ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

Menşure Zühal ERİŞGİN BARAK

FUZZY ORDER ACCEPTANCE AND SCHEDULING ON IDENTICAL PARALLEL MACHINES

DEPARTMENT OF INDUSTRIAL ENGINEERING

ADANA, 2021

ABSTRACT

PhD. THESIS

FUZZY ORDER ACCEPTANCE AND SCHEDULING ON IDENTICAL PARALLEL MACHINES

Menşure Zühal ERİŞGİN BARAK

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF INDUSTRIAL ENGINEERING

Supervisor : Assoc. Prof. Dr. Melik Koyuncu
Year: 2021, Page: 242
Jury : Assoc. Prof. Dr. Melik KOYUNCU
: Prof. Dr. Ali KOKANGÜL
: Asst. Prof. Dr. Fikri EGE
: Assoc. Prof. Dr. Zahide Figen ANTMEN
: Asst. Prof. Dr. Esra KARAKAŞ

This thesis is a study about simultaneous fuzzy order acceptance and scheduling in identical parallel machines. The research problem is about scheduling orders in a make-to-order system with capacity constraints. Due to the limited capacity, some orders can be produced beyond their due date. So, the manufacturer has to pay or discount this delay cost. The manufacturer has to select a subset of orders to have the maximum net profit. The tardiness of orders or outsourcing penalty cost decrease the revenue of orders. The orders coming from the customer are characterized by their revenue, outsourcing penalty cost, weighted delay cost, fuzzy due date, fuzzy processing times, and fuzzy sequence-dependent setup times. OAS on single machine problem is strongly NP-Hard, as a result, FOAS-IPM is also a strongly NP-Hard problem. In this study, first, mixed-integer linear programming is formulated to solve the problem. Next, two genetic algorithms with the name of the GADOC and GATPC algorithms are proposed to solve the large size instances. The performance of the algorithms is compared concerning the total revenue. With extensive computational tests, the proposed algorithms produce and provide high-quality feasible solutions even in big-size instances.

Key Words: Fuzzy order acceptance and scheduling, identical parallel machine scheduling, heuristic approach, mathematical modeling

ÖZ

DOKTORA TEZİ

BAĞLANTISIZ EŞ PARALEL MAKİNELERDE BULANIK SİPARİŞ KABUL VE ÇİZELGELEME

Menşure Zühal ERİŞGİN BARAK

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Doç. Dr. Melik KOYUNCU
Yıl: 2021, Sayfa: 242
Jüri : Doç. Dr. Melik KOYUNCU
: Prof. Dr. Ali KOKANGÜL
: Dr. Öğr. Üyesi Fikri EGE
: Doç. Dr. Zahide Figen ANTMEN
: Dr. Öğr. Üyesi Esra KARAKAŞ

Bu tez, bulanık sipariş kabul ve çizelgeleme problemini özdeş parallel makineli sistemde ele almıştır. Bu çalışma sipariş üzerine üretim yapan ve kapasite kısıtı olan bir sistemde siparişlerin çizelgelenmesi üzerinedir. Kapasite kısıtı nedeniyle, bazı siparişler teslim edilmesi gereken tarihten daha sonra teslim edilmek zorunda kalınmaktadır. Bu durumda firma müşterisine geç kalma süresi kadar bir gecikme cezası ödemektedir. Firma gecikme cezası ödememek ve müşteriye ürünü geç teslim etmemek için, üretilebileceği belli bir grup siparişi çizelgelemek ve siparişlerden maksimum kar elde etmek için bazı siparişleri seçmek zorundadır. Net karı azaltan gecikme cezası ve dış kaynak kullanımıdır. Müşteriden firmaya ulaşan siparişlerin kendilerine özgü, getirisi, dış kaynak kullanım cezası, ağırlıklandırılmış gecikme cezası, bulanık teslim tarihi, bulanık üretim süresi, ve bulanık sıraya bağlı set up süresi vardır. Sipariş kabul ve çizelgeleme probleminin tek makineli sistemde NP-Hard olduğu gösterilmiştir. Bu nedenle çok makineli sistemde çalışılan bu problem oldukça zor NP-Hard bir problemdir. Problemin çözümünde karışık tam sayılı matematiksel model geliştirilmiştir. Daha sonra, büyük veri setlerini çözebilmek için GADOC ve GATPC isimli iki genetik algoritma geliştirilmiştir. Algoritmaların performansları toplam getiriye olan kıyasları ile değerlendirilmiştir. Çok sayıda veri setleri ile denenmesi sonucu, geliştirilen çözüm metotlarının problem çözümünde oldukça etkili, yüksek kalitede, ve kısa sürede olurlu sonuç sunduğu gösterilmiştir.

Anahtar Kelimeler: Bulanık sipariş kabul ve çizelgeleme, özdeş paralel makine çizelgeleme, sezgisel yaklaşım, matematiksel modelleme

EXTENDED ABSTRACT

For more than 30 years, the order acceptance and scheduling problem has been one of the most remarkable machine scheduling studies. Machine scheduling problems are one of the most complex problems by their nature. The parameters and purpose of the scheduling problem determined according to the type of production system vary considerably. The number of orders with different features is very high in a make-to-order system. In addition, companies with capacity constraints suffer from customer-related relationships and reputations due to late delivery orders. Companies that want to deal with capacity constraints seek solutions to deliver their orders on time. Due to capacity constraints, some orders have to be rejected or outsourced. The manufacturer has to select a subset of orders to have the maximum net profit. However, it is not easy to make this decision. The tardiness of orders or outsourcing penalty cost decrease the revenue of orders. Timely delivery is possible with good production scheduling. Scheduling critical to timely delivery is inherently difficult. The company aims to maximize the desired profit while scheduling. Finding the schedule with the highest profit is a very difficult and complex optimization problem.

In a make-to-order company orders reach the company before planning. Scheduling is made by considering the characteristics of the orders in the previously known order set. The orders coming from the customer are characterized by their revenue, outsourcing penalty cost, weighted delay cost, fuzzy due date, fuzzy processing times, and fuzzy sequence-dependent setup times. Fuzzy problem parameters are defined as triangular fuzzy numbers. In this study, the signed distance method is used to be able to compare the fuzzy equations and fuzzy parameters. Revenue and all other parameters are different from each order.

OAS on single machine problem is strongly NP-Hard, as a result, FOAS-IPM (Fuzzy Order Acceptance and Scheduling on Identical Parallel Machine) is also a strongly NP-Hard problem.

The thesis provides optimization and scheduling on the fuzzy order acceptance problem on identical parallel machines. Firstly, an order acceptance and scheduling problem is formulized as mixed-integer linear programming to have the maximum net profit with the scheduling. In this problem, a developed mathematical model enables optimization in terms of both profit and time in the production schedules developed for real-life production planning. On the other hand, it also includes outsourcing costs of orders that have to be rejected. Fuzzy times that exist in real-life systems are handled in this model. For this reason, modeling that is more up-to-date and close to real-life systems has been made than the order acceptance and scheduling studies examined in the literature to date. Besides, as in real life, having more than one machine in the system makes the model more realistic. In this study, first, mixed-integer linear programming is formulated to solve the problem. The mathematical model developed has found optimum scheduling successfully. The scheduling found gave the optimum result to achieve the highest possible profit. However, this has been possible for CPLEX in solving small data sets with 10 orders. To cope with this situation, the genetic algorithm, which is a heuristic algorithm, has been used. GA is a heuristic approach also especially mostly used in optimization problems. Heuristic algorithms are used in scheduling problems due to the large solution space. GA is one of the most used algorithms in machine scheduling problems. Two genetic algorithms with the name the GADOC and GATPC algorithms are proposed to solve the large size instances. 2 different genetic algorithms using 2 different crossover methods have been developed. The performance of the algorithms is compared concerning the total revenue. With extensive computational tests, the proposed algorithms produce and provide high-quality feasible solutions. Three solution methods were tested with small data sets and they were compared according to their solving performance and execution time.

The upper bound deviation is used to compare the solving performance. The upper bound deviation is the portion between the result and the total revenue of each order set. The upper bound deviation calculation is calculated by proportioning the sum of the returns of the orders in the data set and the difference in the results of the algorithm to the total revenue. As a result of this proportioning, the closest to the desired total revenue and the highest net profit as a result of this, the smallest upper bound deviation is desired. Orders of 10 have been resolved with CPLEX, GADOC, and GATCP algorithm, and 3 algorithm performances are compared. The CPLEX algorithm is the best solution for data sets with a slight difference from the GADOC algorithm. GATPC is an algorithm that solves the instances the worst. In large data sets, GADOC and GATPC were compared and the GADOC algorithm performs better. Thus GADOC algorithm with Davis Order Crossover is more successful than the GATPC algorithm with a two-point crossover in finding correct results in a multi-machine system. Although the number of orders and number of machines in the data set varies, GADOC works better and faster than the GATPC algorithm in terms of execution speed and upper bound deviation.

Since the GADOC algorithm offers a better solution than the GATPC algorithm, a more detailed performance study has been carried out for the GADOC algorithm. The GADOC performance in the number of different iterations and the starting population has been studied in more detail. The optimum result can be achieved with the higher starting population and low iteration number and it performs better than the high number of iterations and low initial population number. Therefore, to increase the probability of GADOC and GATPC algorithms to reach optimum results, a high initial population number and appropriate iteration number were determined.

Sensitivity analysis, which offers alternatives for decision-makers, was also carried out in this study. Sensitivity analysis has been performed to examine optimum results in different scenarios. In sensitivity analysis, the effect of changing the objective function coefficients on the objective function. Also after the optimum

result is found, different sequencing alternatives are presented by keeping the value of the objective function constant.

This paper also presents both theoretical and practical implications of production scheduling. This study deals with scheduling, which is the most important point for production planning, in terms of net profit maximization, which is the main objective of the firm. Uncertainties in real life have been included in this study and the durations have been studied as indefinite. On the other hand, unlike other orders and acceptance and scheduling, multiple machine systems have been handled. Also, an outsourcing policy has been considered for rejected orders.

In this study, the success of the scheduling problem between two different crossover methods in the genetic algorithm has been demonstrated. In scheduling problems, the high difference between the chromosomes between generations increases the success of GA. The success of the elitism strategy has also been demonstrated in this study. It has been shown that it is an advantage to transfer the best individuals over generations. In addition, the importance of different iteration numbers and initial population size is shown.

The problem of order acceptance and scheduling has been addressed by dealing with fuzzy logic and outsourcing in this paper which can be an inspiring model for future studies. While establishing the model in this study, some assumptions and problems in some systems are included. The system is modeled as close to reality.

ACKNOWLEDGEMENT

I would like to express my special thanks to my supervisor Assoc. Prof. Dr. Melik KOYUNCU for his insightful comments and endless support of my Ph.D. study and all education life. I also want to offer my deepest gratitude for the guidance provided me as part of my thesis committee to Prof. Dr. Ali Kokangül, Asst. Prof. Dr. Fikri Ege, Assoc. Prof. Dr. Z. Figen Antmen, and Asst. Prof. Dr. Esra Karakaş. Finally, I would like to express my special gratitude to my family because of their love, deep understanding, and heartily support in my whole life.

TABLE OF CONTENTS **PAGE**

LIST OF TABLES **PAGE**

LIST OF FIGURES **PAGE**

LIST OF GRAPH **PAGE**

LIST OF ABBREVIATIONS AND NOMENCLATURE

FOASIPM : Fuzzy Order Acceptance And Scheduling On Identical Parallel Machine

MILP :Mix Integer Linear Programming

GADOC : Genetic Algorithm With Davis Order Crossover

GATPC : Genetic Algorithm With Two Point Crossover

GA : Genetic Algorithm

O : Order Set Index

M : Machine Set Index

i : Order Index

j : Order Index

m : Machine Index

W : The Weighted Delay Penalty Cost Of Order

Co : The Outsourced Penalty Cost Of Order

Ei : The Revenue Of Order

$\tilde{P}$:The Fuzzy Parameter Of Production Times Of Order

$\tilde{D}$:The Fuzzy Parameter Of The Due Date Of Order

$\tilde{S}_0$: The Fuzzy Parameter Of Beginning Setup Times Of Order

$\tilde{S}$: The Fuzzy Parameter Of Sequence-Dependent Setup Times Of

B : Very Big Positive Number

A : Average Of Assigned Order To Each Machine

C: : The Completion Time Of Order

T: : The Delay Time Of Order

1. INTRODUCTION

Order Acceptance and Scheduling (OAS) is a topic that combines the goals of production planning and marketing departments, which have been studied for over 30 years in the literature. (Slotnick, 2011) There are two goals in the OAS problem. The first goal is to increase customer satisfaction with full-time delivery. The second goal is to create a production plan. So, the common purpose of these two goals is the maximization of the firm's profit. On-time delivery is a very key point in customer relations and reliability. On-time delivery is the result of a carefully prepared production planning process.

Well-defined scheduling is the fundamental element of production planning. The information about which product will be produced and delivered when is very critical information for both the production department and the marketing department. On the other hand, as a result of good production scheduling, the company can take other necessary future decisions and make plans. For example, in case of a delay in delivering the order, the company can decide on outsourcing the products that will be delayed in the production process. The delay can cause to loss of the customer or worsen the relation. In this case, production scheduling is one of the most basic elements for a manufacturing company. For this reason, the problem of OAS, which is very valuable in terms of both the production and marketing department, is discussed in this study.

One of the biggest problems for companies is capacity constraints. Firms with capacity constraints cannot manufacture all of the incoming orders. Products that cannot be produced or produced late are delivered to the customer late. Late delivery or rejecting the order negatively affects the customer relations of the companies. For this reason, firms are looking for solutions to cope with capacity constraints and increase profits.

Due to increasing competition in the developing global market, companies have been searching for various methods to boost profit by overcoming capacity

constraints. Today's competitive conditions, increasing product and service prices, shorter delivery times, order manufacturing, and many other reasons lead businesses to improve their production processes. Especially limited capacities companies have been desperate in the face of increasing demands, customer satisfaction levels, and competition in markets. So, businesses have to optimize their production processes correctly and sustainably to reach their goals faster.

After long searches, production planning studies offer different opportunities to companies with limited capacities. Production planning manages all the processes that enable businesses to produce products or services at the desired time, in the desired amount, in the desired quality, as well as save time and cost, and anticipate both customer expectations and business goals of producers. By the time with production planning methods, firms got acquainted with production planning techniques that will increase their profits, decrease their makespan, or the other improvements in the satisfaction of customers to increase their market share.

Production planning is the work carried out to obtain goods and services on time to meet human needs. So, effective production planning is crucial to decide whether or not the production company takes into account the customer's needs and satisfaction. Businesses that aim to meet the needs of society must plan and manage their production functions correctly to produce goods or services on time.

Production planning is crucial for the manufacturing process. For businesses that produce goods or services, production planning includes all stages from order to shipment, planning, and guiding them. Production management enables enterprises to combine materials and manpower resources by using limited raw materials to produce at the desired quality, at the requested time, at the lowest budget, and with the highest profitability. Instead of increasing capacity, more effective production planning became appealing solutions for the companies to decide on outsourcing or other solutions.

To achieve their goals, businesses must determine the quantity and type of goods or services produced, plan demand forecasts, analyze how much and when the

products should be manufactured. The firms can increase their business efficiency by determining their production capacity correctly. Because the needs of the companies in the sector are different from each other, there are differences in the problems that are sought for solutions.

Today, some companies have the strategy of make to order, make to stock, and besides, some companies have the strategy for both. Unlike the production methods of the companies, the purposes of the companies also change according to their needs. Companies that want to have a large number of customers in the market aim to diversify their products and produce a large number of orders. Since some companies want to realize their productions in a short time, they accept orders to be produced in minimum time and the objective of these firms is to minimize production times. Whereas, some companies want to realize their production with the highest profit. So their objective becomes to have the highest net profit. Today, firms generally aim to maximize their profit by increasing customer loyalty and decreasing the penalty cost and manufacturing cost.

Production scheduling is critical to the firm's achievement of its goals. It is important to deliver the product to the customer on time, the arrival and delivery dates of the stocks, the calculation of the intermediate stocks, and the control of all production process activities.

Production scheduling provides many advantages to the company. With the scheduling to be created based on the company's own master production plan, questions such as when and which work can start, which resources should be ready when the works will be processed and when these works will be completed can be answered. As a result of these answers, it can be determined whether the company will deliver to the customer on time or not. So, the company has the opportunity to make other plans for received orders that cannot be produced or be delivered late. The company can also have the advantage of deciding on outsourcing according to the plan to be obtained from scheduling. The scheduling provides preliminary data to the company, making it easy for future moves and sheds light on other plans. Other

advantages of scheduling are determining the work centers where the works will be assigned, determining when the products will be delivered to the customers and calculating the delay times, minimizing the sequence-dependent set-up times between the products, minimizing the inventory stock levels, making the right assignment to use the personnel and equipment effectively.

This thesis with the subject of Order Acceptance and Scheduling (OAS) is a production planning problem about which orders to accept for processing and how to schedule them. Production planning is crucial for the manufacturing process. For businesses that produce goods or services, production planning includes all stages from order to shipment, planning, and guiding them. Production management enables enterprises to combine materials and manpower resources by using limited raw materials to produce at the desired quality, at the requested time, at the lowest budget, and with the highest profitability. Instead of increasing capacity, more effective production planning became an appealing solution for the companies.

The OAS problem has been conducted in the literature for more than 3 decades. In make-to-order (MTO) firms, the orders are usually a big wide range with their revenues, production time, due dates, and other features. It becomes challenging to increase the revenues in firms with limited capacities. So, it is necessary to develop production planning methods that will enable the production to cope with capacity constraints and increase profits. Therefore, the necessity to work on which orders should be accepted and how to schedule these accepted orders in enterprises has arisen. In the presented study, the fuzzy order acceptance and scheduling problem on the identical parallel machines (FOAS-IPM) has been conducted and a new mathematical model is presented.

Today, there are many different types of production systems according to their features. However, in high competition markets, the number of production systems of Make-to-Order is increasing day by day to keep customer satisfaction high and to respond to different order types. This situation brings about a high variety of orders. Being able to produce a large number and variety of orders increases the

variety of machinery in production systems. Even if there are products on the same machine, the change of tools in the machines creates challenging set-up times. The production sequence of the products also affects setup times. Even when many products are produced on the same machine, calculating the total production time becomes a very challenging problem. Already, the problems that include sequence-dependent setup times are defined as NP-hard in the literature. (Slotnick, 2011)

In general, in scheduling problems, the estimated or measured times for the operation time of each product cannot be the same even in every measurement. Within the product's operation parameters such as processing time, preparation time is not possible to predict or measure deterministically, although it is generally assumed to be deterministic to facilitate modeling and solution of problems. However, this assumption causes the problem to diverge from reality. Production times generally have differences even for the same process of the same product due to measurement differences or due to human hand. For this reason, in this study, the production time of each product and the set-up times depending on the sequence are studied with fuzzy numbers to model more closely to reality.

In this study, the company makes a production schedule to deliver its products on the delivery date. The purpose of the scheduling in this study is profit maximization. Products are included in the schedule until the profit obtained for late products and the delay discount are equal to the product's profit. However, products that cause loss instead of profit as a result of delay are outsourced due to delivering to the customer at least on the delivery date. Thus, the company, which bears the price difference for outsourcing, has the advantage of keeping customer satisfaction high as it can deliver to the customer on time. Unlike other production planning scheduling problems, the FOAS-IPM problem aims at more general customer satisfaction and profit maximization of the company, unlike the goals of production planning such as total production time or stock minimization.

First, a mixed-integer linear mathematical model (MILP) was created for the problem. The mathematical model, MILP was coded to solve with the GAMS

program. Data sets were created to test the performance of the MILP. Groups with small, medium, and large order data have been tried. With the MILP, only 10 order-sized data sets could be solved. A heuristic algorithm is used for medium and large data sets. Genetic algorithm, which is one of the most used heuristic algorithms, has been used in machine scheduling problems used in the literature. To compare the solution performance of medium and large data sets with the genetic algorithm and to see the effect of randomness on the solution of the problem, an algorithm using two different crossover methods named GADOC and GATPC has been developed. Solution performances of MILP, GADOC, and GATPC algorithms for only 10 order data sets can be compared, and the performances solving the problem of GADOC and GATPC algorithms for medium and large data sets were compared.

As a result, MILP solved by the GAMS program showed the best performance by solving the mathematical model in 10 data sets. The second-best performance is the GADOC program with a slight margin. The GATPC program has been the heuristic algorithm with the worst performance. However, the solution performances of GADOC and GATPC algorithms developed by the Genetic Algorithm have been compared, since the mathematical model can only solve up to 14 order sets. It has been observed that the GADOC algorithm performs better than the GATCP algorithm in large, medium, and small data sets.

1.1. Outline of the Thesis

In this presented study, the Fuzzy Order Acceptance and Scheduling on the Identical Parallel Machine problem (FOAS-IPM) is examined. In the first section, the definition of the problem, the production system type of problem (MTO), fuzzy logic, outsourcing, due date with customer satisfaction, and contribution of the study to the literature are presented. In the second section, the literature of the FOAS-IPM is explained. The third section of the study is about the definition of the stages of mathematical and heuristic model, and data set. In this section, the mathematical model and developed two Genetic Algorithms with the name GADOC and GATPC

are also explained. The fourth section is about the result of the developed methods for the FOAS-IPM problem. The remainder apart from these sections is the results and discussion, the conclusion, the references, and the other additional results related to this study.

1.2. Make to Order Systems

The main purpose of businesses ensures to use their resources carefully to achieve their main core aim that is the maximization of profit. The type of manufacturing system owned by enterprises is an important factor in the application of production system methods and rules. These systems aim to minimize costs under operating policies, deliver the desired quality product to the customer on time, and obtain the highest profit.

Manufacturing systems factors consist of "customer, input, process, and output". Customer demand is the first step in the systems. Raw materials, staff, and material are the general inputs. Factory, transport, service, tools, and machine are general examples of processes. Finally, goods or services are the final output of the systems.

