.76%	12	2176	3.62%	32	2171	3.38%
13	2184	4.00%	33	2181	3.86%	13	2161	2.90%	33	2172	3.43%
14	2184	4.00%	34	2185	4.05%	14	2162	2.95%	34	2165	3.10%
15	2182	3.90%	35	2184	4.00%	15	2175	3.57%	35	2162	2.95%
16	2188	4.19%	36	2182	3.90%	16	2165	3.10%	36	2168	3.24%
17	2186	4.10%	37	2183	3.95%	17	2167	3.19%	37	2169	3.29%
18	2188	4.19%	38	2184	4.00%	18	2172	3.43%	38	2164	3.05%
19	2184	4.00%	39	2188	4.19%	19	2164	3.05%	39	2163	3.00%
20	2182	3.90%	40	2190	4.29%	20	2162	2.95%	40	2164	3.05%
			Average	2184.53	4.03%				Average	2166.73	3.18%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	2000	# of Schedule	10000			# of Activities	2000	# of Schedule	50000		
No	Result	Deviation	No	Result	Deviation	No	Result	Deviation	No	Result	Deviation
1	3134	11.93%	21	3145	12.32%	1	3050	8.93%	21	3062	9.36%
2	3124	11.57%	22	3137	12.04%	2	3064	9.43%	22	3074	9.79%
3	3144	12.29%	23	3120	11.43%	3	3072	9.71%	23	3076	9.86%
4	3122	11.50%	24	3128	11.71%	4	3058	9.21%	24	3074	9.79%
5	3129	11.75%	25	3145	12.32%	5	3046	8.79%	25	3069	9.61%
6	3130	11.79%	26	3132	11.86%	6	3067	9.54%	26	3070	9.64%
7	3126	11.64%	27	3146	12.36%	7	3058	9.21%	27	3072	9.71%
8	3142	12.21%	28	3153	12.61%	8	3068	9.57%	28	3063	9.39%
9	3137	12.04%	29	3145	12.32%	9	3063	9.39%	29	3056	9.14%
10	3113	11.18%	30	3137	12.04%	10	3053	9.04%	30	3071	9.68%
11	3156	12.71%	31	3138	12.07%	11	3049	8.89%	31	3053	9.04%
12	3150	12.50%	32	3130	11.79%	12	3042	8.64%	32	3062	9.36%
13	3122	11.50%	33	3127	11.68%	13	3062	9.36%	33	3064	9.43%
14	3117	11.32%	34	3143	12.25%	14	3064	9.43%	34	3074	9.79%
15	3140	12.14%	35	3133	11.89%	15	3080	10.00%	35	3071	9.68%
16	3127	11.68%	36	3123	11.54%	16	3077	9.89%	36	3070	9.64%
17	3140	12.14%	37	3126	11.64%	17	3051	8.96%	37	3065	9.46%
18	3122	11.50%	38	3125	11.61%	18	3063	9.39%	38	3078	9.93%
19	3159	12.82%	39	3137	12.04%	19	3060	9.29%	39	3073	9.75%
20	3141	12.18%	40	3119	11.39%	20	3064	9.43%	40	3061	9.32%
			Average	3134.10	11.93%				Average	3064.23	9.44%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	2000		# of Schedule	100000		# of Activities	2000		# of Schedule	250000	
No	Result	Deviation	No	Result	Deviation	No	Result	Deviation	No	Result	Deviation
1	3020	7.86%	21	3024	8.00%	1	2969	6.04%	21	2946	5.21%
2	3024	8.00%	22	3020	7.86%	2	2970	6.07%	22	2962	5.79%
3	3024	8.00%	23	3016	7.71%	3	2972	6.14%	23	2964	5.86%
4	3023	7.96%	24	3026	8.07%	4	2959	5.68%	24	2962	5.79%