Production is defined as "creating benefit" by economists, whereas it is defined by engineers as "making a change that will increase its value". The goal of the production systems is an appropriate amount of output, minimization of costs at labor, distribution, stock, or material, quality and product reliability, delivery on time, investments, the flexibility of product changes, and the ability to adapt to the changes in the amount of product quantity. (Çiftçi, 2020)

Manufacturing systems types can be categorized according to their management type, production type, production amount, or flow type. In general production systems are grouped as *continuous, intermittent, projects,* and *mix-type production systems*. (Akyurt, 2012) *Continuous flow-type production* and *mass production/assembly (repetitive) type production* are examples of continuous production systems. In these production types, there are usually less variety of

products and great demand. *A continuous production system* has evolved to produce less variety of products, thus a fast flow rate has been achieved. *An intermittent production* system has a wide range of products on the other hand the production amount is small. Machines with the same features are grouped in an intermittent production system. In Intermittent manufacturing systems, the goods are manufactured specially to fulfill orders made by customers rather than for stock. The flow of material is not continuous, it is intermittent. The manufacturing facilities are flexible enough to handle a wide variety of products and sizes. It is possible to produce different products by mounting some apparatus on the machines. Considerable storage between operations is required, so that individual operation can be carried out independently for further utilization of men's power and machines. The characteristics of intermittent production systems are producing small quantities, an unbalanced workload, and the necessity of highly skilled operators, large in-process inventory, and flexibility to suit production varieties. (Çiftçi, 2020) According to the process specifics, continuous production technology products are measured by weight and volume measures whereas intermitted systems products are measured by pieces or combined set of product flow. (Nenad Perši, 2008) Project type production, job type production, and batch type production can be the subgroup systems of the intermittent manufacturing systems.

In recent years in the sector while periods of order decrease, the number of orders increased, and delivery times shortened and the orders are more customized. So, more specialized products must be produced. Manufacturing firms must differentiate their systems more efficiently to achieve to produce unique products.

Make-to-order (MTO) production strategy is generally used in an intermittent manufacturing system. On the contrary, the make-to-stock (MTS) strategy is used mainly in continuous flow-type production systems. . In recent years in the sector while periods of order decrease, the number of orders increased, and delivery times shortened.

Manufacturing to stock is costly when the variety of products is large. Also, the risk is very high when demand is highly changeable or goods have short life cycles. Therefore, great growth in product variety goes with a shift from a Make-To-Stock (MTS) to a Make-To-Order (MTO) mode of production. In the MTO mode, production cannot start until a customer order is received. While this strategy eliminates finished goods inventories and reduces a firm's exposure to financial risk. (Diwakar Gupta, 2004)

MTO system is the realization and manufacturing of products according to customer's specific conditions. Customer requests can be about time, quality, design, or quantity. The firms working on the make-to-order operating principle have a necessity of a high level of decision-making practice and prevision because the incoming orders would have no standards on their features that change the production time and the cost parameters. Besides, the customers can have constraints of the delivery time of the orders as a further complicating issue for the decision-making process. In a profit-maximizing make-to-order firm, all of these constraints and aims must be carefully considered and handled for a feasible and beneficial decision. (Bilginturk, 2007)

In this study, a make-to-order manufacturing company production system is examined. Customized orders and initiation of production with customer orders are the main characters of this MTO system. The firms gain revenue if a production order is accepted and manufactured using the resources. In this system because of limited capacity received orders can't be produced until the due date or not. In case of a delay, the firm offers a discount to the customer. The longer the delay, the smaller the profit is. If the difference between income and discount is zero or lower, the product is not included in the schedule list.

1.3. Fuzzy Logic, Fuzzy Sets, and Fuzzy Numbers

The real world is complex to explain in mathematical expressions. This complexity is generally caused by uncertainty, lack of information, lack of precise

thinking, and decision-making difficulties. In the real world, non-exact and non-precise information sources are called fuzzy resources. In real-life systems, data cannot always be obtained completely and precisely due to measurement error, the complexity of the systems, lack of information, and flexibility of the language used in human nature. Classical mathematical expressions are insufficient in solving problems involving uncertain situations. Such uncertainties that classical logic could not define until then, mostly considered unscientific data. In the early 19th century, many philosophers put forward specific ideas on such uncertainties. Einstein expressed one of the famous quotes “As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality.” for expressing the real-world uncertainty with mathematics. (Göksu, 2015)

The fundamentals of the fuzzy logic theory were first defined in the first half of the 20th century. But the fuzzy logic theory was laid in computer science in 1965. (Zadeh, 1965) Fundamentals of fuzzy logic were founded in the 1920s by the Polish logician Jan Lukasiewicz. The propositions which are different from classical logic, are put forward "multivalued" logic principles that can take fractional truth values between one and zero. In an article published by Max Black in 1937, he applied "multivalued logic" to sets of lists or objects and drew the fuzzy cluster curves. Later, in 1965, Lotfi A. Zadeh published his groundbreaking article "Fuzzy Sets". In this article Zadeh explains a complete set of fuzzy sets applying Lukasiewicz's logic to all elements of a set. And also he developed algebra and made geometric descriptions of the fuzzy sets. (Zadeh, 1965)

Fuzzy logic was first used in a work by Ebrahim H. Mamdani. The fuzzy logic is used for the system of blurry for the steam engine with a control system that works with in the 1970s. From the work of Ebrahim H. Mamdani to the present day on, fuzzy logic, quite a lot of work has been done. Also since then, the term "fuzzy logic" has been used more express any mathematical or computer system with the help of fuzzy sets. (Kahraman, 2006)

Today, fuzzy systems are the systems closest to reality and the most used systems in the industry. Today, fuzzy systems, besides the applications such as process control systems in the industry, are used mainly in management science, decision making, air conditioning, medical industry, mechanical control systems such as automobile control systems, intelligent system designs, tissue identification, medical computing applications, robotics and motion control, used in civil, chemical and industrial engineering applications. Although the foundations of fuzzy logic were first laid in Europe, the center of use in this regard was Japan and Far Asian countries. One of the most impressive examples of these applications in Sendai city in Japan is the metro system. Since 1987, the fuzzy control system has been enabling trains to move rapidly on their routes, accelerate and brake smoothly, enter the stations precisely without wasting seconds and shaking passengers. Japanese companies like Matsushita, Nissan, Mitsubishi, and Sony have risen rapidly in their market and become giant companies with the products they put on the market using fuzzy methods. They produced washing machines evaluating the amount of laundry to be washed using fuzzy logic and adjusting the required amount of detergent, water temperature, and washing method. They also produced video cameras separating the shaking of the holding hand from the movement of the subject and automatically adjusts the lenses and produce clear images. TVs that automatically adjust contrast, brightness, color, and clarity using fuzzy logic were produced and dominated the market today. There are fuzzy logic applications in every technological development that improves our quality of life in daily life. Therefore, this situation further increases the commercial and academic interest in fuzzy logic. (Göksu, 2015)

In the industrial engineering field, fuzzy logic is one of the most used methods when defining the system. Fuzzy logic studies are categorized into several subtitles related to the industrial engineering field. Statistical decision-making is one of the important fields related to industrial engineering. Statistics is a science that includes the principles of collecting data for a specific purpose, summarizing with tables and graphs, interpreting the results, explaining the confidence levels of the

results, generalizing the results obtained from the samples for the population, investigating the relationship between properties, making predictions in various subjects, conducting experiments. It is based on the collection, classification, analysis, and interpretation of the results for a specific purpose. It can be applied in a wide range of fields from physics to natural sciences and social sciences. Provided with lots of experience, data mining techniques are widely used to extract suitable management skills from the data. For this reason, the most used method in studying data is fuzzy logic due to its proximity to real life in the statistic. Another area using fuzzy sets related to IE (Industrial Engineering) is the manufacturing system. In the manufacturing system, there are many interest areas, such as job shops, machine efficiency, line balancing, job shop design, etc. are researched under vagueness. Other studies using fuzzy sets related to IE are single criterion and multiple criteria decision-making studies. The other fields using fuzzy sets related to IE are information systems, fuzzy expert systems, artificial intelligence, engineering economics, ergonomics, production planning and control, aggregate planning, quality management, acceptance sampling, statistical process control, project scheduling, facility location, and layout, forecasting. (Kahraman, 2006).

1.3.1. Crisp Sets

A crisp set is defined as a well-defined object collection. A crisp set can be named as a classical set, or simply a set which are denoted by capital letters A, E, G, and the members of these sets are by a, e, g, c, etc. To demonstrate the membership an element of set A can be shown as$a \in A$. The negation of membership of an element can be shown as $a \notin A$ which means a is not a member of set A. if set A set have no elements is called an empty set and denoted by $\emptyset$.

The set A is a subset of B written as B $\supset$ A if every element of A is also a member of B. If $B = A,$ sets A and B to have the same elements. Therefore two sets

A and *B* are equal if and only if $B \supset A$ and $A \supset B$. The set of all subsets of a given set *A* is called the power set of *A* and is denoted by P (A). (Buckley, 2002)

The union *A* U *B* of sets *A* and *B* is defined to be the set of all elements which are members of A or B or both, or AUB = {x ∈ U lx ∈ A or x ∈ B}

The intersection A ∩ B of sets A and B is defined to be the set of all elements which are members of both A and B, in notation A ∩ B = {x ∈ U|x ∈ A or x ∈ B}

Commutativity : AU B = B U A, A∩ B = B∩A,
Associativity: AU (B U C) = (AU B) U C,
Associativity: A∩(B∩C) = (A∩B) ∩C,
Absorption: AU(A∩B) = A, A∩ (AUB)= A,
Distributivity : A∩(BUC) = (A∩B)U(A∩C),
Distributivity: AU (B∩C) = (AUB) ∩ (AUC),
Identity : AU¢ = A, AUX = X, A∩X = A, A∩ Ø=Ø (Buckley, 2002)

1.3.2. Fuzzy Sets And Fuzzy Numbers

1.3.2.1. Fuzzy Sets

Real-life systems are complicated and complex for modeling in a computer environment. When systems are modeled in the computer environment, they must be simplified. With simplification and certain assumptions, system modeling becomes easier. One way to model complex systems is by decreasing a certain amount of system imprecision, vagueness, and uncertainty. Of course, as a result of this, the results are not very close to the solution of the real-life problem and are not perfect, but most of the time the solutions become closer to realistic results. One example of the uncertainties is in human expressions. To explain uncertainty in life, the following example can be given: "Ali is old". This sentence does not exactly tell us how old Ali is. In this sentence "Ali is 50-55 years old", there is a "vulnerability" condition. There is uncertainty in betting games which is a subject of probability

In classical logic and fuzzy logic, the important is true propositions. But, in real life propositions are generally only partly true. In this sentence “Ali is young”, it is difficult to be sure about the age of Ali in this sentence. Ali could be 20 years old or Ali could be 28 years old. It is hard to characterize the truth of "Ali is young" as true or false if Ali is 25 years old. In fuzzy logic, it can be defined as t (Ali is young) to take on other values in the interval [0, 1] in addition to zero and one. The starting point of fuzzy logic is to allow truth values to be any number between 0 and 1. If p is a proposition in life, then t (p) demonstrates the truth of p. Therefore, t (p) ∈[0, 1] is for any proposition in fuzzy logic. If t (p) = 1, it means that p is true. If t (p) = 0, it means that p is false. If t (p) = 0.60 just means that the truth of p is 0.60. In fuzzy logic, the value of a proposition can be between 0 and 1. Let's define a fuzzy logic t (p), and negation is the same with t (∼p) = 1- t (p). (Buckley, 2002)

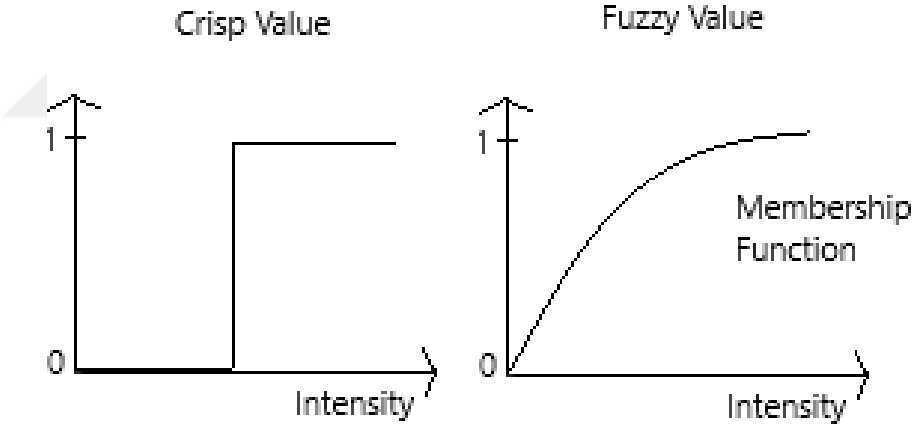


Figure 1.1. Membership Function of Crisp and Fuzzy Sets (Sleit, 2016)

E is a universal set, which contains all the elements of interest for our present application. Let A be a subset of E. The membership function of A is a function on E, with values zero or one, so that it equals one at x in X whenever x is in A and otherwise it equals zero. The membership function is as A(x). Then A(x) = 1 if x is a member of A and A(x) = 0 if x is not a member of set S.

The power set of A, written P (A), is the set of all subsets of S. The subset of S having no elements is called the empty set∅. Set A equals set B if A ⊇B and B ⊇A. The complement of set A^c, is defined as $A^c(x) = 1- A(x)$, for all x in A. The symbol ∩ is for intersection and the symbol U is for union Let set C is the intersection of set A and set B, written C = A∩B, if

$$C(x)\begin{cases} 1 & if\ A(x) = B(x) = 1 \\ 0 & Otherwise \end{cases}$$

D is the union of A and B, D = A U B, if

$$D(x)\begin{cases} 1 & if\ A(x) = 1\ or\ B(x) = 1 \\ 0 & Otherwise \end{cases}$$

Commutativity: AU B = B U A, A∩ B = B∩A,

Associativity: AU (B U C) = (AU B) U C,

Associativity: A∩(B∩C) = (A∩B) ∩C,

Distributivity : A∩ (BUC) = (A∩B)U(A∩C),

Distributivity: AU (B∩C) = (AUB) ∩ (AUC),

Identity : AU∅ = A, AUX = X, A∩X = A, A∩ ∅=∅

Absorption : AU (A∩ B) = A,A∩ (AUB) =A (Buckley, 2002)

If there is a relation between A and B are as ∀x ∈ E and $\mu_A(x) \leq \mu_B(x)$, the fuzzy set A is a subset of the fuzzy set B which is notated as B ⊇A. If set A is a subset of set B and ∀x ∈ E and $\mu_A(x) \neq \mu_B(x)$, A is a proper subset of B and notate as B ⊃ A. If the membership function value of members of two set A and B take the same value, these two set A and B are equal set and notate as ∀x ∈ E, A=B, and $\mu_A(x) = \mu_B(x)$. Vice versa ∀x ∈ E and $\mu_A(x) \neq \mu_B(x)$, A≠B when two set doesn't equal to each other. Complement of a set can be notate as ∀x ∈ E and $\mu_A(x) = 1- \mu_B(x)$,

$B=A^c$ or $A=B^c$. ($A^c=\overline{A}$ and $B^{c=}\overline{B}$). For universal complement set of A is $\forall x \in E$ and $\mu_{A^c}(x) = 1- \mu_A(x)$. (Tuş, 2006)

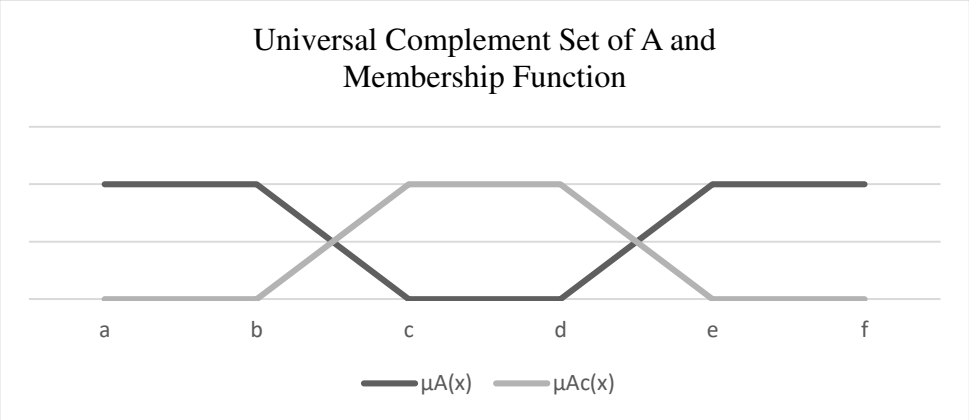


Figure 1.2. A complement set of a fuzzy set(Tuş, 2006)

The union of two fuzzy sets is AUB= $\int_x(\mu A(x) \vee \mu B(x))/x$ and $E \supset A, B$ $\forall x \in E$ and $\mu_{AUB}(x) =\max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x)$

Intersection of two fuzzy set is A∩B AUB= $\int_x \left(\mu A(x) \wedge \left(\mu B(x)\right)\right)/x$ and $E \supset A, B$ $\forall x \in E$ and $\mu_{AnB}(x) =\min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x)$

Multiplication of two fuzzy sets is $\forall x \in E$ and $\mu_{A.B}(x) =\mu_A(x).\mu_B(x)$

Sum of two fuzzy sets A and B is $\forall x \in E$ and $\mu_{A+B}(x) =\mu_A(x) +\mu_B(x) - \mu_A(x).\mu_B(x)$

Power of a set is A^a and a is a positive number, then $A^a= (\mu_A(x))^a$.

"a" is a positive number, times of a fuzzy set A is $\mu_{Aa}(x)=a\mu_A(x)$.

The membership function of subtraction of a two fuzzy set A-B, $\forall x \in E$ and $\mu_{AnB^c}(x) =\min(\mu_A(x), \mu_{B^c}(x))$ (Tuş, 2006)

1.3.2.2. Fuzzy Numbers

Fuzzy numbers are very important in fuzzy systems. Fuzzy numbers are a special fuzzy subset of numbers in a real number set. Closed interval and real

numbers are special fuzzy numbers. In studies generally, a common type of fuzzy numbers are the triangular (shaped) and the trapezoidal (shaped) fuzzy numbers. The extension principle and the a-cut with interval arithmetic methods are used in the arithmetic of fuzzy numbers. Finding the distance between two fuzzy numbers, finding the maximum and minimum of fuzzy numbers, ordering fuzzy numbers are the most used properties in fuzzy numbers. Fuzzification and defuzzification are the common methods in the application of fuzzy numbers and equations while precise data is converted into imprecise data or vice versa.

Let's $\bar{G}$ is a fuzzy subset of reel numbers set **R**,

1-The core of $\bar{G}$ is non-empty

2- α-cut of $\bar{G}$ are closed, bounded, intervals

3- The support of $\bar{G}$ is bounded

The core of $\bar{G}$ is of $\bar{G}[1]$, the α=1 cut. So, $\bar{G}$ must be a normal fuzzy set. The α-cut of$\bar{G}$,$\bar{G}[\alpha]$, are $[g_1(\alpha), n_2(\alpha)]$ and $0 \leq \alpha \leq 1$. $\bar{G}[\alpha]$ is a closed, bounded, the interval for all $0 \leq \alpha \leq 1$ means that $\bar{G}[\alpha]=[g_1(\alpha), g_2(\alpha)]$ where $g_1(\alpha)$ is the left and $g_2(\alpha)$ is the right endpoint of this interval.

A triangular fuzzy number (TFN) $\overline{G}$ is defined by three numbers a<b<c where the vertex of the triangle is at x=b, its base is the interval [a,c], and also $\bar{G}$=[a/b/c] for triangular fuzzy numbers.

A trapezoid fuzzy numbers $\overline{D}$defined by the numbers a<b<c<d and $\overline{D}(x)=1$ on [b,c] and base is [a,d]. D = (a/b,c/d) for a trapezoidal fuzzy number.

The triangular fuzzy number is one of the most used types in applications.

$$\mu_A(x)\begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a < x < b \\ \frac{c-x}{c-b} & b < x < c \\ 0 & c < x \end{cases}$$

Properties of arithmetic in triangular fuzzy numbers (TFN) are:

1- The result of the sum or subtraction of two TFN is TFN.
2- The result of the multiplication or division of two TFN might not be TFN.
3- The result of the maximum or minimum of TFN might not be TFN. (Buckley, 2002)

Fuzzy numbers are very difficult to understand by computer or machine-based systems, so a defuzzification process is required for expressing fuzzy numbers of systems as real numbers. As we mentioned in the previous section in set definitions, fuzzy numbers are subsets of real numbers. (Buckley, 2002)

To explain fuzzy numbers to a machine or a model in a computer environment, it is necessary to convert fuzzy numbers to crisp numbers. There are methods used in this conversion process. Some examples of defuzzification methods are the α-cut method, the weighted average method, the maxima method, and the centroid method. (Buckley, 2002)

Since the fuzzy numbers in this study are triangular fuzzy numbers, the ranking methods of triangular fuzzy numbers will be explained in this section. Numerous fuzzy number ranking methods have been found in the literature until today. The first study to fundamental ranking triangular fuzzy numbers was done by Jain R. 1976 and a bit later in 1980 by Yager PR. In Yager's study, the calculation is made with the abscissa of the center of gravity of the triangle formed by fuzzy numbers. (Emrah Akyar, 2012) In 1981 his another study for ranking triangular fuzzy numbers is a weighted mean value or in other words centroid calculation method for triangular fuzzy numbers. (Yao & Wu, 2000) Since then, many ranking methods like the distance index based on the two coordinates of the centroid have been developed for ranking triangular fuzzy numbers. (Emrah Akyar, 2012)

In 2000, a new method was developed by Yao for ordering triangular fuzzy numbers. The signed distance method is used in ordering the triangle fuzzy numbers, just as it is applied in real numbers. In that study, the properties of signed distance in triangular fuzzy numbers are close to the property of signed distance in real numbers. For fuzzy triangle numbers, the signed distance value is calculated. Calculation with this formula is as follows. Let F= (u_1, u_2, u_3) is a triangular fuzzy number. (Yao & Wu, 2000)

$$d(F,0) = \frac{1}{2} \times \int_0^1 [u_1 + (u_2 - u_1)\alpha + u_3 - (u_3 - u_2)\alpha]d\alpha = \frac{1}{4}(2u_2 + u_1 + u_3)$$

1.4. Outsourcing

Businesses have to make some innovations to continue their activities due to global competition. Companies have to renew and review their business processes and operating policies to make themselves known in the global market and gain profit share from the market**.** One of these renewal methods is outsourcing. For businesses to continue their activities more effectively, they have the opportunity to transfer their business to companies that are specialized in this field**.**

Outsourcing reasons are respectively, reducing costs, staff utilization flexibility, quality by increasing, reducing risk, keeping pace with global innovations, increasing competitiveness, increasing capacity or creating additional capacity, and increasing performance.

There is no exact date when outsourcing, which is extremely important for businesses, began to be implemented in businesses. It is seen that the first practices date back to the Roman period. In the Roman period, it is seen that external resources were used to make tax collection work more efficiently and systematically.

With the industrial revolution, outsourcing has become widespread and used in many production areas. Outsourcing the metal products used in firearms in the 18th and 19th centuries can be shown as an example of outsourcing. Besides, the

maintenance and operation of street lamps in the UK, many activities such as management, collection of taxes, collection of waste materials, and road maintenance are left by the state to private companies. Again in the 19th century, the transfer of postal services to private companies in the USA and Australia is an example of outsourcing. The transfer of the construction, maintenance, management of railways, and the management of water tanks to private companies in France has been an example of outsourcing between the state and the private sector. (Erer, 2018)

At the end of the 20th century, manufacturing companies in America were supplying their product parts from Fareast countries as a part of these American companies. At first, there was a company partnership. But later, these suppliers worked as independent companies for American companies as suppliers.

American companies have a large number of international suppliers in various regions all over the world. But unlike US international companies, European and Japanese international companies have increased their production capacity by making heavy investments in their own companies even if they have enough outsourcing investment sources. They improved the quantity and shape of their core product. Conversely, American companies rely more on new product development as another way to maintain a competitive advantage rather than a well-coordinated manufacturing strategy.

American companies started to purchase finished products from countries such as China and South Korea, from Far East countries where they would cost production cheaper to exist in the global market where competition is very high. Later, world giants Far East companies like Sony in Japan have adapted to this competitive idea.

It is an indisputable fact that international trade is now quite advantageous in using outsourcing. Production costs are also very low, especially in countries where labor costs, specialization, resources, and taxes are relatively lower. The best example that can be given to this country is China today. Many international

companies, nowadays have their products manufactured in China and export from there due to their production costs.

International services have a significant positive impact on outsourcing efficiency, but no productivity gains for exporters without domestic and foreign capital exporters. Although outsourcing is generally defined as the export of other activities that are not the core of the company, companies that want to increase the reliability of the delivery date are now outsourcing their activities, too.

Companies that want to increase customer loyalty and do not want to lose their customers in the market have started to give more importance to customer relations than before. On-time delivery is important, but not so easy to achieve. For this reason, companies that want to deliver on time have turned to use outsourcing. In some studies, it has been shown that disabling secondary activities that do not directly contribute to performance will be the key to the success of businesses.

Indeed, due to globalization and competition, companies need to focus on their point of activity within the company to respond to demands faster. It is necessary to be able to respond to the demands and expectations of the market in a timely and complete manner, to offer products that comply with the standards, and to be able to respond quickly to demand to become a brand. Outsourcing is a very suitable technique to achieve this.

Most of the manufacturing organization encounters the situation of surplus demand than the capacity to produce. The firms have to supply all the market demand to prevent other major competitors from penetrating their strong market area also to continue their reputations for on-time delivery. (Chakraborty, 2006) For example, due to lack of raw material or delays in raw materials of the company may cause damage to the company in terms of late delivery of the order, idle machines, an increase in unfinished material stock, and other issues. Due to capacity constraints, failure to produce all orders received and not deliver them to the customer on the promised dates will cause problems. The delivery date, which is one of the most critical points in customer satisfaction, will not be realized. This situation will

adversely affect the relationship with the customer and will reduce customer loyalty and even lead to losing the customer.

In the case of capacity problems or delays in the process, outsourcing of the company can solve the problems and provides an additional opportunity for the company to manage the process better. Another option for the company is not to lose the customer by bearing a little more cost instead of delivering the product to the customer late and moving the relationship with the customer to a bad place. It has also been shown in some studies that the inclusion of outsourcing in its production is more profitable than the production of the company with only its resources. (Chakraborty, 2006)

In many respects, outsourcing is at a very advantageous point for companies. It offers flexibility to the company in increasing customer satisfaction, possible delays, and possible extra penalties. For this reason, the company discussed in this study is outsourcing the products not to lose the customer or to keep good relations with the customer, in a way that does not make any profit or loss from the product it accepts.

1.5. Customer Satisfaction and Delivery On Time

Customer satisfaction is one of the most important key factors of any business in the world. The expression ‘customer satisfaction means the level of happiness of the customer about the quality of goods and services. Another a bit different explanation of customer satisfaction is “the feeling of satisfaction as a result of the comparison between perceptions of a product’s performance and expectations. “Another definition of customer satisfaction is customer’s fulfillment response which depends on the features of the product or service. The customer satisfaction level depends on the level of good or bad experience and happiness that the product or service provides for the customers. It has been shown that customer satisfaction is very important for the profitability of the company. Also, it has been shown that the cost of finding a new customer is more expensive than keeping a satisfied

customer. Also, customer loyalty is high with a satisfied customer. A satisfied customer generally makes fewer complaints or defends the company in a dissatisfying situation. Satisfied customers always advertise positively about the company. Sometimes they even become covert defenders and protectors of the firm in the market. Especially, a positive experience leads to a better reputation, more profit, and more loyal customers. Also, loyal customers spread positive experiences and information about the company. This is one of the most reliable advertisements for the company.

Satisfied customers generally don't change unless a very dissatisfying situation happens about the firm which destroys the trust between the customer and the company. But In today's world where there are many options offered to customers by companies due to high competition in the market, it is not easy to ensure continuous customer satisfaction and long-term customer loyalty. Today, one of the most used tools that bring companies and customers together as a result of high technology development is undoubtedly the internet. Especially with the incredible development of technology, customer can reach their need easily with online shopping on the internet. As a result of this, studies related to customer relations are shifting towards a field that is more related to online shopping. Also on online shopping, the customers' behaviors can be easily analyzed. The time they spent on a brand, the money they spent, or the other information about a customer can be collected easily. So, with lots of information, the researches on online shopping can be easily analyzed. Today, especially with the incredible development of technology, customer can reach their need easily with online shopping on the internet. As a result of this, studies related to customer relations are shifting towards a field that is more related to online shopping because of collecting information and analyzing easily. Researches on online shopping showed that spending on the brand with which a customer is satisfied shows that customers spend 2.6 times more than somewhat satisfied customers and 17 times more than somehow dissatisfied customers. This study is proof that all customers are in this direction. It is very

important to build loyalty in customer relations. It is also shown that companies' profitability is in direct proportion with customer satisfaction. (Cihan, 2017)

Satisfied customers generally don't change unless a very dissatisfying situation happens about the firm which destroys the trust between the customer and the company. But In today's world where there are many options offered to customers by companies due to high competition in the market, it is not easy to ensure continuous customer satisfaction and long-term customer loyalty.

Product variety, low price, reliability, and delivery performance are the key factors in customer satisfaction. The reliability of the product depends on the quality and the quantity of the product. The product variety is another important key for customers. Low prices and saving money are important in customer satisfaction. The promised delivery time of the products must not exceed the duration promised by the company furthermore, delivery time should be short. Any delayed delivery may harm customer satisfaction. (Cihan, 2017).

The importance of the delay in the delivery time varies depending on the sector and the customer base it addresses. For example, for the automotive industry, the delivery rate after production is recalculated with customer surveys after each shipment. It is a target parameter that is calculated and sent by each customer. On-time delivery is important, but not so easy.

There are multiple reasons which prevent the company from delivering on time. One of the reasons is that the order placed by the customer is delivered late and the process is delivered late. Lack of raw materials, delays in raw material delivery, obligation to reproduce due to faulty production, unexpected machine breakdowns, lack of follow-up in the production process, delay in customer payment, an early deadline, delay due to outsourcing, lack or insufficiency of personnel, change in customer demand, dynamic changes during production, order production sequence changing due to importance among customers, the dynamically changes of production, and incorrect delivery date given to the customer.

1.6. Contributions to the Literature of Fuzzy Order Acceptance and Scheduling in Identical Parallel Machine in Make to Order Systems

Order Acceptance and Scheduling is one of the new topics that has attracted much attention in the literature of manufacturing systems and there has been a limited amount of studies in the literature about OAS.

In OAS literature, the problems and solutions are mainly on single-machine systems. However, real-life production systems are mostly multi-machine systems. Therefore, multi-machine systems have been studied in this thesis. Firms with an MTO strategy generally group machines with the same feature. So, as in MTO manufacturing firms, in this study, a system with identical parallel machines has been examined.

Fuzzy Order Acceptance and Scheduling in Identical Parallel Machine (FOAS-IPM) is the more specialized sub-problem of the OAS problem. In real life, the systems and production processes are not certain. Especially the factors that depend on time are fuzzy. Production times, due dates, completion times, and delay times are generally not certain in real-life systems. In MTO systems, the number, and variety of orders are large. So the parameter features of the production system are not certain like the continuous manufacturing systems.

The OAS problem has been studied in the literature for more than 20 years. While stochastic system studies were more common in earlier studies, more recent studies focused on more deterministic system studies. (Slotnick, 2011) Therefore, a deterministic modeled system is considered in this study. The study of Fuzzy Order Acceptance and Scheduling in Identical Parallel Machines (FOAS-IPM) aimed to solve the problem of order acceptance and scheduling in parallel machines with fuzzy system parameters.

Uncertainty analysis and modeling have been studied by researchers for years. Various approaches have been developed in this regard. Fuzzy logic is a method used in solving uncertainty problems. In this study, a fuzzy mathematical

model has been developed to handle uncertain production times, due dates, completion times, and delay times.

It is necessary to create a system close to the real system in which the solution is tried to be found, and the system needs to be expressed with a mathematical model. In this presented study, first, a mathematical formulation of the system has been examined. This formulation has been developed to solve the FOAS-IPM problem to have maximum profit by using the properties of the orders. The completion time of an order depends on the sequence of orders on a machine. If the order is the first order to be processed on a machine, its completion time depends on setup time and production time. But if an order is not the first order on a machine, then its completion time depends on its sequence set up time, production time, and the completion time of the previous order. So, the sequence of orders is very crucial. The completion time and the due date relation is the main key to give the acceptance to production schedule or outsourcing decision. If the completion time of an order is not further than its due date, the order includes an accepted order list and its net profit is its revenue. If the completion time of the order, further than its due date, the delayed punishment calculates. If the delayed punishment is less than or equal to its revenue, the order includes an accepted order list. Otherwise, the order includes in the outsourcing list and occurs an outsourcing penalty cost. The order Acceptance and Scheduling (OAS) problem hasn't been studied with outsourcing penalty cost in maximization of net profit in identical parallel machine systems with fuzzy durations in literature since now. So this is the first study that includes outsourcing penalty cost in the OAS problem.

FOAS-IPM problem has fuzzy properties. The parameters, production times, sequence-dependent setup times, and due dates are triangular fuzzy numbers. As the product's process times change in each measurement in real life and the production time is uncertain in the production process in general, these factors become fuzzy. In this system, orders and their specifications are ready at the beginning of the planning process. Orders revenue, fuzzy production times, fuzzy due dates, tardiness

penalty cost, and outsourcing cost are known. When the delay occurs, the tardiness penalty per each delay time decreases the revenue of the order because the firm must deliver the products on time but if not, the firms must pay the delay discount to the customer. If the discount is obtained as a result of the production delay time and it is equal to the revenue, the product cannot be included in the scheduling list since the product can no longer bring profit to the firm. When the order doesn't include in the scheduling list, an outsourcing fee occurs.

Outsourcing is generally used with the minimization of makespan or the number of rejected orders in the literature. (Zhong et al., 2014) Outsourcing is generally used with the minimization of the number of rejected orders in the literature (Shabtay et al., 2012) In this study if an outsourcing fee occurs, this parameter decreases the profit in a maximization problem. In this study, outsourcing is used as a parameter to decrease profit in the maximization problem. In this study, this rejecting occurs due to outsourcing. Outsourcing cost and delay cost decrease the total net revenue if both or one of them occur. The outsourcing parameter is generally used in the objective function of minimization of the rejected order number in the literature. In this study, outsourcing is used as a parameter to decrease profit in the maximization problem if outsourcing occurs.

The outsourcing penalty occurs due to the pricing policies between companies. So if an order cannot be included in production scheduling, this order must be produced in or purchased from another company that has different product prices. In this study, the firms bear this extra outsourcing cost at least to be able to deliver the order on time which leads not to lose customer loyalty and keep good relations with the customer. Customer loyalty is one of the key points to increase profit. Studies show that to seek price a new customer in a market is higher than keeping the relation with an old customer. (Cihan, 2017)

2. LITERATURE REVIEW

In this section, the literature review of previous research on Order Acceptance and Scheduling is presented. In this chapter, the studies have investigated the problems that are related to the OAS problem and give the solution methods proposed for these if applicable. OAS problem has been studied for more than 30 years in literature. In 2011 a great taxonomy has been done by Slotnick. (Slotnick, 2011). In this taxonomy, major themes in OAS research are presented as net present value objective, lead time or due date, joint decision making, and value of the information. In the related paper, the literature on OAS has been categorized into 4 parts. Single machine and multiple machines in the stochastic system and the deterministic system. In this study, the articles in the literature have been contributed after this taxonomy has been added to this thesis literature review part. The papers discussed in this literature survey can be categorized into two subtitles as single and multiple machine studies on order acceptance and scheduling.

2.1. Order Acceptance and Scheduling Studies with Single Machines

OAS studies in this chapter addressed the scheduling problem in a single-machine environment. The studies take their place under this title according to their publishing year.

In 2009, Rom and Slotnick studied order acceptance and scheduling. This paper developed a genetic algorithm to solve the order-acceptance problem with tardiness penalties. This study models the order-acceptance decision with static arrivals, deterministic processing times, and customer weight and revenue for each job in a single machine environment. In this paper, there are two main decisions. The first decision is which jobs to accept, and the second decision is in what order to process the selected subset. The objective is profit maximization which the sum of per-job revenues minus total weighted tardiness. The objective function is maximizing the total revenue. (Rom & Slotnick, 2009)

In 2010, Oguz et al. studied order acceptance and scheduling decisions in make-to-order systems. In their paper, the OAS problem has been examined with the orders defined by their release dates, due dates, deadlines, processing times, sequence-dependent setup times, and revenues in a single machine environment. The objective of the study is to maximize total revenue. The revenue is a function of tardiness and deadline. They developed a MILP OAS formulation that can be solved to optimality up to 15 orders. For large-sized problems, three heuristic algorithms are developed to solve the problem. Computational tests showed that the algorithms are computationally efficient and effective for instances of up to 300 orders. (Oguz, 2010)

In 2012, Cesaret et al studied order acceptance and scheduling decisions problems with a heuristic algorithm. They developed a tabu search algorithm to solve the order acceptance and scheduling problem on a single machine with release dates and sequence-dependent setup times. In their study, they compare and analyze the performance of the tabu search algorithm on a large set of test instances with up to 100 orders and compare the results with two heuristics from the literature. (Cesaret, 2012)

In 2014, another paper on order acceptance and scheduling with a single-machine model has constraints about machine availability. The system has restrictions on time availability to process orders and the model allows the manufacturer to reject or outsource some of the orders. When an order isn't accepted, due to rejecting or outsourcing, each order has a cost of penalty which decreases the net revenue. The objective of this study is the minimization of accepted orders makespan and the number of rejected orders. (Zhong et al., 2014)

In 2016, order acceptance and scheduling with earliness and tardiness penalties were studied in a single machine system. The objective function includes setup costs, job rejection penalties, weighted tardiness penalties (for the jobs completed after their due date), and weighted earliness penalties (for the jobs starting before their initially planned release dates) and the objective is the minimization of

total penalties. Linear and quadratic earliness and tardiness penalties are studied. The earliness penalties model is expressed as the price of a delivery of raw material and the release dates of the jobs. In this study, job preemption is not allowed, whereas idle times are allowed. Taking into account a single-machine system, this study is the first to simultaneously consider rejection and earliness/tardiness penalties with idle time. (Thevenin et al., 2016)

In 2017, a simultaneous order acceptance and scheduling problem was studied with fuzzy parameters in a single machine system. At the beginning of the planning horizon, a firm receives a set of customer orders. Each order has some specifications like revenue value, fuzzy processing times, fuzzy due date, and fuzzy set up times. The signed distance method is used to handle the fuzzy parameters in the model. Mixed-integer linear programming is developed to solve the problem with fuzzy parameters. Due to Np-hardness, only small-size instances can be solved by MILP in the study. The objective of this study is the maximization of revenue which calculates the differences between revenue and tardiness penalties (Koyuncu, 2017)

In 2017, a mathematical programming model, and two population-based metaheuristic algorithms, biogeography-based optimization (BBO) algorithm, and genetic algorithm (GA) were proposed for solving order acceptance and scheduling problems. The system in the study considers a single machine environment. The capacity of accepted orders is limited and the orders are on due dates, processing times, revenues, weights, and sequence-dependent setup times. The objective is total net revenue maximization which is calculated as total revenues minus the total weighted tardiness. This study showed that the BBO algorithm outperforms GA, especially at large-size instances. (Zandieh & Roumani, 2017)

In 2019, an artificial bee colony-based hyper-heuristic for the single machine order acceptance and scheduling problem was presented. In this study, order acceptance and scheduling problem which is more complex due to a typical *NP*-hard includes set up times between the orders. Solving an *NP*-hard problem through exact approaches is computationally difficult or sometimes impossible and the result

becomes failing to handle large-size instances. So, they proposed a hyper-heuristic in which an artificial bee colony (ABC) algorithm is employed as a search methodology for the OAS problem. As a result, the algorithm performs better and the results show that the integration of the ABC algorithm into hyper-heuristic outperformed the other approaches in terms of average and minimum deviation from the upper bound. (Chaurasia & Kim, 2019)

In 2019, the order acceptance and scheduling problem was handled in a more environmentally friendly way. One of the first studies in the literature considers sustainable scheduling. In this paper, the OAS with carbon emission reduction and electricity tariffs on a single machine was presented. The objective function is to maximize the total revenue of the accepted after subtracting the carbon emission cost and the electricity cost under different time intervals on a single machine. This study considers sequence-dependent setup times and release dates. A mixed-integer linear programming model (MILP) and a relaxation method of the MILP model were proposed to solve this problem. The study shows an apparent trade-off between green scheduling and electricity cost. (Chen et al., 2019)

In 2019, another order acceptance and scheduling with maximizing the total net revenue was presented. They propose a new mimetic algorithm called Sparrow that was developed to solve the problem. It is a union of hybridization of biased random key genetic algorithm (BRKGA) an adaptive large neighborhood search (ALNS). Sparrow integrates the exploration ability of BRKGA and the exploitation ability of ALNS. After being proposed the order acceptance and scheduling problem with sequence-dependent setup times with mixed-integer linear programming (MILP) method for this problem in 2010 then two simple greedy constructive heuristics and simulated annealing based heuristic algorithm called iterative sequence first-accept next (ISFAN) were developed. After these methods, for the same problem, tabu search (TS) algorithm, an exact branch-and-bound (B&B) with genetic programming (GP), a 100 iterated local search (ILS) method, a hybridization of population-based methods with local search (LS), an artificial bee colony (ABC)

algorithm, a diversity controlling genetic algorithm (DCGA), a dispatching-rule-based genetic algorithm (DRGA), a GP method with stochastic dispatching rules, a hybrid particle swarm optimization (PSO) method with TS using dispatching rules learned by GP, a hyper-heuristic algorithm, which is a learning and optimizing system (LOS), a hybrid steady-state genetic algorithm (HSSGA), an evolutionary algorithm with guided mutation (EA/G-LS), and finally an ABC-based hyper-heuristic was developed to solve the same problem. The heuristic called Sparrow was compared with these algorithms. The study showed that Sparrow performs better than these algorithms. (He et al., 2019)

In 2019, a genetic algorithm was proposed for fuzzy order acceptance and scheduling in a single machine system. After developing mix integer linear programming, a genetic algorithm was developed for the fuzzy OAS problem. In this study, six instances with different numbers of orders are examined and. Two different ranking methods are compared. The calculations showed that the signed distance method and the integral value method with alpha value 0.5 give similar results. It is shown that diversity-based selection and local search improves the solution, especially when the size of the problem increases. (Karakaş & Özpalamutçu, 2019)

In 2019, two mixed-integer programmings (MIP) models are formulated for order acceptance and scheduling. Branch-and-bound based algorithm is developed for solving complicated instances with the principle of “divide and conquer”. Extensive computational experiments on various instances are done and the results show the efficiency of the enhancement techniques for the formulations, also the effectiveness and efficiency of the formulation-based branch-and-bound algorithm. The objective of the study in both the MILP model was the maximization of total net revenue. The first MIP model is based on the concept of a dummy job and the second model is based on the linear ordering variables. In this paper, the developed branch-and-bound algorithm can handle instances with up to 50 orders approximately in half an hour. Seven methods, MIP1, MIP2, MIP1V, MIP2V, the B&B algorithm with *b*

= 1, 5, 10 denoted as “BB1”, “BB5”, and “BB10”, respectively are compared with test instances of the problem. The results of the paper showed that the enhanced formulations perform much better compared to basic ones according to the average computation time. (Wang & Ye, 2018)

In 2019, the order acceptance with two-stage assembly scheduling was presented. The objective of the study is to maximize the profit which is calculated as the sum of revenues of the accepted orders minus total weighted tardiness. First, a mixed-integer linear programming model is formulated and then a non-semi and semi permutation genetic algorithm (SPGA) is developed to solve the problem. The solutions of SPGA are compared with those of CPLEX and non-semi permutation genetic algorithm (N-SPGA). Until the medium size of instances, CPLEX can find an optimal solution but for larger instances, SPGA and N-SPGA can be used. SPGA is nearly fast as CPLEX in finding an optimal solution in a small size of instances. But for large size of instances, SPGA performs far better than N-SPGA. For analyzing the results ANOVA is used in the study. (Yavari et al., 2019)

In 2020, an order acceptance and scheduling problem was studied with earliness and tardiness penalties. The objective of this study is the maximization of total net revenue. The total net revenue depends on revenue and the penalties due to tardiness and earliness of production time. Each order has specific revenue, processing time, release date, due date, deadline, and earliness and tardiness penalties. The examined system is a single-machine system that also calculates the idle times between orders. A MILP model is formulated for the problem. Then, two exact algorithms are developed to solve the problem. The first algorithm, denoted by DPIA-GR, is a dynamic programming-based algorithm. The second algorithm DPIA-LRGR developed from DPIA-GR by incorporating Lagrangian relaxation (LR). The computational result shows that both DPIA-GR and DPIA-LRGR solve the problem efficiently and outperform CPLEX and GA. When DPIA-GR compares with DPIA-LRGR, DPIA-LRGR performs better. (Li et al., 2020)