5	3017	7.75%	25	3026	8.07%	5	2967	5.96%	25	2971	6.11%
6	3020	7.86%	26	3025	8.04%	6	2962	5.79%	26	2969	6.04%
7	3030	8.21%	27	3023	7.96%	7	2954	5.50%	27	2965	5.89%
8	3015	7.68%	28	3014	7.64%	8	2959	5.68%	28	2970	6.07%
9	3016	7.71%	29	3018	7.79%	9	2963	5.82%	29	2969	6.04%
10	3022	7.93%	30	3021	7.89%	10	2965	5.89%	30	2975	6.25%
11	3021	7.89%	31	3015	7.68%	11	2959	5.68%	31	2974	6.21%
12	3020	7.86%	32	3026	8.07%	12	2959	5.68%	32	2964	5.86%
13	3036	8.43%	33	3028	8.14%	13	2956	5.57%	33	2967	5.96%
14	3018	7.79%	34	3019	7.82%	14	2956	5.57%	34	2960	5.71%
15	3019	7.82%	35	3022	7.93%	15	2960	5.71%	35	2961	5.75%
16	3023	7.96%	36	3025	8.04%	16	2965	5.89%	36	2961	5.75%
17	3019	7.82%	37	3025	8.04%	17	2963	5.82%	37	2967	5.96%
18	2998	7.07%	38	3018	7.79%	18	2959	5.68%	38	2966	5.93%
19	3021	7.89%	39	3019	7.82%	19	2960	5.71%	39	2969	6.04%
20	3017	7.75%	40	3022	7.93%	20	2962	5.79%	40	2957	5.61%
			Average	3020.88	7.89%				Average	2963.45	5.84%

Table B.1. Experiments for All Schedule Numbers (Continued)

# of Activities	2000		# of Schedule	500000		# of Activities	2000		# of Schedule	1000000	
No	Result	Deviation	No	Result	Deviation	No	Result	Deviation	No	Result	Deviation
1	2936	4.86%	21	2928	4.57%	1	2917	4.18%	21	2912	4.00%
2	2923	4.39%	22	2927	4.54%	2	2903	3.68%	22	2910	3.93%
3	2936	4.86%	23	2930	4.64%	3	2918	4.21%	23	2904	3.71%
4	2930	4.64%	24	2934	4.79%	4	2914	4.07%	24	2910	3.93%
5	2919	4.25%	25	2926	4.50%	5	2908	3.86%	25	2914	4.07%
6	2926	4.50%	26	2933	4.75%	6	2910	3.93%	26	2915	4.11%
7	2931	4.68%	27	2932	4.71%	7	2926	4.50%	27	2906	3.79%
8	2932	4.71%	28	2930	4.64%	8	2913	4.04%	28	2908	3.86%
9	2943	5.11%	29	2932	4.71%	9	2911	3.96%	29	2902	3.64%
10	2936	4.86%	30	2925	4.46%	10	2911	3.96%	30	2904	3.71%
11	2938	4.93%	31	2928	4.57%	11	2906	3.79%	31	2916	4.14%
12	2930	4.64%	32	2931	4.68%	12	2895	3.39%	32	2914	4.07%
13	2934	4.79%	33	2929	4.61%	13	2915	4.11%	33	2912	4.00%
14	2934	4.79%	34	2922	4.36%	14	2909	3.89%	34	2908	3.86%
15	2924	4.43%	35	2926	4.50%	15	2910	3.93%	35	2910	3.93%
16	2930	4.64%	36	2928	4.57%	16	2912	4.00%	36	2911	3.96%
17	2928	4.57%	37	2933	4.75%	17	2907	3.82%	37	2910	3.93%
18	2934	4.79%	38	2925	4.46%	18	2902	3.64%	38	2902	3.64%
19	2924	4.43%	39	2932	4.71%	19	2897	3.46%	39	2908	3.86%
20	2931	4.68%	40	2928	4.57%	20	2906	3.79%	40	2914	4.07%
			Average	2929.95	4.64%				Average	2909.50	3.91%