2.2. Order Acceptance and Scheduling Studies with Multiple Machines

In 2012, a survey about order acceptance and scheduling with rejection is with many of OAS problem studies in this survey. In contrast, in this study, a survey has been done about the objective is minimizing completion time and rejecting order cost. The especially bi-objective function is discussed in the paper. The system in this study is deterministic and the orders are known in the beginning horizon. The system is a multi-machine system and also single machine systems are studied. (Shabtay et al., 2012)

In 2013, order acceptance and machine scheduling on two identical machine system was presented. Customer orders are ready at the beginning of planning. Each order has a specific revenue value, processing time, due date, and tardiness weight. First, one exact algorithm was developed to solve the small-sized problem then two heuristic methods have been done to solve the large-size problem due to the inefficacy of the exact algorithm. (Wang et al., 2013)

In 2015, simultaneous order acceptance and machine scheduling problem in a non-identical six machines system was done. Randomly generated up to 40 instances are tried to be solved by a heuristic and by a monolithic mix integer linear programming. In this study, orders have their due date, revenue, tardiness penalty, different processing time on the machines, and sequence-dependent set-up times. The objective is the maximization of revenue. Lagrangian relaxation algorithm is developed to solve the same problem sets with monolithic MILP. As a result, in the study, the Lagrangian relaxation algorithm performs better results than monolithic MILP. (Emami et al., 2015)

In 2015, order rejection and scheduling on a system with two identical parallel machines were studied. The objective is the minimization of makespan and the total penalty of the rejected jobs. First, an integer program that is relaxed into linear programming is formulated. Then, two heuristics are formulized to solve the problem. A deterministic 3-approximation algorithm and a randomized 3-approximation algorithm are used to solve this problem. (Feng Lin, 2015)

In 2015, order acceptance and scheduling on two identical parallel machines are studied. Orders are ready at the beginning and each has a revenue value, processing time, due date, and tardiness weight. The objective of this study is to maximize the total revenue from orders. In the system, there are two identical machines to schedule the accepted orders. Because of being intractable, they develop two heuristics and an exact algorithm based on some optimal properties, and also the Lagrangian relaxation technique is used. They compare the results with computational experiments. The result of the study is presented that the heuristics are approximately efficient when they solve the large-sized instances of the problem, whereas the exact algorithm can only handle small-sized instances. (Wang et al., 2015)

In 2015, an order acceptance and scheduling problem in a flow shop was presented. In this study, mixed-integer linear programming and an effective parallel neighborhood search algorithm are formulated to handle this problem. There are two objectives to this study. The first objective is minimizing the makespan and the second one is maximizing the total net revenue. In this paper, an effective parallel neighborhood search (PNS) is applied to simultaneously minimize makespan and maximize total net revenue. Two-string representation and three neighborhood formulations are examined to generate new solutions. PNS is compared with other algorithms, namely, VNS and TS on the same problem instances. Computational results presented that the PNS method can provide better results than VNS and TS in most instances. (Lei & Guo, 2015)

In 2016, order scheduling with rejection and no simultaneous machine available time on identical parallel machines is presented. In this model, the orders are ready at the beginning of scheduling, whereas the machines are not available at the beginning. If an order isn't accepted, a rejection cost occurs. The objective is to minimize the makespan and the total cost of rejecting orders. Because of its *NP*-hard property, they first studied a polynomial-time algorithm to obtain a lower bound of the optimal objective value of the problem and then examined a heuristic by using

some relaxation techniques. In the study, it is showed that its worst-case ratio bound is 2. (Jiang & Tan, 2016)

In 2018, The OAS problem without the due date on an identical parallel machine environment with weighted completion time is worked out with a mixed-integer linear programming model to maximize the revenue minus the scheduling cost and an exact B&B algorithm. An order acceptance and scheduling (OAS) problem in an identical parallel machine environment was presented. In this study, each order has unique revenues, processing times, and weights. A mixed-integer linear programming (MILP) model is proposed for the problem and the objective function is maximizing the revenue apart from the completing cost. The problem is NP-hard so a branch and bound (B&B) algorithm is developed to solve instances of the problem. After that, the extended B&B algorithm is proposed to solve the larger size of the problem instances to obtain ϵ-optimal solutions. (Palakiti, 2018)

In 2018, an order acceptance and scheduling problem with identical parallel machines was presented. In this study, the processing times of orders are machine-dependent which are identical parallel and have eligibility constraints. (OAS-ME) This implementation of the OAS-ME problem considers two objectives which are to maximize the total net profit concerning the revenue, tardiness cost, and production cost and to minimize the makespan, which is a classic scheduling criterion concerning productivity

Both objectives are conflicting. To solve the problem, the mathematical model of this problem is formulated as multiobjective mixed-integer linear programming. First, a classic list scheduling (LS) rule named the first available machine rule is applied, and then three new LS rules are examined, which are the maximization of the net profit, the minimization of the makespan, and the trade-off between the two objectives, respectively. A list-scheduling-based multiobjective parthenogenesis algorithm (LS-MPGA) is studied to solve the problem with parthenogenesis operators and Pareto-ranking and selection method. Results showed

that the performance of the LS-MPGA is better than the other three algorithms. (Wang & Wang, 2018)

In 2018, order acceptance and scheduling on the identical parallel machine were presented. In this study, a system with identical parallel machines is in a make-to-order manufacturing company. In this study, the objective is the maximization of total net revenue. The aim is to determine accepting orders and how to arrange the accepted orders in a processing sequence. Due to scheduling, the tardiness penalties, the revenue decreases. This study takes into account the due dates, order profits, tardiness penalties, and sequence-dependent setup times of each order. Two water flow algorithms are developed. With the same problem instances, two algorithms were compared. Two algorithms are compared with, particle swarm optimization embedded with variable neighborhood search, to find out their solution qualities in different sizes of problems. The results showed that the second water flow-like algorithm performs better. (Wu et al., 2018)

In 2019, the problem was studied with order acceptance and scheduling pricing problems (OASP) in a parallel machine environment. Each order has its due date, release date, deadline, controllable processing time, sequence-dependent setup time, and price in the MTO system. A linear demand function with primary market volume and demand price sensitivity of order is used while defining the revenue. A mixed-integer linear program was formulated to maximize the net profit. Compression amount and expansion amount are also considered as lost from revenue. In the study, the results of sample problems depending on the structure of data generation based on different values of R and τ on different machines and orders are examined and the easiest instances set for solving by the MILP are shown in the study. (Sahebe Esfandiari, 2019)

In 2020, an order acceptance on an identical parallel machine scheduling problem was studied. The objective of the study is the maximization of total net revenue. To solve the problem, first, a mixed-integer program was formulated. Then, a Branch-Relax-and-Check exact method is developed to solve the problem. The

mixed-integer program model solves 13% of instances optimally, achieving an average gap of 9.8%. The Branch-Relax-and-Check exact method solves 50% of instances optimally, achieving an average gap of 3.3%. (Naderi & Roshanaei, 2020)

In 2020, one of the latest studies on order acceptance and scheduling includes the constraints of energy consumption costs and machine launch costs. The objective of this study is the maximization of revenue. Until now parallel machine selection scheduling problems have been studied to minimize the makespan, number of the machine, and holding cost. This study, unlike other studies, studied maximization of revenue. The parallel machine selection, the release time of orders, and deteriorating jobs in the production of the accepted orders simultaneously are studied in a green manufacturing system. The NP-hardness of the proposed problem was also proven. For deciding on machine and order, a dynamic programming algorithm is used in the fitness measure method of the MVNS. For increasing the space of the neighborhood solution, the MVNS contains the characteristic of an encoding and decoding procedure with a Random-key to Value rule and two unique neighborhood structures, crossover neighborhood structure (CNS) and a mutation neighborhood structure (MNS). Based on the dynamic programming algorithm, the proposed MVNS algorithm that includes CNS and MNS neighborhood structures is compared with the seven VNS-based algorithms and with the other three meta-heuristic algorithms. The computational result showed that the proposed algorithm MVNS outperforms the other compared algorithms. (Kong et al., 2020)

2.3. Order Acceptance and Scheduling Studies with Single and Multiple Machines

In 2020, simultaneous order acceptance and scheduling on a single and also on 3 parallel machines were presented. In this study, orders have a different release date, a processing time, and a penalty. Each job can be processed by one machine or rejected. If an order is rejected penalty cost is paid. The objective of this study is to minimize the maximum completion time of all accepted jobs and the total penalties

of all rejected jobs. If the jobs have the same release dates $(3/2 - 1/2m)$-an approximation algorithm is used to solve the problem for the parallel machines. On the contrary, if the orders have general release dates, a 4/3-approximation algorithm is used for the problem on a single machine and a $(1 + \max [0.618, 1 - 1/m])$-approximation algorithm for the parallel machine problem, respectively. (Liu & Lu, 2020)

3. MATERIAL AND METHOD

3.1. Material

Fuzzy order acceptance and scheduling with the identical parallel machine (FOAS-IPM) is a decision problem in which order should be accepted and scheduled to have the maximum net profit. In this study, the firm has a make-to-order (MTO) system strategy and the orders are simultaneously received from the customers. Orders are known at the beginning of the planning horizon.

In this firm, each order has different specifications due to an MTO system strategy. Every order has different properties because of customer demands. Every customer needs different products. Therefore, in this firm with an MTO system, order variety is really large. Different types, different shapes, different quantities, different delivery dates of products can be requested by customers. It is very difficult for the manufacturer to cope with the process of producing many different kinds of products also with considering customer satisfaction. Especially the delivery or due date is at a critical point in customer relationship and satisfaction. Therefore, the firm wants to receive the products to the customers on the delivery date and achieve high customer satisfaction which also increases customer loyalty.

Customer loyalty can be achieved by following promises and agreements made with the customer. At this point, production planning is at a very critical point. To keep the promises given to the customers, the production facts and capacity of the company must be analyzed well. Good production planning is important to be delivered to the customer at the promised date, in the given quantity and quality. So in our study, due dates have a big importance. The relation between the due date and the completion date of the order constitutes the decision point of accepting or outsourcing. Thus, the company will keep its promise to the customer. Delivery will be guaranteed in case of late supply or on the promised date.

Continuous good relationships and loyalty are desired by companies rather than looking for new customers. Customer loyalty is very important for the company

to have a good advertisement in the business environment. Thus, the company with high customer loyalty will be easier to find more customers and its reliability in the business environment will be increased. Rejecting or refusing an order is a reason to lose the customer or at least it makes the relationship worse. So the firm in order not to lose and reject the customers, the customer orders are outsourced to another firm. But, other firms have usually different production and pricing policy. Due to different pricing policies and outsourcing pricing, companies tolerate small amounts of losses in order not to lose customers.

Outsourcing can sometimes have negative consequences for the company. In this study, in outsourcing, the manufacturer company is used for outsourcing companies due to their capacity constraints. However, sometimes due to the pricing policies, outsourcing can cause to pay for some products at higher prices than the company. Because of the firms' pricing policies, the goods or services purchased from outside are higher than the pricing policies applied by the firm. The company sometimes bears this cost in order not to lose the customer and maintain a long-term relationship with the customers. In this study, the firm must accept an order or outsource the order. If an order is outsourced, the firm doesn’t have any profit from the product and also loses a little amount of money due to the differences in outsourcing in pricing policies.

In this study, a firm wants to deliver the products to its customer on time. However, there are struggles between the due date and completion time. The completion time of an order is related to production time and sequence-dependent setup times. Although there is more than one identical parallel machine, the company cannot deliver some products before the delivery date because of capacity constraints. The completion time of the products depends on the production time and the sequence in which the orders are found. The completion time of the products depends on the completion time of the previous product on the machine and its own production time and the sequence-dependent set-up times of the order in the machine. If the order completion time is earlier than its due date, the product is

included in the production schedule. If the order's completion time is later than the order delivery time, the delay time is calculated. The delay discount for each of the customers is calculated by multiplying the delay time and the weighted delay penalty. Discounts are applied to the customer as much as the delay penalty cost. However, if the delay discount is equal to the revenue of the product, this order can no longer be accepted. So the rejected orders cannot be added to the scheduling list of products to be produced. These orders are added to the list of outsourced orders. For each outsourced order, an outsourced penalty cost occurs or no profit can be obtained from these outsourced orders.

In this section, the symbols and the properties of the parameters and the data generation of the parameters to test the formulation of the system will be defined. The data generation method also will be explained in each parameter definition section.

In this study, data sets containing 10,15,20,25,50, and 100 orders were created to test the developed algorithms. These data sets are named as small data set with 10 and 15 orders, medium data set with 20 and 25 orders, and large data set with 50 and 100 orders.

Data sets for 10, 15, 20, 25, 50, and 100 orders were created separately for the 2, 3, 4, and 5-machine systems. In total, 24 separate data sets groups were created. These 24 data sets groups contain a total of 4400 order data as shown in table 3.1 below.

Figure 3.1. Total 4400 instances of orders in each set of 10,15,20,25,50 and 100 orders

Order Set	Machine Number	Total Instances of Order
10	2,3,4,5	200
15	2,3,4,5	300
20	2,3,4,5	400
25	2,3,4,5	500
50	2,3,4,5	1000
100	2,3,4,5	2000
	Total	4400

3.1.1. The Crisp and Fuzzy Parameters of the Study

Some concepts that exist in real life and are easily perceived by the human brain are difficult to understand by the computer. This information needs to be transferred to the computer modeling environment to perform real-life system analysis.

Being able to deal with this situation and increase profits can be realized with a really good production system modeling and planning. Models have difficulty understanding real life like the human brain. In models installed with the aid of a computer can only realize the cases which are reduced to zero and one like the exact facts and propositions that are true or false.

Due to the obscurity of real life, it is sometimes impossible to predict the exact specification of a system. The uncertainty that exists in real life is also included in production systems. Uncertainty can be related to everything in a system. Especially while modeling a real-life system exact methods are insufficient to describe the uncertainty of the real world. Fuzzy logic offers better options to build models closer to real life.

In this study, each order has its characteristics. At the starting point of the planning horizon, all the parameter values of each order are known. In this study, orders are represented with the letter "*i* ", and order *i* have different parameter values which are revenue (e_i), fuzzy production time($\tilde{p}_i$), fuzzy due dates ($\tilde{d}_i$), fuzzy sequence dependent set up ($\tilde{s}_{ij}$,) the weighted penalty cost (w_i), and the outsourcing penalty cost (Co_i).

The properties and description of the parameters in the problem are given in this section. On the other hand, since the problem is addressed for the first time, there are no sample data sets previously developed for the problem. For this reason, new data sets suitable for the problem have been developed by using data sets previously studied in the literature. In this section, information is given about both the parameters and the formulation of the data sets produced.

3.1.1.1. The Crisp Parameter of Revenue of Order i (e_i)

The most important goal of a firm is to increase its net profit. On the contrary, the profit balance between delay costs, and the income from the customer should be analyzed well. In a make-to-order system, incomes are generally different from each other due to the variety of orders. The revenue is a crisp parameter. In this study, orders are represented with the letter "*i* "and the revenue of each order i is different from the other orders' revenue and represent as "e_i ". In this study, each revenue has an integer value between 1 and 20. Data generation of the revenue is generated randomly from a uniform distribution in the interval of [1, 20]. (Oguz, 2010)

3.1.1.2. The Crisp Parameter of Outsourcing Penalty Cost of Order i (Co_i)

The outsourcing penalty cost occurs when the orders are not included in the scheduling order list. The orders' due dates are important while giving the acceptance decision. The completion date of the product should be less than the due date not to pay the outsourcing penalty cost. The outsourcing cost is a crisp parameter. In this study, orders are represented with the letter " *i* " and the outsourcing penalty cost of each order i is generally different from the other orders' outsourcing penalty cost and represent as " Co_i ". In this study, each outsourcing penalty cost has an integer value between 0 and 4. Data generation of the outsourcing penalty cost is generated randomly from a uniform distribution in the interval of [0,4].

3.1.1.3. The Crisp Parameter of Weighted Tardiness Penalty Cost of Order I (w_i)

The weighted tardiness penalty cost occurs when the orders are included in the scheduling order list and also have a delay on the completion time of the order. The orders' due dates are important when giving this decision. The completion date of the product should be less than the due date not to pay the tardiness penalty cost.

The weighted tardiness penalty cost is a crisp parameter. In this study, orders are represented with the letter " *i* " and the weighted tardiness penalty cost of each order i is different from the other orders' weighted tardiness penalty cost and represents as " w_i ". In this study, each weighted tardiness penalty cost data is used from the literature. (Oguz, 2010)

3.1.1.4. The fuzzy Parameter of Production Times of Order i ($\tilde{p}_i$)

The fuzzy parameter of the production time of an order depends on the product. On each machine production of an order are equal to each other. Therefore, the fuzzy production times of orders do not depend on machines Due to the fuzzy environment of production times, there are 3 different production times for each order. The production times of the orders are generated integer numbers between 1 and 20. The completion date of the product depends on its production time. In the literature, production times Data generation of production times is generated randomly from a uniform distribution in the interval of [1,20]. (Oguz, 2010). However, the production time is a triangular fuzzy number. Due to the fuzzy environment of the production times, there are 3 different production times for each order. The triangular numbers of the data set of production time were produced with the formula {p-[a], p, p+[b]} a and b are integers between 1 and 2 and are randomly generated and p is the production time in the literature data set.

3.1.1.5. The fuzzy Parameter of Sequence Dependent Setup Times of Order i ($\tilde{s}_{ij}$)

The fuzzy setup time is the time between the transitions of a work center to another product that completes the operation on a product. Setup time is sometimes referred to as setting and sometimes called setup time. During the setup, production stops and there is no production activity at that time. So while calculating the completion of an order, the setup times must be considered and included. The fuzzy parameter of the sequence-dependent setup times depends on the product type. The

fuzzy sequence-dependent setup times of the orders are different from each other and it depends on the order and the sequence of the orders. However, the fuzzy setup times are the same for the same sequence of orders on each machine. Due to the fuzzy environment of setup times, there are 3 different setup times for each order. The fuzzy set up times of the orders are integer numbers and generated randomly from a uniform distribution between 1 and 10. Due to the fuzzy environment of the setup times, there are 3 different setup times for each order. The completion date of the product depends on its production time, sequence-dependent setup times. However, the setup time is a triangular fuzzy number. The triangular numbers of the data set of setup time were produced with the formula $\{\tilde{s}_{ij}$ - [a], $\tilde{s}_{ij}$, $\tilde{s}_{ij}$ + [b]$\}$ a and b are integers between 1 and 2 and are randomly generated and $\tilde{s}_{ij}$ is the setup time in the literature data set. (Oguz, 2010)

3.1.1.6. The fuzzy Parameter of Due dates of Order i ($\widetilde{d}_i$)

Due Date or in other words delivery date refers to the date when, in case of purchasing products in case of manufactured items, items to be manufactured such as products or the products are scheduled to be completed. Generally speaking, however, the delivery date refers to the date when products should be delivered to the customer company or the customer. In this study, if a product is completed after the due date, the product isn't included in the manufacturing scheduling list. These products are included in the manufacturing scheduling list if the delay discount is smaller than the revenue of the product. The delay time is the difference between completion time and delivery or due date. If the delay discount is bigger than the revenue, the product has to be outsourced by the firm and the firm must pay a outsource penalty cost. The due date is critical while deciding on scheduling. Due to the fuzzy environment of the due date, there are 3 different due dates for each order. The fuzzy due date of the orders are integer numbers and generated randomly from a uniform distribution related to production time and set up times. (Oguz,

2010). While generating due dates in this study, generated due dates are divided by the number of each machine group (m=2,3,4,5). And round to the closest integer. However, the due date is triangular fuzzy number. The triangular numbers of the data set of setup time were produced with the formula $\{\tilde{d}_i$ -[a], $\tilde{d}_i$, $\tilde{d}_i$ +[b]$\}$ a and b are integers between 1 and 2 and are randomly generated and $\tilde{d}_i$ is the due date in the literature data set. (Oguz, 2010)

3.2. Method

In this section, the properties of the formulation are explained. After explaining the formulation, the properties of the system are explained. Afterward, the mathematical model is shown, the objective function and the constraints of the mathematical model in the study are released. System properties and the assumptions in the study are also explained. After explaining the mathematical model of the study, the meta-heuristics for solving fuzzy order acceptance and scheduling (FOASIPM) are discussed. In the other section, the proposed metaheuristic method is explained and the details of the algorithm are shown. In the last section, the suitable code of genetic algorithm for FOASIPM and the developed program for solving the problem are introduced, given information about, and discussed.

3.2.1. Problem Formulations

In FOASIPM problems, the system is analyzed carefully. The properties of the system are the most important part while modeling the system. The mathematical formulation of the system depends on the system parameters. In this section, firstly the features of the system are mentioned. Later, the mathematical model of the system is introduced. The objective function and constraint equations of the mathematical model will be explained.

3.2.1.1. Definition of the System and Assumptions

The company with this problem has a system that produces a high range of product variety. In this study, when the planning production is done, all order features are known at the beginning of the planning process. In the system, there are more than identical parallel machines. Machines do not remain idle in the system. So there is no free time for machines. Jobs entering the machines cannot be stopped. So interruption or preemption is not allowed in this system. Products can be produced on any machine. There is no priority between the machines. Also, there is no time limitation on machines. Any product can be produced on any machine. The machines have the same features. As mentioned before, the machines are positioned parallel. Machines are assumed not to be broken while the system is working. Machine breakdown times in the system are not included.

The order numbers produced by the machines are required to be close to each other. However, the total production times in the machines do not have to be equal. It is desired to have an order schedule that will bring the highest profit to the producer. Order completion times depend on production sequences. There are penalties for rejection and delay in the system. Order scheduling is required to achieve the highest profit by minimizing these penalties.

Orders are known at the beginning. As explained in Figure 3.1. the orders are ready before planning. According to their delay penalty cost, the orders can be accepted to the manufacturing scheduling list or outsourced order list. Accepted orders are scheduled to give the highest net profit. Orders are randomly assigned to machines. Machines have no priority over each other. Same way orders also do not have priority over each other.

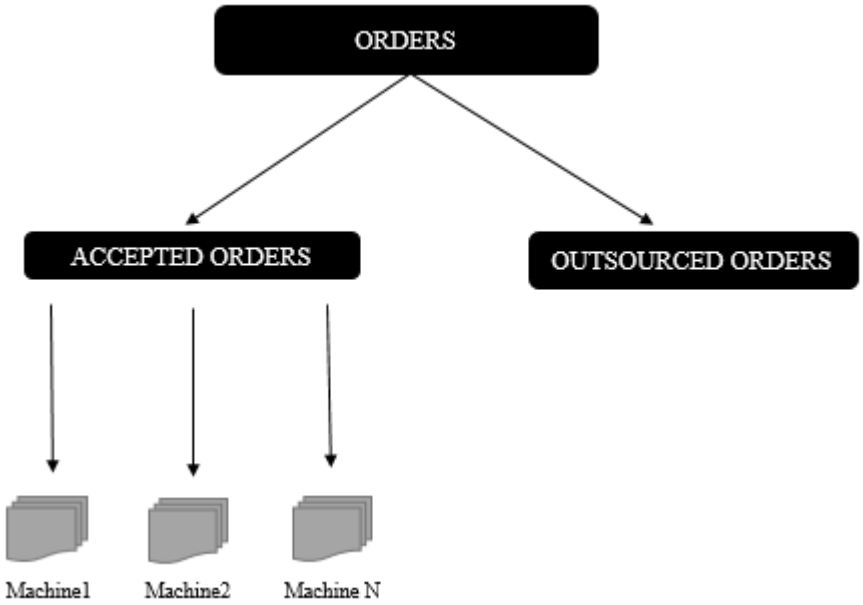


Figure 3.1. The flow of the received orders in the system

All order properties are known at the beginning of system planning. All orders have a specific revenue of orders, production time, set-up times depending on the order, delivery date, weighted delay penalty, and outsourcing penalty cost. The completion of an order depends on the sequence of the order. Order completion for the orders produced on the same machine depends on the completion time of the previous order. There is a chain relationship between orders produced on the same machine. Before an order is completed on the same machine, another order cannot be produced.

The completion date of an order depends on its sequence on a machine, production time, and sequence-dependent setup times. Set up times of the system changes according to the order and the sequence of the orders but doesn't change according to the machine. Also, the setup times between the products are variable. Setup times are not dependent on the machine and only depend on the order and also depend on the sequence of the orders.

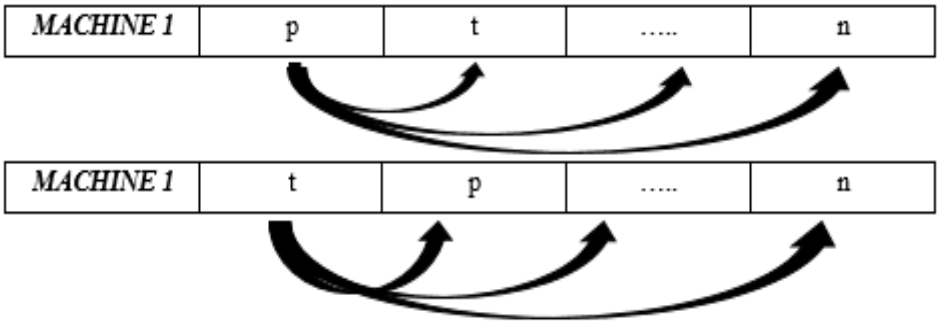


Figure 3.2. The sequence and the reverse sequence on the same machine for the same orders set up times

In Figure 3.2. the sequence and the reverse sequence on the same machine for the same orders set up times which are not equal are represented for the same machines $S_{pt} \neq S_{tp}$

The sequences of the orders in each machine are important because of the sequence-dependent setup times and completion date of the product. For example, for order n on machine 1, the completion time of order n depends on the completion time of the previous products t,p, and until the product n-1 including their sequence-dependent set up times in the sequence chain. Another example can be on machine 1, the completion time of order n depends on the completion time of previous orders p,t, and until the order n including their sequence-dependent set up times that are different from the first example due to sequence of the same orders p and t.

Another assumption of this study is having the flexibility to charge either equal or different prices to each customer. The main objective of the study is to provide insights regarding the benefits of price customization, inventory flexibilities, and lead time in various market settings. The problem is formulated as mixed-integer linear programming.

The number of the solution space for the FOAS-IPM problem can be calculated in the following manner: Suppose that there are n incoming orders for the firm. And, an acceptance set that contains j orders. (Oguz, 2010) The sequence

number of different sequences possible for this acceptance set for one machine is j! If there are two machines, suppose that the accepted orders j which is $j=m+n$ where m is the orders on the first machine and n is the orders on the second machine. m! ×n! ×2!. The number of n orders and j different selections for set A is C (n, j).

The total number of solutions with n orders, j accepted orders and two machine includes feasible and infeasible in the solution space is $\sum_{j=1}^{n} C(n,j)$×m!*n!*2! As seen in the formulation when the parameters n, j, and the number of machines increases, the problem and the solution space will increase, and solving the problem becomes very time-consuming. For each order also there is a possible solution space.

In the next section, the developed mathematical model for fuzzy order acceptance and scheduling is presented.

3.2.1.2. Mathematical Model

Notation for fuzzy order Acceptance and Scheduling on the identical parallel machine Sets:

O : order set O

M : machine set M

Index:

i : order index (i=1, ..., O)

j : order index (j=1, ..., O)

m : machine index (m=1,...,M)

Parameters:

w_i : The weighted delay penalty cost of order i, i∈O

Co_i : The outsourced penalty cost of order i, i∈O

e_i : The revenue of order i, i∈O

$\tilde{p}_i$: The fuzzy parameter of production times of order i, i∈O

$\tilde{d}_i$: The fuzzy parameter of the due date of order i, i∈O

$\tilde{S}_{0i}$: The fuzzy parameter of beginning setup times of order i, i∈O

$\tilde{S}_{ij}$: The fuzzy parameter of sequence-dependent setup times of order i, i∈O

B: Big positive number

A: Average of assigned order to each machine

Decision variables:

C_i: The completion time of order i, i∈O

T_i: The delay time of order i, i∈O

a_i: Acceptance of order i,if accepted 1, otherwise 0. i∈O

b_{im}: Assignment of order i on machine m as 1, if not 0. i∈O

x_{ijm}: Assignment of order j after order i on machine m as 1, if not 0. i∈O

Objective Function

$$\text{Max} \sum_{i \in O}(K_i) \quad (3.1)$$

Constraints

$$a_i = \sum_{m \in M} b_{im} \qquad \forall i \in O \quad (3.2)$$

$$\sum_{m \in M} b_{im} \leq 1 \qquad \forall i \in O \quad (3.3)$$

$$B.\, a_i \geq C_i \qquad \forall i \in O \quad (3.4)$$

$$C_i \geq \tilde{s}_{0i} b_{im} + \tilde{p}_i b_{im} \qquad \forall i \in O\ \forall m \in M \quad (3.5)$$

$$C_i \geq \tilde{s}_{ij} x_{ijm} + \tilde{p}_i b_{im} \qquad \forall i \in O\ \forall m \in M \quad (3.6)$$

$$C_j - C_i + B(3 - x_{ijm} - b_{im} - b_{jm} \geq \tilde{s}_{ij} + \tilde{p}_j \qquad \forall i \in O\ \forall j \in O\ \ i \neq j \quad \forall m \in M \quad (3.7)$$

$$C_i - C_j + B(2 + x_{ijm} - b_{im} - b_{jm} \geq \tilde{s}_{ji} + \tilde{p}_i \quad \forall i \in O \ \forall j \in O \ i \neq j \quad \forall m \in M \tag{3.8}$$

$$T_i \geq C_i - \tilde{d}_i \qquad \forall i \in O \tag{3.9}$$

$$x_{ijm} \geq b_{jm} \qquad \forall i \in O \ \forall j \in O \ i \neq j \quad \forall m \in M \tag{3.10}$$

$$x_{ijm} \geq b_{im} \qquad \forall i \in O \ \forall j \in O \ i \neq j \quad \forall m \in M \tag{3.11}$$

$$R_i = (e_i \times a_i) - (w_i \times T_i) \qquad \forall i \in O \tag{3.12}$$

$$K_i = (R_i - ((1 - a_i) * Co_i) \qquad \forall i \in O \tag{3.13}$$

$$A \geq \sum_{i \in O} b_{im} \qquad \forall m \in M \tag{3.14}$$

$$C_i, T_i \geq 0 \ \ C_0 = 0 \qquad \forall i \in O \tag{3.15}$$

$$a_i, b_{im}, x_{ijm} \in \{0,1\} \qquad \forall i \in O \ \forall j \in O \ i \neq j \quad \forall m \in M \tag{3.16}$$

Crisp constraints (17), (18), (19),(20) and (21) are converted from fuzzy constraints respectively (5),(6), (7), (8) and (9) with the signed distance method for triangular fuzzy numbers.

$$C_i \geq \tilde{s}_{0i} b_{im} + \tilde{p}_i b_{im} \tag{3.5}$$

$$C_i \geq \frac{1}{4} b_{im}[(s1_{0i} + 2s2_{0i} + s3_{0i}) + (p1_i + 2p2_i + p3_i)] \quad \forall i \in O \ \forall j \in O \ i \neq j \ \ \forall m \in M \tag{3.17}$$

$$C_i \geq \tilde{s}_{ij} x_{ijm} + \tilde{p}_i b_{im} \quad (3.6)$$

$$C_i = \frac{1}{4}\left[\left(\left(s1_{ij} + 2s2_{ij} + s3_{ij}\right) * x_{ijm}\right) + \left(\left(p1_i + 2p2_i + p3_i\right) * b_{im}\right)\right] \quad \forall i \in O \; \forall j \in O \; i \neq j \; \forall m \in M \quad (3.18)$$

$$C_j - C_i + B(3 - x_{ijm} - b_{im} - b_{jm}) \geq \tilde{s}_{ij} + \tilde{p}_j \quad (3.7)$$

$$C_j - C_i + B(3 - x_{ijm} - b_{im} - b_{jm}) \geq \frac{1}{4}\left[\left(s1_{ij} + 2s2_{ij} + s3_{ij}\right) + \left(p1_j + 2p2_j + p3_j\right)\right] \; \forall i \in O \; \forall j \in O \; i \neq j \quad \forall m \in M \quad (3.19)$$

$$C_i - C_j + B(2 + x_{ijm} - b_{im} - b_{jm}) \geq \tilde{s}_{ji} + \tilde{p}_i \quad (3.8)$$

$$C_i - C_j + B(2 + x_{ijm} - b_{im} - b_{jm}) \geq \frac{1}{4}\left[\left(s1_{ji} + 2s2_{ji} + s3_{ji}\right) + \left(p1_i + 2p2_i + p3_i\right)\right] \; \forall i \in O \; \forall j \in O \; i \neq j \quad \forall m \in M \quad (3.20)$$

$$T_i \geq C_i - \tilde{d}_i \quad (3.9)$$

$$T_i \geq C_i - \frac{1}{4}\left(d1_i + 2d2_i + d3_i\right) \quad \forall i \in O \quad (3.21)$$

The Objective Function

The objective function (Equation 3.1) of the Fuzzy Order Acceptance and Scheduling on Identical Parallel Machine (FOAS-IPM) problems aims to maximize the total net revenue. Total net revenue is the difference between the revenue of orders and delay penalty (R_i). If the differences between the revenue and the delay penalty are equal to 0 or negative, the outsourcing penalty for the order must be paid or no profit can be gain from the product not to lose the customer (K_i). Each order has a different revenue, weighted tardiness penalty, and outsourced penalty.

Constraints

The first constraints function (Equation 3.2) ensures that if an order is accepted its acceptance binary variable becomes 1 and an accepted order can be assigned only one machine. If an order isn't accepted, its acceptance binary variable takes the value of 0. So, this order cannot be assigned to any machine.

The second constraints function (Equation 3.3) ensures that if an order is assigned to a machine, this order can be assigned to only one machine.

The third constraints function (Equation 3.4) ensures that if an order isn't accepted, its completion time must be 0. On the contrary, if the product is accepted, there is no restriction on its completion time.

The fourth constraints function (Equation 3.5) ensures that the completion time of the first order produced on a machine is at least, must be equal to the sum of set up time and production time.

The fifth constraints function (Equation 3.6) ensures that the completion time of an order produced on a machine is at least, must be equal to the sum of set up times and its production time.

The sixth constraints function (Equation 3.7) calculates the differences between the completion times of orders. If two orders are two consecutive orders on the same machine, the difference between completion times is at least the sum of the set-up time and the production time. This constraint does not apply if two orders are not two consecutive orders on the same machine or different machines.

The seventh constraints function (Equation 3.8) calculates the differences between the completion times of orders. If two orders are not two consecutive orders on the same machine, the difference between completion times is at least the sum of the set-up time and the production time. This constraint does not apply if two orders are not two consecutive orders on the same machine or different machines.

The eighth constraints function (Equation 3.9) calculates the delay time of order. If the production of an order is completed after the delivery date, the delay time of the order is the difference between the completion date and the due date. If

the production of an order is not completed after the delivery date, there is no delay for that product.

The ninth constraints function (Equation 3.10) and the tenth constraints function (Equation 3.11) ensure that if two orders are consecutive orders on a machine, these two orders must have been assigned to that machine.

The eleventh constraints function (Equation 3.12) is the calculation of the net revenue if a delay occurs. The equation refers to the net profit. If there is no delay, the net profit is equal to the revenue of the order.

The twelfth constraints function (Equation 3.13) is the calculation of the net revenue if an outsourcing cost occurs. The equation refers to the net profit. If the order is outsourced, the net profit of the order becomes zero or a negative value. If the order isn't outsourced, the net profit of the order becomes equal to the net revenue with or without delay punishment (equals net return at equation 3.12).

The thirteenth constraints function (Equation 3.14) ensures that orders be assigned to each machine in equal numbers for production.

The fourteenth constraints equation (Equation 3.15) ensures that completion time and delay time must be at least zero or greater.

The fifteenth constraints equation (Equation 3.16) is defining the binary variables of accepting the order, assigning it to the machine, and sequencing in the machine which can take the value of 1 or 0.

The equations 3.17, 3.18, 3.19, 3.20, and 3.21 are the equations that have been developed to calculate the parameters expressed by fuzzy numbers of expressed equations 3.5, 3.6, 3.7, 3.8, and 3.9 respectively. To obtain these fuzzy equations the signed distance method is used which is expressed and defined in the first chapter in fuzzy sets.

3.2.2. Meta-Heuristic Algorithms

Metaheuristic solution methods are one of the methods that are used more and more day by day in solving optimization problems. Today, thanks to the

developing mathematical methods and computer technologies, the problems of more complex and difficult systems have been tried to be solved. Especially in optimization problems, when optimizing complex systems, deterministic programs that have come up to are not sufficient. The term “heuristic” means that a method of learning or solving problems that computers to discover things themselves and learn from their own experiences and the term “meta” means that beyond anything. So, the term “meta-heuristic” can mean moving forward with the determined method while looking for a solution on its own.

Optimization is finding the solution that is closest to the desired objective among the possible solutions of a problem. Many algorithms have been developed today to solve optimization problems. Optimization algorithms are generally divided into two groups, those are heuristic optimization algorithms and mathematical optimization algorithms. While mathematical optimization algorithms aim to solve by scanning the entire solution set, heuristic optimization algorithms approach the solution set intuitively and aim to reach the best solution or a solution close to the best solution. Mathematical optimization algorithms are costly to use in problems with a large set of solutions. Therefore, mathematical optimization problems are not suitable for problems with a large set of solutions. Scheduling problems in particular are large problems as their solution sets. For example, in a data set with n order sets, possible n! sequencing is possible. Because of the large solution set, it is not easy to scan all possible solutions in a mathematical optimization algorithm.

Heuristic optimization algorithms are more advantageous and more preferred in solving very large problems in the solution set. Metaheuristics are not specific to the problem type. However, optimization algorithms also have differences in solving different problems. An optimization algorithm cannot be expected to be successful in all types of problems or test functions. For this reason, it is necessary to determine carefully which type of problem find best solved with which algorithm. Nowadays, as a result of the effective use of basic heuristic methods, algorithms named “Metaheuristic” have been developed. Metaheuristic algorithms find the

optimum solution in the solution space faster according to the other methods because of using efficient search processes in a high-level working environment. (Çelik 2019)

Metaheuristic algorithms are methods that manage the search process to solve the problem and find the desired or the closest solutions to the desired result. Their goal is to get the best or near-best results by exploring the search space effectively. They have a structure ranging from local search techniques to learning processes. They offer an approximate solution rather than a complete solution, often non-deterministic methods. Metaheuristics are not specific to a problem type. They can be adapted to any problem. In general, they have structures in the search space that will avoid getting stuck in the local best positions.

A wide variety of heuristic algorithms are used for machine scheduling problems. Meta-heuristic methods are among the popular methods with very successful results in recent studies on machine schedules. The heuristic methods commonly used in the literature are the Genetic Algorithms, Simulated Annealing Algorithm (SA), Tabu Search Algorithm (TS), Artificial Bee Colony Algorithm (ABC), Ant Colony Optimization, and Particle Swarm Optimization (PSO).(Barak, 2015)

In this study, the mathematical model MILP is developed and was solved with the Cplex Solver in the GAMS program. However, due to the large solution set of FOAS-IPM problem solving, only a small number of data sets (up to 14 orders) were solved by the developed MILP with the GAMS program. (As shown in Figure 3.3)

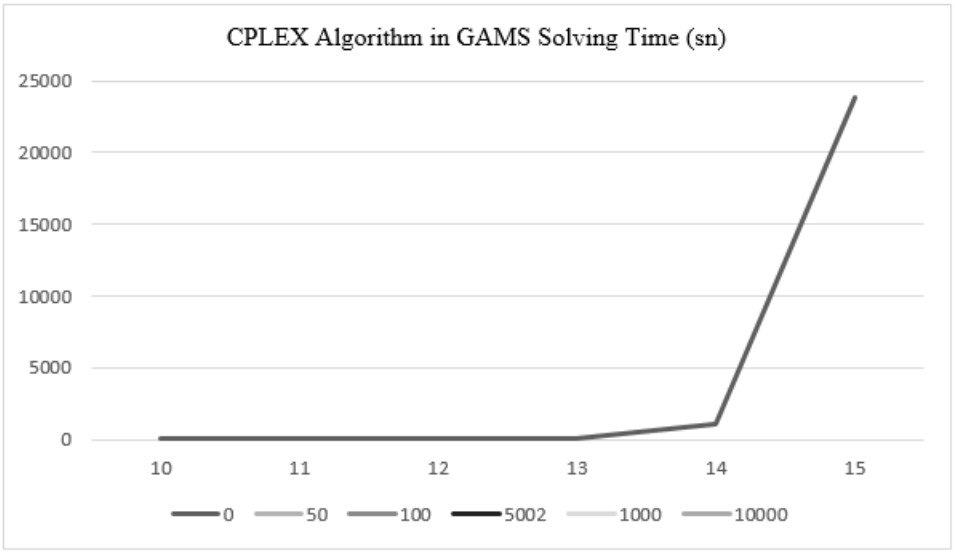


Figure 3.3. CPLEX Algorithm in GAMS Solving Time (sec) the Data Set with 10,11,12,13,14 and 15 Order on 3 Identical Parallel Machines

The CPLEX Algorithm in the GAMS program solves the MILP with the data sets with the numbers 10, 11,12,13,14, and 15 on 2, 3, 4, and 5 identical parallel machines. The program found the optimum solution for 10, 11, 12, and 13 data sets for 2, 3, 4, and 5 machines in a maximum of 35 seconds. However, when the number of orders in the data set increased to 14, the solution time increased considerably and the solution time was approximately 1020 seconds for 2, 3, 4, and 5 machines. When the number of order data sets is 15, the CPLEX algorithm in the GAMS program for 2, 3, 4, and 5 machines, the problem could not be solved the problem by even working 36.653,75 seconds which is the upper bound of the program for solving this problem.

Therefore, to solve the large size of this FOAS-IPM problem, Metaheuristic algorithms are developed. The algorithm developed by generating random numbers is successful in solving problems with large solution sets thanks to its structure that scans the solution space with randomization solution finding methods.

3.2.2.1. The Second Proposed Approach: The Genetic Algorithms

In this section, the definition of the Genetic Algorithm (GA) and the definition of the parameters, the most important methods used, and the working principle of the Genetic Algorithm are explained. In this study, two Genetic Algorithms are proposed with the name GADOC (Genetic Algorithm with Davis Order Crossover) and GATPC(Genetic Algorithm with Two-Point Crossover) to handle the problem on the Fuzzy Order Acceptance and Scheduling On Identical Parallel Machine (FOAS-IPM). The proposed algorithms are tested with small, medium, and large-sized dataset instances for comparing the performance of the proposed Genetic Algorithms.

The Genetic Algorithm (GA) is an evolutionary heuristic algorithm that is inspired by natural evolution that has been known in the literature for more than 30 years. Genetic algorithms (GA), invented by John Holland in the 196Os, are the most widely used approaches to computational evolution. (Mitchell, 1998) The implementation area of the GA is quite wide. Research related to GA has extended from computer science to engineering, and to such other fields as molecular biology, immunology, economics, and physics. GA is used for both problem solving and modeling. Today, the usage areas of genetic algorithms are expanding. Some of them are workshop scheduling, artificial neural networks design, display control, electronic circuit design, optimization, expert systems, packaging problems, machine, and robot learning, traveling salesman problems, economic modeling, etc.(Dr. Öznur İşçi, 2003)

GA is a heuristic approach also especially mostly used in optimization problems. Heuristic algorithms are used in scheduling problems due to the large solution space. GA is one of the most used algorithms in machine scheduling problems. Inspired by the science of genetics, GA seeks a solution to the problem with crossover and mutation operators. Genetic algorithms imitate the evolutionary process in a computer environment to solve problems. As in the science of genetics, while searching for new solutions in the solution space, GA uses parts from old

solutions. As described in the science of genetics, when a new individual is formed, the individual has the genes both from a mother and a father.

The working principle of GA is generally characterized by chromosome, gene, population, fitness ratio, selection, crossover, and mutation parameters. In the Genetic Algorithm, *the gene, the chromosome, the population*, and *the fitness function* are the main parts of the algorithm. *The gene* represents the smallest member of the algorithm. *The chromosome* is the collection of these parts to show a group solution or a group to calculate a solution. *The fitness function* is the solution of each chromosome or each gene. *The population* is the first community, to compare the fitness function and represents the community in which the crossover is made in each iteration. *The chromosome* is a series of individuals in which the problem is defined to solve the problem. As described in figure 3.3, GA is a population-based search consisting of five main operators: *Initialization, Selection, Crossover, Mutation, and Replacement.* (Rakesh Kumar, 2013)

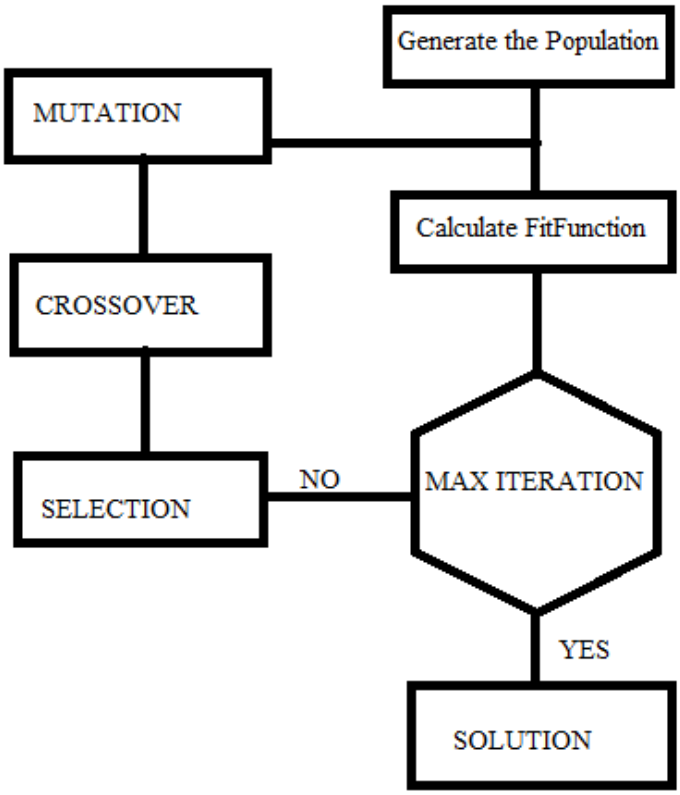


Figure 3.4. Genetic Algorithm Simple Pseudo Code

The Genetic Algorithm begins by defining the parameter sequences and chromosomes (individuals) to be optimized. It should be designed in such a way that the parameters that will take place on the chromosome and all the information necessary for the algorithm to search for the best solution in the solution space. Thus, it allows obtaining the desired results with the processes in the genetic algorithm. The smallest significant structure in the chromosome is called *a gene*. Each parameter in the chromosome creates *the gene*. In scheduling problems, the smallest structural unit gene represents an order. The sequence of genes on a chromosome represents the order sequence

The initialization in the GA is one of the most important steps in the algorithm. Especially, initial population size determination is a very important step

for GA. The size of the population, in other words, the number of chromosomes or individuals, is one of the important factors affecting the success of GA or the solving time and chance of reaching the optimum result. For the optimization cycle to start and work correctly, the parameters that need to be optimized must be transformed and defined under the structure to be applied. This is called encoding. Coding is a core issue for GA. Because the information obtained from the system is transferred to the genetic algorithm by coding. The gene string stores the problem-specific information. Each element called a gene is usually expressed as a string of variables. Variables can be represented as binary or real numbers, and the range is specific to the problem. In this scheduling problem, each gene is identified by a random number between 0 and 1. Each of these defined values is given a different number of labels as much as the order number. All chromosomes to be processed in the algorithm or other words, individuals come together and constitute the population. Determining how large the size of the population should initially be, in terms of scanning the solution space of the algorithm and reaching the result. It is a very important stage. Population size, in other words, the number of the initial chromosome or the number of individuals in the population affects the success of the algorithm or the time to reach the optimum result. Initial population size is one of the most important factors that affect the performance of the algorithm. Underestimating the initial population size is a narrow method of reaching the result in the scanning solution space of the algorithm. The narrow solution space will increase the probability of remaining out of the optimum solution space. When the space to be scanned by the algorithm gets larger, the algorithm solution space becomes too large and increases the probability of meaningful results. On the contrary, due to scanning large solution space, the algorithm reaches the result slow and late. To get the more efficient result from the searching, the algorithm must have a large solution space but the constraint of time must be determined well. So, the optimization of solution space size and solution time is very crucial and this balance between them must be determined carefully. The fitness value serves to understand how appropriate these solutions for each

chromosome are according to the constraints of the problem. This value, which indicates the rate of finding a good or bad solution in the population of the chromosome, allows us to understand the importance of this chromosome in the population. The fitness value of a chromosome, and the ability to find good results in the population according to this fitness value, are important for the selection of individuals in obtaining the next generation. To create the genetic algorithm and find the best solution for the problem, the individuals who will produce solutions should be shown correctly, the fitness function should be created effectively, the right genetic processors should be selected. While the new population is being produced, it is ensured that the individuals with a high fitness value in the previous population match with the individuals whose fitness functions are worse, so that the new individual is provided with a better result than the bad individuals. Thus, as new individuals or new chromosomes enter the population, these individuals become individuals that give better results. The higher the fitness value of a chromosome solution increases the chance of survival and reproduction of that chromosome, and the rate of representation of this chromosome in the next generation is high. (Yapici, 2012)

The selection in the GA is the process of selecting the chromosomes (individuals) to be produced from the existing population to create a new population. The determining selection method is also used in the selection of chromosomes to enter the crossing and mutation processes, which are the further parts of the algorithm. In short, the selection is aimed at choosing a new individual to be treated. There are several selection methods used in GA. The commonly used selection methods like the *Elitism Selection, Roulette Wheel Selection, Tournament Selection, Rank Selection, Steady State Selection, and Boltzmann Selection.* These are the methods used in the genetic algorithm for selecting. In this study, the elitist strategy is used for the selection method in the genetic algorithm because in studies elitism prevents the loss of good solutions found during the search process in the Genetic Algorithm. (P., 2003)Chromosomes that do well in elitist strategy are more likely to

be kept alive in progressive populations. In both developed algorithms, the population is ranked in descending order according to the fitness function value. As a result of this ranking, the good outcome half of the population is split equally in the form of elite and the other half nonelite. All of the genes on the chromosomes of elite individuals are directly passed on to other generations. But non-elite genes must pass from the crossover process. After non-elite chromosomes enter the crossover process, their genes pass on to the next generation. In the crossover process, the selection of parents is done as follows. The best parent in the elite division and the worst parent in the non-elite division are selected for the crossover operation. The previous individual from the second elite individual and the non-elite individual with the worst fitness function value, respectively, are selected for the crossover process. As explained in the figure, all non-elite chromosomes are subjected to a cross-over process, and the gene transfer to another generation can be done in that way.

Figure 3.5. Classification of Population According Their Fitness Function Value

The crossover in the GA is the process of generating new springs by selecting the chromosomes (individuals) to be produced from the existing population to create a new population. Many different crossover methods are used in the literature. Different crossover methods have been used for both developed genetic algorithms in this study. In this study, half of the newly created population comes from elite individuals from the previous population as described in figure 3.3. Offspring in the newly created population is formed after the crossover of elite and non-elite individuals. (Lei He, 2019) By using an elitist strategy, an individual with the worst fitness value is replaced by an elite with an individual who is more likely to have a better fitness function after crossover, so that the new society consists of better individuals. Thus, on the one hand, the possibility of not transferring the individual with the highest fitness value to the next generation is eliminated. Besides, the probability of an increase in the average fitness value of the newly created population will also increase, the population will approach the desired optimum value.

In this study, the best elite individual and the worst non-elite individual were selected to undergo a crossover process as described in figure 3.4. This process continues by performing a crossover between the next best elite individual and the next worst non-elite individual, respectively. The process continues until all elite individuals and all non-elite individuals are crossed. All elite individuals pass directly to the new population. But for the new springs, there is a comparison of fitness functions to pass the new population. The fitness functions of the newly formed springs must be better than the fitness functions of the old non-elite individuals. After the comparison process, it is decided whether the new individual will join the new population or not. If the fitness function of the newly formed spring is better than the old individual, the new individual is included in the new population. However, if the fitness function value of the newly formed individual is worse than the old individual, the newly formed individual is subjected to the mutation process and added to the new population as described in figure 3.6.

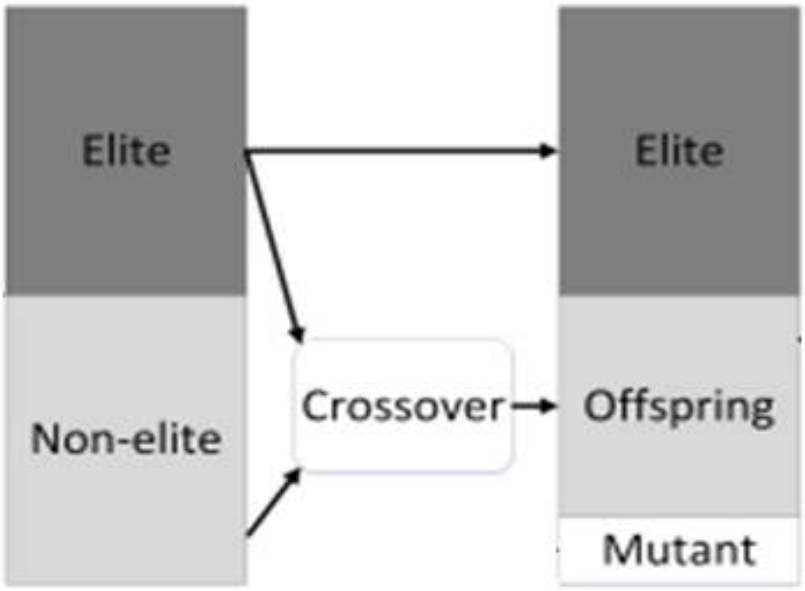


Figure 3.6. The Crossover and Mutation Process of Both Genetic Algorithms

Davis' Order Crossover and Two-Point Crossover operators are used in this study. Davis' Order Crossover for permutation-based crossovers to transmit information about relative ordering to the off-springs. Davis' Order Crossover works first with detecting two random crossover points in the elite parent and copies all the segments between them from the first parent to the first offspring. After, starting from the second crossover point in the second parent, copy the remaining unused numbers from the second parent to the first child, wrapping around the list as described in figure 3.7. (Mitchell, 1998)

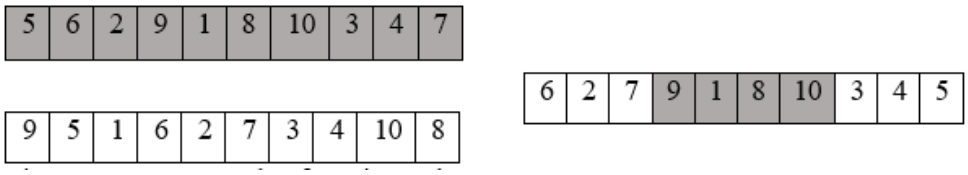


Figure 3.7. An example of Davis' Order Crossover

The other Crossover method used in this study is Two-Point Crossover which is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs as described in figure 3.8. In this study, Two-Point Crossover has been done. The range of the order number and the 2 points on the same chromosome are chosen randomly. Also, the distance between these two selected points was chosen randomly. After the Two-Point Crossover process, two new offspring chromosomes are formed. However, only half of the new population can consist of new offspring. Therefore, one of the new offspring individuals is selected. The offspring, which has a better fitness function than the other offspring, pass to the new population.

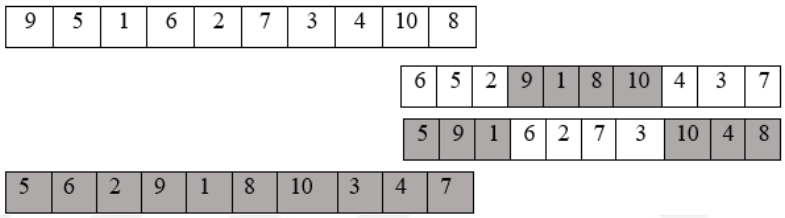


Figure 3.8. An example of a Two-Point Crossover for n=10

The mutation in the GA is the operation of generating new springs by applying the small random tweak in the chromosomes (individuals) to differentiate, maintain and introduce diversity in the genetic population and is usually applied with a low probability to the new population. The mutation used in GA has emerged from the idea of mutation existing in genetic science. The mutation is the change of genes in a chromosome rather than a crossover. The mutation is an operator used for the alteration or differentiation of chromosomes. It is made to create chromosomes that cannot be obtained by crossover or are different than those obtained in the crossover. Thus, by creating different chromosomes from the parents, these new chromosomes can produce different results than previous chromosomes. With this change, the new chromosomes can reach the optimum result faster. Depending on the nature of the

problem, one of the different mutation operators can be used. There are different kinds of mutation operators in the literature. Some of the mutation operators are Swap Mutation, Displacement, Scramble Mutation, and Inversion Mutation.

In swap mutation, two points on the chromosome are chosen randomly and the values are interchanged as described in figure 3.9 Swap mutation is permutation-based.

Figure 3.9. Example of Swap Mutation for n=10

Scramble Mutation is a popular mutation with permutation representations. In a chromosome, a subset of the entire chromosome is chosen, and their values are scrambled or shuffled randomly as described in figure 3.10.

Figure 3.10. Example of Scramble Mutation for n=10

In inversion mutation, a subset of genes like in scramble mutation is chosen, but instead of shuffling the subset, the entire string in the subset is merely inverted as described in figure 3.11.

Figure 3.11. Example of Inversion Mutation for n=10

In this study, the mutation operator is used when the newly formed offspring are not better than their parents. The newly formed offsprings' fitness functions must be better than the old individuals'. With this comparison process, it is decided whether the new individual will join the new population or be mutated. If the fitness

function value of the newly formed offspring is worse than the old individual, the newly formed offspring will be mutated and added to the new population. In this study, the Swap Mutation operator is used for the mutation process. The process is carried out by changing the places of the two chosen random orders on two random spots on the newly formed offspring chromosome. After this mutation process, a new offspring chromosome with a new different sequence is added to the newly formed population.

The Replacement is the final process of the Genetic Algorithm. As described in figure 3.5, the elite chromosomes pass directly to the new population without any change. But non-elite chromosomes must be a crossover with the elite chromosomes. As a result of the cross-over process, if the new offsprings have better fitness function value than non-elite individuals, they replace the non-elite individual in the new population. New offspring that are better than their non-elite parents can join the new population after the mutation process. New offspring have worse fitness function than their non-elite parents can join the new population after the mutation process. Thus, by comparing the fitness function and increasing the diversity in the population, this new population comes closer to finding the optimal solution.

```
{
    Generate initial Population P0
    Evaluate Population P0
    While (stopping GA criteria not satisfied) Repeat
        {
        Sort chromosomes descending order of FitFunc
        Find elite chromosomes (Half of the population)
        Add elite chromosomes to the new population
        Remove non-elite chromosomes
        Crossover (Create new chromosomes from elite and nonelite)
        Evaluate new chromosomes
            If Fitfunc_newchromosome < Fitfunc_nonelitechromosome
            Mutation new chromosomes
            Add new chromosome to the new population
            Else
            Add new chromosomes to the new population
        }
}
```

Figure 3.12. General Puseodocode of Genetic Algorithms

3.2.2.2. GADOC and GATPC Algorithm for Fuzzy Order Acceptance and Scheduling

Fuzzy Order Acceptance and Scheduling on an identical Parallel Machine (FOAS –IPM) is a special order scheduling problem. The aim of this problem is net profit maximization. In this problem, the production process is carried out in a multi-identical parallel machine system. FOAS IPM problem has been studied in this study with different data sets ranging from 10 to 100. Also, systems with 2, 3, 4, and 5 machines were studied. It is very difficult to analyze the developed mathematical model with the GAMS program. Regardless of the number of machines, a solution was found only for an order set of 14 with the mathematical model in the GAMS

program. For this reason, two genetic algorithms are used to solve data sets with an order of 14 or more. Previous studies in the literature showed that OAS problems are non-deterministic polynomial-time hard (NP-Hard) problems when penalties for tardiness are considered. (Wu et al., 2018)

Two Genetic Algorithms are developed with the name GADOC (Genetic Algorithm with Davis Order Crossover) and GATPC(Genetic Algorithm with Two-Point Crossover) to handle the problem on the Fuzzy Order Acceptance and Scheduling On Identical Parallel Machine (FOAS-IPM). The proposed Genetic Algorithm are tested with small, medium, and large-sized dataset instances for comparing the performance of proposed Genetic Algorithms.

1. Algorithm: GADOC (Genetic Algorithm with Davis Order Crossover)

In the GADOC Algorithm, *the gene* represents each order received from the customer. *The chromosome* is the sequence of all received orders or in other words represents the order of all orders that will be produced. Each chromosome represents the sequence of all orders that will be produced. The *population* is a collection of multiple chromosomes. There are multiple chromosomes in a population that represent different order sequences from each other. Each of these chromosomes represents a possible solution for the FOAS-IPM problem for all machines. *The fitness function* is the total net revenue for each solution of each chromosome. GADOC is a population-based search consisting of five main operators: *Initialization, Selection, Crossover, Mutation, and Replacement*.

At the initialization of the GADOC, random numbers between 0 and 1 are generated. After these generated numbers were labeled from 1 to n (order number), these numbers were ordered in descending order and a random order sequence was obtained for the initialization of the GADOC algorithm. And these created numbers are named with different labeled numbers as described in figure 3.12.

5	6	3	2	1	7	8	9	10	4
0,9512	0,8427	0,7345	0,6979	0,6543	0,3428	0,2979	0,1749	0,0467	0,0056
m1	m1	m1	m1	m2	m2	m2	m3	m3	m3

Figure 3.12. Representing the initial sequence of a chromosome with n=10 and m=3

The initial population first was created with 2000 chromosomes. One of the main factors that also determine the initial population is the size of the order data set. Small initial populations were sufficient for small data sets, while large initial populations were used for large data sets. Thus, according to the size of the data set, the initial population was changed between 100 and 5000 in the study. Also, the effect of the initial population size on how to affect the result of the algorithm was examined. The assignment of the orders on two machines for data set with 10 orders is described in figure 3.13. The numbers of orders assign on each machine must be equal or as close as to each other. In figure 3.14, the data set with 10 orders assigned to 3 machines equally but the number of assigned orders on each machine cannot be equal to each other as in figure 3.12. But data set with 10 orders assigned to 5 machines and the number of orders can equally be assigned on each machine.

Order	4	7	1	10	3	9	6	8	2	5
Machine	m1	m1	m1	m1	m1	m2	m2	m2	m2	m2

Figure 3.13. The assignment of orders on the machine on a chromosome with m=2 and n=10

Order	4	7	1	10	3	9	6	8	2	5
Machine	m1	m1	m1	m1	m2	m2	m2	m3	m3	m3

Figure 3.14. The assignment of orders on the machine on a chromosome with m=3 and n=10

Order	4	7	1	10	3	9	6	8	2	5
Machine	m1	m1	m1	m2	m2	m2	m3	m3	m4	m4

Figure 3.15. The assignment of orders on the machine on a chromosome with m=4 and n=10

Order	4	7	1	10	3	9	6	8	2	5
Machine	m1	m1	m2	m2	m3	m3	m4	m4	m5	m5

Figure 3.16. The assignment of orders on the machine on a chromosome with m=5 and n=10

In this presented study, the data set orders are with 10, 15,20,25,50, and 100 orders and the machine sets are with 2,3,4, and 5 machines. For example, if the data set with 100 orders assigned on 3 machines, there must be 34 orders on the first machine, 33 orders on the second machine, and 33 orders on the third machine. So the first 34 orders on the chromosome of this example show the orders on the first machine, the second 33 orders on the same chromosome show the orders on the second machine and the last 33 orders on the same chromosome show the order on the third machine.

Suppose that n is the number of orders and m is the number of machines and $n=mxp+k$ where $n>m$, $n>k$, $n>p$ and $k<m$ where $n,m,p>0$ and $0\leq k$. Finally, the number of assigned n orders on each m machine must be at least p or at most p+1. So for all data set in the GADOC algorithm, this rule is implemented on each chromosome.

The selection process in the GADOC is the process of deciding on which of the chromosomes can pass to the next generation. Elitism is used to decide and pass the chromosomes from the existing population to the new population. The selection method is also crucial in the selection of chromosomes to enter the crossing and mutation processes. The elitist strategy is a selection method in the genetic algorithm

for selecting the best chromosomes in the population. In this study, elitism prevents the loss of good solutions found during the search process in the GADOC. In the population, chromosomes are ordered according to their fitness function value from the best to the worst. And half of the population becomes elite chromosomes and the other half becomes non-elite chromosomes in this population according to their fitness function value. Elite chromosomes can pass directly to the other new population. On the contrary, non-elite chromosomes can pass the other population after the crossover process. After the crossover process, new offspring must have a better fitness function value than their parents. If not, these offspring must pass the mutation process. The best parent in the elite division and the worst parent in the non-elite division are selected for the crossover operation. The previous individual from the second elite individual and the non-elite individual with the worst fitness function value, respectively, are selected for the crossover process. All elite and non-elite chromosomes are subjected to a cross-over process.

The crossover in the GADOC is the process of generating new springs by selecting the chromosomes from the existing population to create a new population. In this study, the best elite individual and the worst non-elite individual were selected to undergo a crossover process. This process continues by performing a crossover between the next best elite individual and the next worst non-elite individual, respectively. The process continues until all elite individuals and all non-elite individuals are crossed. All elite individuals pass directly to the new population. But the fitness functions of the newly formed springs must be better than the fitness functions of the old non-elite individuals. After the comparison process, it is decided whether the new individual will join the new population or not. If the fitness function of the newly formed spring is better than the old individual, the new individual is included in the new population. However, if the fitness function value of the newly formed individual is worse than the old individual, the newly formed individual is subjected to the mutation process and added to the new population.

In GADOC, for crossover Davis' Order Crossover operator is used in this study. Davis' Order Crossover for permutation-based crossovers to transmit information about relative ordering to the off-springs. Davis' Order Crossover works first with detecting two random crossover points in the elite parent and copies all the segments between them from the first parent to the first offspring. In GADOC, for increasing the randomization of the algorithm, the size of the segment between two random crossover points is random from the size of 1 order to 4 order. In figure 3.17, the size of two random points is selected in 4 order. After, starting from the second crossover point in the second parent, copy the remaining unused numbers from the second parent to the first child, wrapping around the list. So, with this crossover method, the sequence of at least 1 and at most 4 orders on the chromosome of the new offspring are the same as the elite parent.

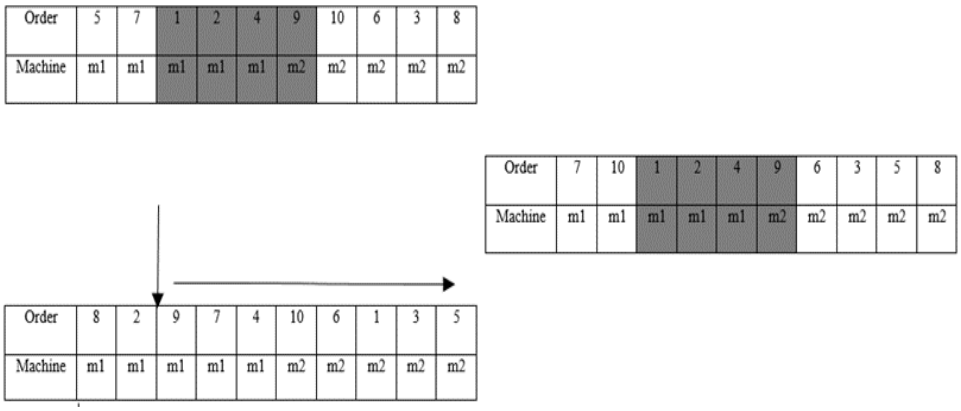


Figure 3.17. An example of the GADOC Crossover Process with n=10 and m=2

The mutation in the GADOC is used when the offspring are not better than their parents after the crossover process. The offsprings' fitness functions must be better than their non-elite parents. After this comparison process, it is decided whether the new individual will join the new population or it is mutated. In the GADOC, the Swap Mutation operator is used for the mutation process. The process is carried out by changing the places of the two chosen random orders on two random

spots on the offspring chromosome as described in figure 3.18. After this mutation process, a new offspring chromosome with a new different sequence is added to the new population.

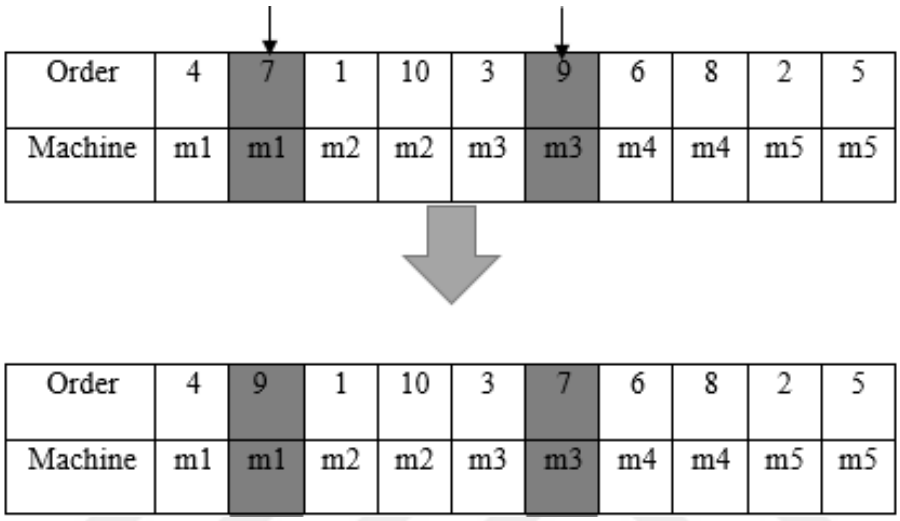


Figure 3.18. An example of the Swap Mutation in GADOC for n=10 and m=5

The Replacement is the final process of the GADOC. As described in figure 3.5, the elite chromosomes pass directly to the new population without any change. But non-elite chromosomes must be a crossover with the elite chromosomes. As a result of the cross-over process, if the new offspring have better fitness function value than non-elite individuals, they replace the non-elite individual in the new population. New offspring that are better than their non-elite parents can replace the new population after the mutation process. New offspring have worse fitness function than their non-elite parents can join the new population after the mutation process. Thus, by comparing the fitness function and increasing the diversity in the population with crossover and mutation, the probability of coming closer to the optimal solution for the new population is increasing.

GADOC Algorithms in this study can be defined as below:

```
{
  Generate initial Population P0
  Evaluate Population P0
  While (stopping GA criteria not satisfied) Repeat
            {
        Sort chromosomes descending order of FitFunc
        Find elite chromosomes (Half of the population)
        Add elite chromosomes to the new population
        Remove non-elite chromosomes
              Crossover (Create new chromosomes from elite and nonelite with
Davis Order Crossover)
        Evaluate new chromosomes
              If Fitfunc_newchromosome < Fitfunc_nonelitechromosome
              Mutation new chromosomes with swap mutation
              Add new chromosome to the new population
             Else
              Add new chromosomes to the new population
          }
}
```

Figure 3.19. General Puseodocode of GADOC Algorithms

2. Algorithm: GATPC (Genetic Algorithm with Two-Point Crossover)

In the GATPC Algorithm, as in GADOC*, the gene* represents the order and *the chromosome* represents the sequence of all received orders. The *population* is a collection of multiple chromosomes that have different order sequences from each other. Each of these chromosomes represents a possible solution for the FOAS-IPM problem for all machines. *The fitness function* is the total net revenue for each solution of each chromosome. GAMPC is a population-based search consisting of

five main operators: *Initialization, Selection, Crossover, Mutation, and Replacement*.

At the initialization of the GATPC, random numbers between 0 and 1 are generated and these generated numbers were labeled from 1 to n (order number) which are ordered in descending order and a random order sequence was obtained for the initialization of the GATPC algorithm. And these created random numbers which are named with different labeled numbers create a chromosome also that shows the randomized sequence of orders.

The initial population first was created generally with 2000 chromosomes. But the initial population depends on the size of the order and when the data set gets larger, the initial population number must increase too because of increasing the solution space. Small initial populations were sufficient for small data sets, while large initial populations were used for large data sets. Thus, according to the size of the data set, the initial population was changed between 100 and 5000 in the study as in GADOC. Also for both GADOC and GATPC algorithm, the effect of the initial population size investigated how to affect the result of the algorithm. The assignment of the orders on the machines for different sizes of data set with 10 orders is described in the GADOC algorithm

In this presented study, the data set orders are with 10, 15,20,25,50, and 100 orders, and the machine sets are with 2,3,4, and 5 machines are used for both algorithms for the performance test. The rule of data set assignment on machines is the same as GADOC. For example, if the data set with 50 orders assigned on 3 machines, there must be 17 orders on the first machine, 17 orders on the second machine, and 16 orders on the third machine. So the first 17 orders on the chromosome of this example show the orders on the first machine, the second 17 orders on the same chromosome show the orders on the second machine and the last 16 orders on the same chromosome show the order on the third machine.

Let's n is the number of orders and m is the number of machines and n=mxp+k where n>m, n>k, n>p and k<m where n,m,p>0 and 0≤k. Finally, the number of assigned n orders on each m machine must be at least p or at most p+1.

The selection process in the GATPC as in GADOC deciding on which of the chromosomes can pass to the next generation. Elitism is used for the selection The selection method is also crucial in the selection of chromosomes to enter the crossing and mutation processes. The elitist strategy is a selection method in the genetic algorithm for selecting the best chromosomes in the population. In the old population, chromosomes are ordered according to their fitness function value from the best to the worst. And half of the population becomes elite chromosomes and the other half becomes non-elite chromosomes in this population according to their fitness function value. Elite chromosomes can pass directly to the other new population. On the contrary, non-elite chromosomes can pass the other population after the crossover process. After the crossover process, new offspring are evaluated according to their fitness function value which compares with their non-elite parents. If not, these offspring must pass the mutation process. The best parent in the elite division and the worst parent in the non-elite division are selected for the crossover operation. The previous individual from the second elite individual and the non-elite individual with the worst fitness function value, respectively, are selected for the crossover process. All elite and non-elite chromosomes are subjected to the crossover process.

The crossover in the GADOC is the process of generating new springs by selecting the chromosomes from the existing population to create a new population. In this study, the best elite individual and the worst non-elite individual were selected to undergo a crossover process. This process continues by performing a crossover between the next best elite individual and the next worst non-elite individual, respectively. The process continues until all elite individuals and all non-elite individuals are crossed. All elite individuals pass directly to the new population. But the fitness functions of the newly formed springs must be better than the fitness

functions of the old non-elite individuals. After the comparison process, it is decided whether the new individual will join the new population or not. If the fitness function of the newly formed spring is better than the old individual, the new individual is included in the new population. But, if the fitness function value of the offspring is worse than the non-elite parent, these offspring are passed to the mutation process and added to the new population.

In GATPC, for crossover Two-Point Crossover operator is used in this study. There are different crossover operators for point crossover in the literature. (Paksoy, 2007) Some of them are Single-Point Crossover, Two-Point Crossover, Multi-Point Crossover, and Uniform crossover. In the single-point crossover, randomly one point is selected where the region's change is done between two chromosomes as described in figure 3.17. In the Two-Point crossover, randomly two points are selected where the region's change is done between two chromosomes as described in figure 3.18. In this presented study, In the Two-Point crossover, in addition to the 2 randomly chosen points, the width of the region to change between the two chromosomes was also randomly selected between 1 and 4 orders. In a two-point crossover, chromosome pairs are cut from two different places and each parent chromosome is divided into three parts. The crossing is made by exchanging the parts mutually. If a single piece of crossing is exchanged between two parents, two new chromosomes are obtained. If two randomly selected pieces are replaced by each other, four different new chromosomes are obtained. But due to elitism strategy selection, only half of the new offspring population can survive. With the Two Point Crossover method, 2 offspring are breeding from 2 parents. However, since half of the population consists of elite parents, only 1 of the 2 offspring has to be selected and join the new society instead of the non-elite parent. The one with a better fitness function value survives from 2 offspring. Those who have a bad fitness function value are eliminated like non-elite parents.

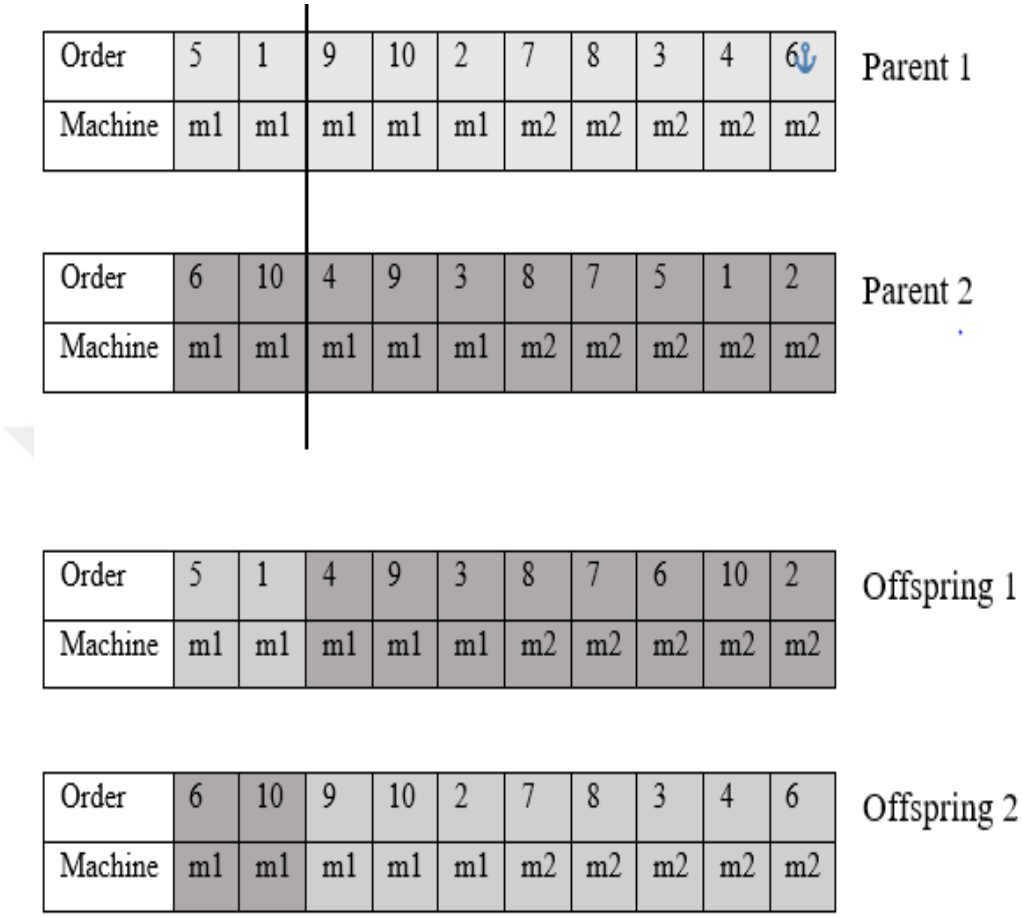


Figure 3.20. An example of One-Point Crossover for n=10 and m=2

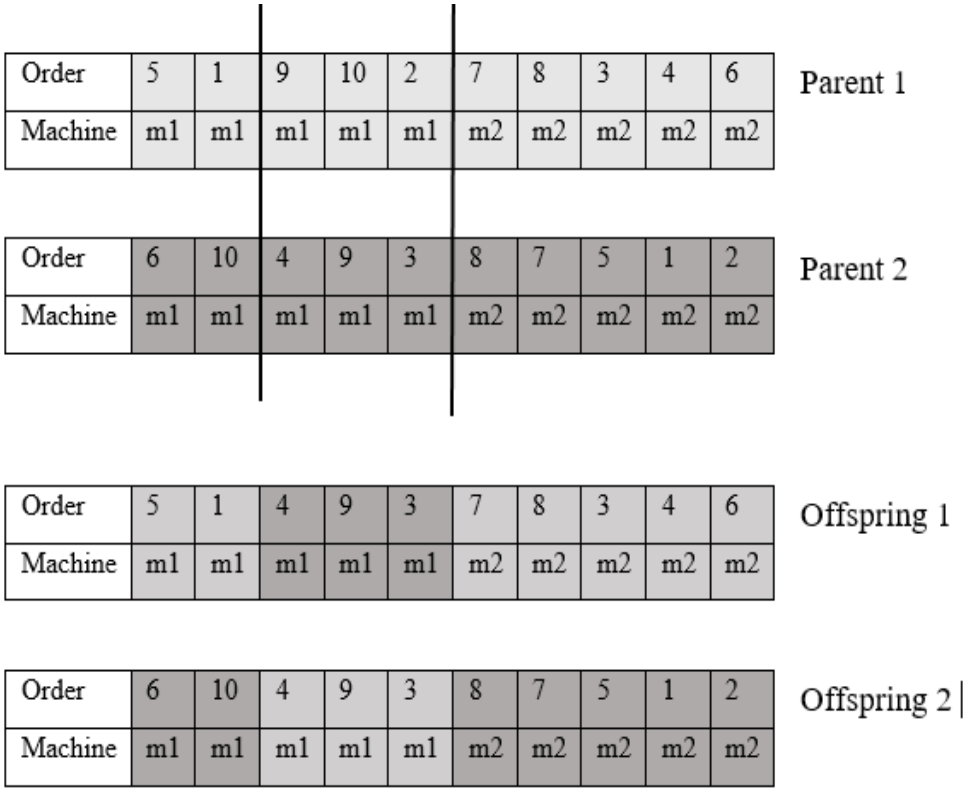


Figure 3.21. An example of Two-Point Crossover for n=10 and m=2 with the randomized region of 4 orders.

The mutation in the GATPC is used when the offspring are not better than their parents after the crossover process. The offspring's fitness functions must be better than their non-elite parents. After this comparison process, it is decided whether the new individual will join the new population or it is mutated. In the GATPC, the Swap Mutation operator is used for the mutation process. The process is carried out by changing the places of the two chosen random orders on two random spots on the offspring chromosome After this mutation process, a new offspring chromosome with a new different sequence is added to the new population.

The Replacement is the final process of the GATPC. The first replacement is for the elite chromosomes that can pass directly to the new population without any

change. But non-elite chromosomes must crossover with the elite chromosomes. As a result of the cross-over process, if the new offspring have better fitness function value than non-elite individuals, they replace the non-elite individual in the new population. New offspring that are better than their non-elite parents can replace the new population after the mutation process. New offspring have worse fitness function than their non-elite parents can join the new population after the mutation process. Thus, by comparing the fitness function and increasing the diversity in the population with crossover and mutation, the probability of coming closer to the optimal solution for the new population is increasing.

```
{
  Generate initial Population P0
  Evaluate Population P0
  While (stopping GA criteria not satisfied) Repeat
          {
     Sort chromosomes descending order of FitFunc
     Find elite chromosomes (Half of the population)
     Add elite chromosomes to the new population
     Remove non-elite chromosomes
     Crossover (Create new chromosomes from elite and nonelite with Two
   Point Crossover)
     Evaluate new chromosomes
          If Fitfunc_newchromosome < Fitfunc_nonelitechromosome
          Mutation new chromosomes with swap mutation
          Add new chromosome to the new population
         Else
          Add new chromosomes to the new population
       }
}
```

Figure 3.22. General Puseodocode of GATPC Algorithms

The following steps have been applied for the GADOC and GATPC genetic algorithms used in this study.

Step1. Create a new chromosome

Step1.1.Create a chromosome random numbers between [0,1] as many as order number n

Step1.2.Label random numbers between 1 and n. Assigned orders to the machines.

Step1.3.Order these numbers in descending sequence and get a random sequence of orders.

Step2.Repeat step1. until reaching the population size.

Step3.Calculate the fitness function of a chromosome

Step3.1. Group the orders on a chromosome according to their machine-id. Create a list of orders on each machine.

Step3.2. Take the first order i in the sequence on each machine and calculate the completion time C_i

If the sequence of order i on a machine=1, $C_i=C_i$

Else $C_i=C_i+C_{i-1}$

Step3.2. Calculate the Fuzzy Due Date D_i

Step3.3. Compare C_i and D_i

Step3.4. If $C_i<D_i$, $FitnessFunction_i=E_i$, Add order i to the accepted order list (E_i is the revenue of order i

Else if $C_i=D_i$, $FitnessFunction_i=E_i$, Add order i to the accepted order list

Else if $C_i>D_i$

$T_i=C_i-D_i$

If $(E_i- T_i*w_i)\geq 0$, $FitnessFunction_i= E_i- T_i*w_i$, Add order i to the accepted order list (w_i is the weighted tardiness of order i)

Else

Remove the order i to the outsourced order list and go back to Step 3.1

FitnessFunction$_{=}$ (FitnessFunction$_{i+}$ FitnessFunction$_{i-1}$)$_{machine1}$+……..(FitnessFunction$_{i+}$ FitnessFunction$_{i-1}$)$_{machinen}$

Stop when the orders on the chromosome are finished.

Step4.Order these chromosomes in descending order according to their fitness function

Step5.Label half of the population as Elite that has better Fitness Function and the other half Non-Elite that have worse Fitness Function among the chromosomes. Add Elite chromosomes to the new population.

Step6. Crossover the elite chromosomes from the best to the worst with the non-elite chromosomes from the worst to the best respectively.

Step7. Compare the FitnessFunction of Offspring and Non-Elite.

If FitnessFuntion(Non-elite parent)<= FitnessFuntion(Offspring)

Add Offspring chromosome to the new population.

Else do swap mutation to the Offspring chromosome.

Add Offspring chromosome to the new population.

Stop when the Offspring chromosomes are finished.

Step8. Return to Step3.

Stop when the maximum iteration is reached.

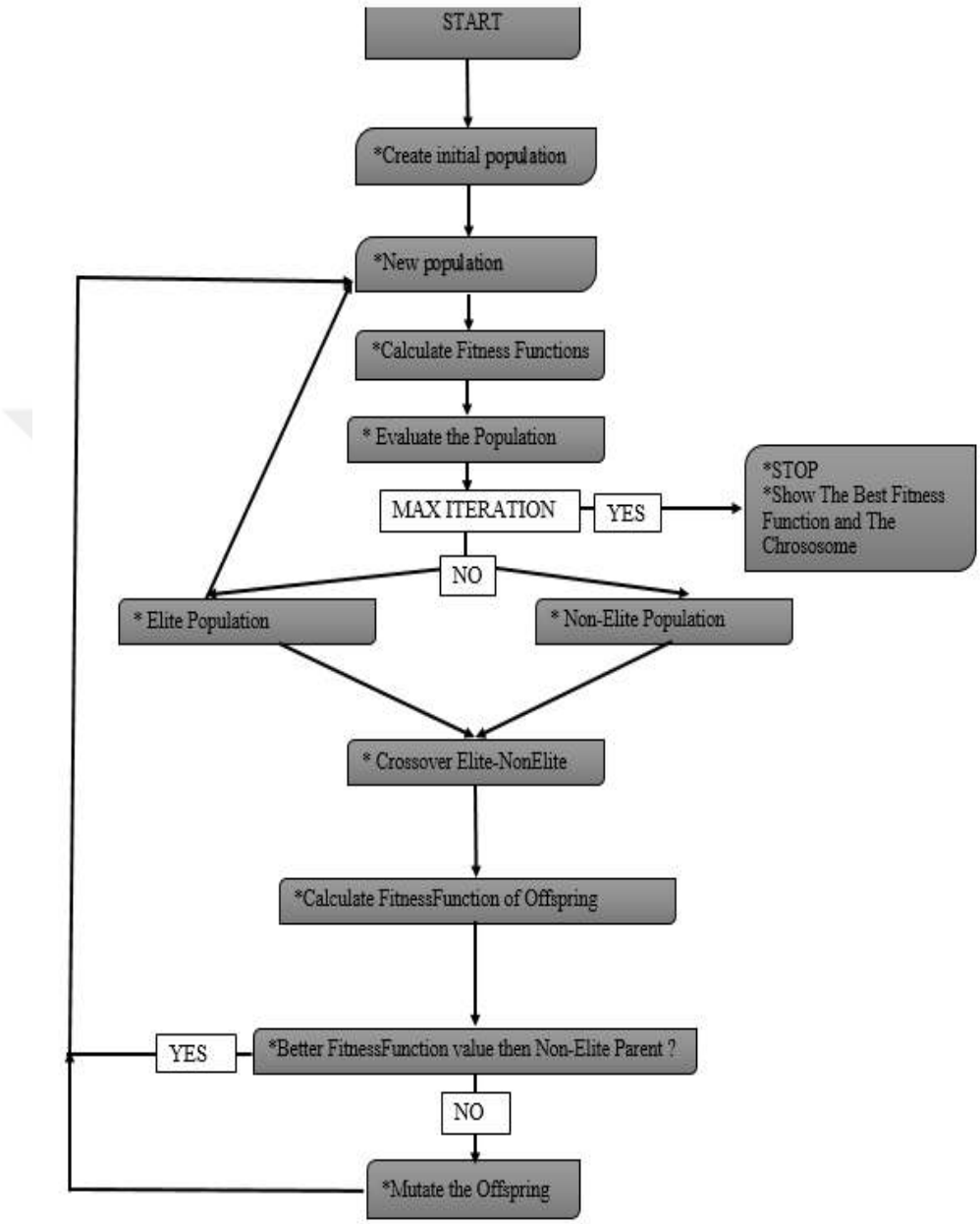


Figure 3.23. Flow chart of GADOC and GATPC Algorithms

4. RESULT AND DISCUSSION

In this section, the solutions obtained with data sets as a result of analyzing the mathematical models and heuristic algorithms developed are evaluated. In this study, Fuzzy Order Acceptance and Scheduling Problem on Identical Parallel Machines (FOAS-IPM) is examined. The objective of the problem is to find out the best schedule to have a maximum net profit. The main constraint in this study is the due dates of the orders. The completion time of orders must be earlier than their due dates or the differences between revenue and discount of delay punishment that is net profit must be at least zero. Otherwise, the company must outsource the product to deliver the product to the customer at least due to date in order not to lose the customer and not to cause a bad reputation on the delivery date to gain the customer loyalty. However, outsourcing can cause a decrease in net profit due to differences between companies in pricing policies. To have a maximum profit, it is better to accept more orders in the production schedule.

4.1. Experiment Evaluation

To solve the mathematical model of FOAS-IPM problems with the computer base GAMS (The General Algebraic Modeling System) program is used. The GAMS program is an optimization program to solve especially Linear, Non-Linear, Mix Integer Linear Optimization problems in Industrial Engineering subjects. This program helps to model and solve optimization problems with its excellent modules. To solve the optimization problems with the program GAMS, the model must be defined and apply carefully to program coding language. The program GAMS offers different kinds of solvers. In this study, the CPLEX solver is used to hand the FOAS-IPM problem in the GAMS program.

Since the order data sets and machine numbers are variable, the mathematical model is also differentiated to be able to test each machine and order data set. For this reason, mathematical models were written in 4 differentiated codes

in GAMS programs so that 10 different order data set sizes can be tested for the systems with 2, 3, 4, and 5 machines. Each of these written models has been tested with 5 data sets. This model has been tested with 5 data sets for each machine and 20 different cases have been examined in the GAMS program.

GADOC and GATPC algorithms developed using genetic algorithms were coded using the Matlab program. Matlab (Matrix Laboratory) program is an advanced matrix-based programming platform developed by Math works Company to generate codes and work more easily in engineering studies. Writing code for numerical analysis, modeling, and algorithm development in the Matlab program can be performed more easily than the other programs. Being matrix-based makes it very useful to code engineering problems. Since order data sets and machine numbers are variable, algorithms must be differentiated to be able to test each machine and order data set. For this reason, 24 differentiated sub-algorithms were written to test for 6 different order data set sizes with 10, 15,20,25,50, and 100 with 2, 3, 4, and 5 identical parallel machine systems. Likewise, the algorithm in GATPC has been differentiated as sub-algorithms and 24 different sub-algorithms have been written. Each execution result was taken with a Microsoft Excel File as shown in Appendix C.

The solving of data sets belongs to the FOAS-IPM problem is examined on a computer 64 bit with Intel Core i5 7200 CPU 2.71 GHz and 8 RAM.

To be able to analyze the performance of the methods developed for the FOAS-IPM problem, small, medium, and large-size instances are created. The features and problems of the developed data sets have been explained in the previous sections under the topic of parameters. Since this problem has not been solved before in the literature, the data sets have been redeveloped for this study. The mathematical model MILP developed for the solution of the problem could only provide solutions up to 14 data sets for each 2, 3, 4, and 5 identical parallel machine system. Therefore, with the program GAMS, small datasets could be solved for each machine number. (n = 10 and where m = 2, 3, 4 and 5) Two algorithms GADOC and GATPC, have

been developed with the Genetic Algorithm, by which the results obtained from the mathematical model developed MILP can be compared. The first algorithm is the GADOC (Genetic Algorithm Davis Order Crossover) using the Davis Order Crossover method. The second algorithm is the GATPC (Genetics Algorithm Two Point Crossover). Solution performance of GADOC and GATPC algorithms developed with small (10, 15), medium (20, 25), and large (50,100) order number data sets were compared. For small-size data sets with 10 orders, 3 developed solution methods can be compared. However, data sets with medium and large order numbers were solved with the developed GADOC and GATPC methods and the obtained results from two algorithms were compared for testing their performance.

The most important criterion in performance comparison is the deviation from the upper bound. The deviation from the upper bound is calculated as the difference between the total revenue of all orders in the dataset and the results of the program divided by the total revenue of all orders into the data set. In formulation 4.1 *Upper Bound* is the sum of the revenues of all orders in each data set.

$$UpperBoundDeviation(UBD) = \frac{Upper\ Bound - Solution}{Upper\ Bound} \quad (4.1)$$

The most important thing for the firm is the maximization of the profit to be obtained. Therefore, the program with the smallest UBD is the most successful and the best in solving this problem. Besides UBD, the solution time is also very important. The most significant factors affecting the solution time for heuristic algorithms are the initial population and iteration number. All of these cases are discussed in this section.

4.1.1. Results for Small-Sized Instances (n=10, n=15) on 2, 3, 4, and 5 Identical Parallel Machines

In this section, the upper bound deviation and execution time results of MILP, GADOC, and GATPC performance of small data sets with 10 orders are mentioned. Since 15 data sets cannot be solved with MILP in the CPLEX algorithm, the upper bound deviation and execution time comparison result of GADOC and GATPC algorithm solution performance of data sets with 15 orders are compared to analyze how far the solution methods deviated from the upper bound and solution times were compared.

4.1.1.1. Results for 10 Order Sized Instances on 2, 3, 4, and 5 Identical Parallel Machines

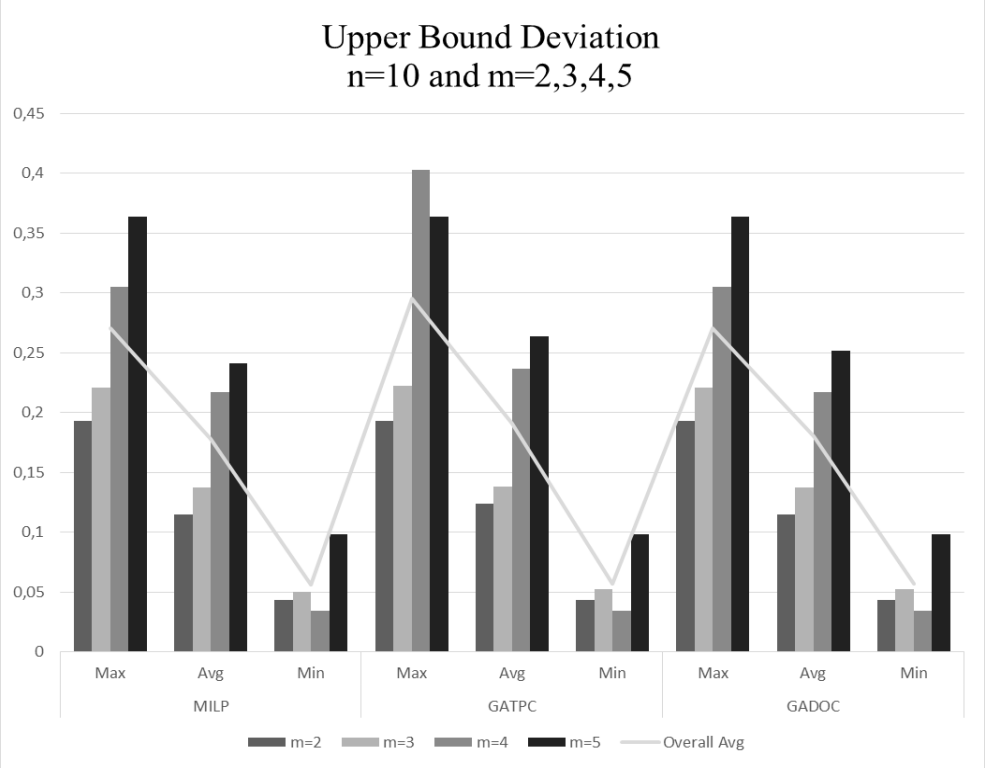
In this section, the upper bound deviation and solution times obtained by solving the data sets developed for systems with 10 orders on 2, 3, 4, and 5 identical parallel machines with 3 different methods are compared. 5 different data sets were created for each machine number. The data sets with 10 orders were solved with the MILP and the CPLEX algorithm in the GAMS program. By solving these data sets, optimum order scheduling and maximum profit results were obtained. Upper Bound Deviation was calculated by proportioning the maximum profit result obtained with the total profit to be obtained when all orders are accepted. Besides, the execution times of solving the 10 order data set with the MILP in the CPLEX algorithm were compared. The desired result is lower upper bound deviation and execution time. Lower upper bound deviation means the result is much closed to the total revenue and lower execution time means is solving the problem fast. The maximum, minimum, and average deviation of the results were calculated and evaluated. The values obtained as a result of the solution are shown in table 4.1. and also in graph 4.1.

Table 4.1. Upper Bound Deviation Performance of MILP, GATPC, and GADOC Algorithm for 10 orders on 2, 3, 4, and 5 identical Parallel Machine

UBD		MILP			GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=10	m=2	0,193	0,115	0,043	0,193	0,124	0,043	0,193	0,115	0,043
	m=3	0,221	0,137	0,05	0,222	0,138	0,052	0,221	0,137	0,052
	m=4	0,305	0,217	0,034	0,403	0,237	0,034	0,305	0,217	0,034
	m=5	0,364	0,241	0,098	0,364	0,264	0,098	0,364	0,252	0,098
Overall Avg		0,27075	0,178	0,05625	0,2955	0,19075	0,05675	0,27075	0,18025	0,05675

In the solution of the CPLEX algorithm for MILP, the deviation of the 5 data sets created for the system with 2 identical parallel machines is 0.115, 0.114 for the GATPC algorithm, and 0.115 for the GADOC algorithm. With the MILP, the maximum deviation from UBD is the same amount as GADOC and GATPC and it is 0.193, and likewise, the minimum deviation values are the same and 0.043. With a mean deviation of 0.115, MILP and GADOC showed the same solution finding performance for 10 orders on 2 identical parallel machines. However, GATPC's performance is lower with an average deviation of 0.124. On 3 identical parallel machines, MILP and GADOC have the same solving performance with an average of 0,137 upper bound deviations. GATPC has a very close upper bound deviation average of 0,138. On 4 identical parallel machines, GADOC and MILP have the same solving performance with 0,217 upper bound deviations but GATPC is not as good as with the other methods. On 5 identical parallel machines, MILP has the best solving performance with an average of 0.241 upper bound deviations. The second-best performance belongs to GADOC and the algorithm with the worst performance is GATPC. When the average upper bound deviation of the algorithm compares for all machine groups, MILP has the best and the lowest upper bound deviation. After the MILP, the second-best algorithm performance with an average deviation value of 0.180 is the GADOC. The worst algorithm performance is the GATPC with a 0.191 upper bound deviation average for all machine groups. Also, the maximum

upper bound deviation average is 0,296 for GATPC which is 0,271 for the GADOC and MILP. (Graph 4.1.2.1.1)

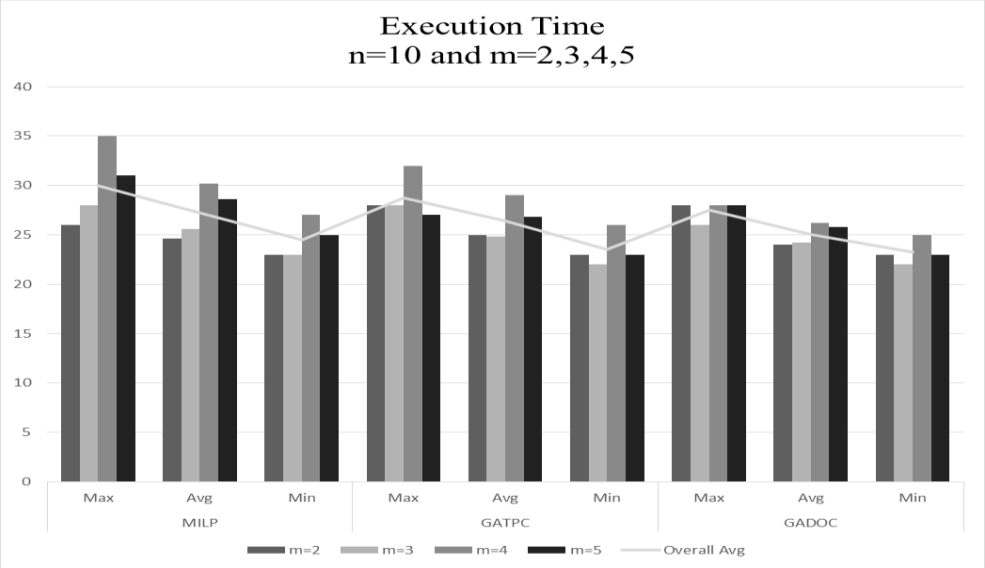


Graph 4.1. Upper Bound Deviation Performance of MILP, GATPC, and GADOC Algorithm for 10 orders on 2, 3, 4, and 5 Identical Parallel Machine

Table 4.2. Execution Time Performance of MILP, GATPC, and GADOC Algorithm for 10 orders on 2, 3, 4, and 5 Identical Parallel Machine

Execution Time (sec)		MILP			GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=10	m=2	26	24,6	23	28	25	23	28	24	23
	m=3	28	25,6	23	28	24,8	22	26	24,2	22
	m=4	35	30,2	27	32	29	26	28	26,2	25
	m=5	31	28,6	25	27	26,8	23	28	25,8	23
Overall Avg		30	27,25	24,5	28,75	26,4	23,5	27,5	25,05	23,25

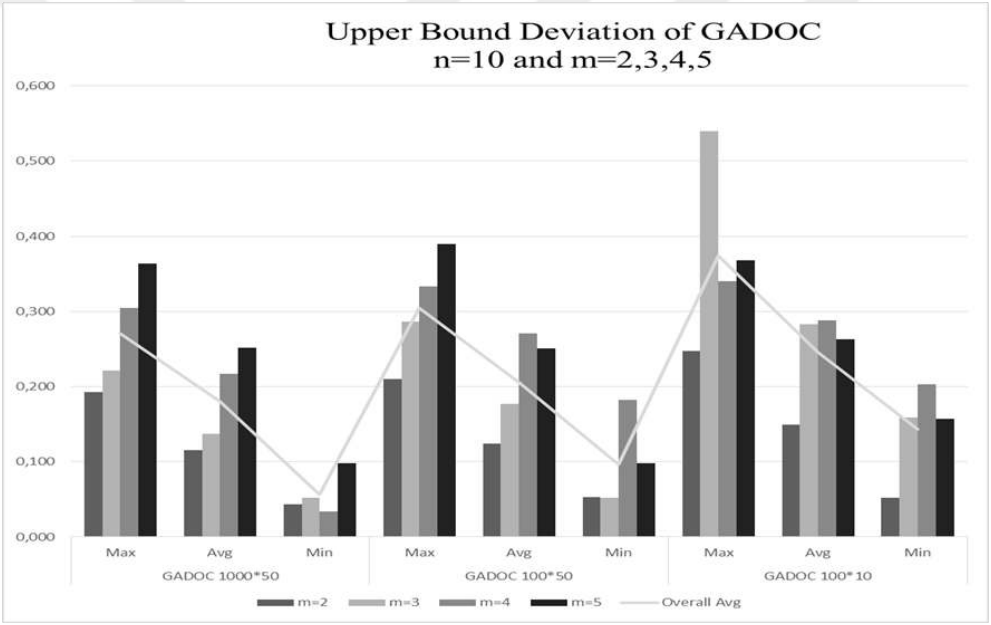
MILP has solved 10 order data sets regardless of the number of machines with an average of around 24 thousand iterations with the CPLEX algorithm. For GADOC and GATCP Algorithm, this value was determined as 1000 initial population and 50 iterations, and these algorithms were run only 1 time each. The total number of solutions scanned for GADOC and GATPC is 50000. It is explained how the number of iterations and the starting population number are determined under the title of execution performance of GADOC. The execution time of the CPLEX algorithm has been resolved in an average of 24, 6 seconds for 10 orders on 2 machines. While the GATPC algorithm solves in an average of 25 seconds, the GADOC algorithm has found a solution in an average of 24 seconds. GATPC and GADOC algorithms were run only for one recurrence but the probability of finding the correct result is increased with the high initial population and iterations.



Graph 4.2. Execution Time Performance of MILP, GATPC, and GADOC Algorithm for 10 orders on 2, 3, 4, and 5 Identical Parallel Machine

Table 4.3. Upper Bound Deviation Performance of the GADOC Algorithm for 10 orders on 2, 3, 4, and 5 Identical Parallel Machine with Different Initial Population and Iteration Number

UBD		GADOC 1000*50			GADOC 100*50			GADOC 100*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=10	m=2	0,193	0,115	0,043	0,201	0,128	0,053	0,247	0,149	0,052
	m=3	0,221	0,137	0,052	0,286	0,177	0,052	0,540	0,283	0,159
	m=4	0,305	0,217	0,034	0,333	0,271	0,182	0,340	0,288	0,203
	m=5	0,364	0,252	0,098	0,390	0,257	0,098	0,368	0,263	0,157
Overall Avg		0,271	0,180	0,057	0,303	0,208	0,096	0,374	0,246	0,143



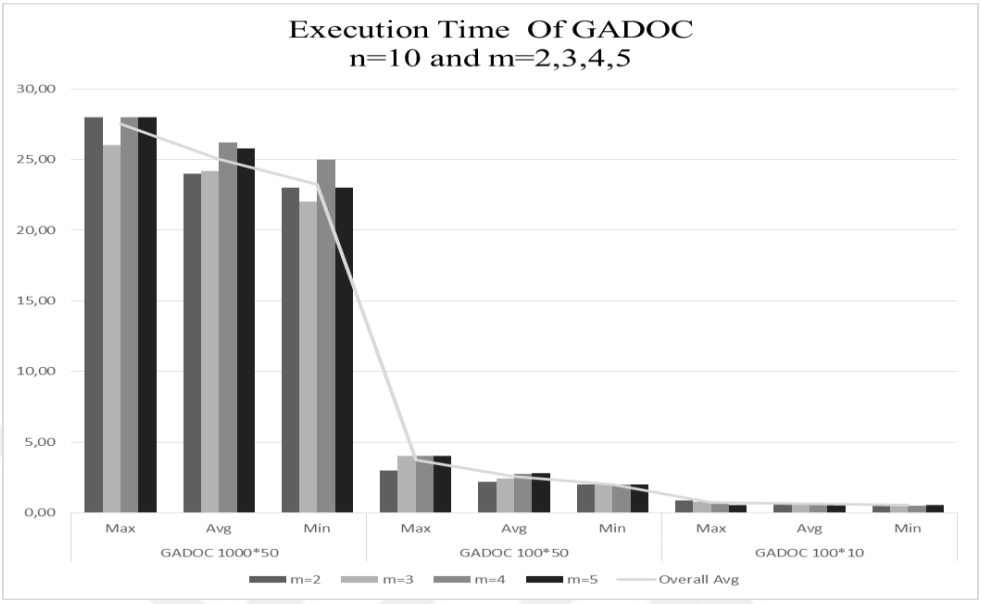
Graph 4.3. Upper Bound Deviation Performance of The GADOC Algorithm for 10 orders on 2, 3, 4, and 5 Identical Parallel Machine with Different Initial Population and Iteration Number

The GADOC showed the best performance among Genetic Algorithms. Therefore, when the starting population and the number of iterations were changed, the performance of the GADOC algorithm was compared according to the Upper Bound Deviation (UBD) and the solution time value. When the initial population is

decreased from 1000 to 100 and the number of iterations is kept constant at 50, the average of UBD for all number of machine groups is increased from 0,180 to 0,208. When the initial population stays constant with the value 100 and the number of iterations is decreased from 50 to 10, the average of UBD for all number of machine groups is increased from 0,208 to 0,246. In general, when the table is examined, it has been observed that the min and max values gradually increase like the average values with the decrease of initial and iteration numbers. It is seen that only in some cases the max and min values remain constant at close values due to randomness in the genetic algorithm.

Table 4.4. Execution Time Performance of the GADOC Algorithm for 10 orders on 2, 3, 4, and 5 Identical Parallel Machine with Different Initial Population and Iteration Number

Execution Time (sec)		GADOC 1000*50			GADOC 100*50			GADOC 100*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=10	m=2	28,00	24,00	23,00	3,00	2,20	2,00	0,85	0,70	0,56
	m=3	26,00	24,20	22,00	4,00	2,40	2,00	0,75	0,61	0,54
	m=4	28,00	26,20	25,00	4,00	2,75	2,00	0,68	0,60	0,51
	m=5	28,00	25,80	23,00	4,00	2,80	2,00	0,54	0,65	0,54
Overall Avg		27,50	25,05	23,25	3,75	2,54	2,00	0,71	0,64	0,54



Graph 4.4. Execution Time Performance of The GADOC Algorithm for 10 orders on 2, 3, 4, and 5 Identical Parallel Machine with Different Initial Population and Iteration Number

When the execution time performance of the GADOC Algorithm is examined for 10 order instances on 2, 3, 4, and 5 Identical Parallel Machine with Different Initial Population and Iteration Number, the execution time performance increases. The GADOC algorithm finds the solution in short times. When the initial population is decreases from 1000 to 100 and the iteration number stays constant at 50, the average execution time decreases from 25, 05 to 2, 54. And when the initial population stays constant at 100, and the iteration number decreased from 50 to 10, the execution time goes down from 2, 54 to 0,64. When the execution time is examined, it is observed that in no case the max, min, and avg values are not the same and there is a clear decrease.

4.1.1.2. Results for 15 Order Sized Instances on 2, 3, 4, and 5 Identical Parallel Machines

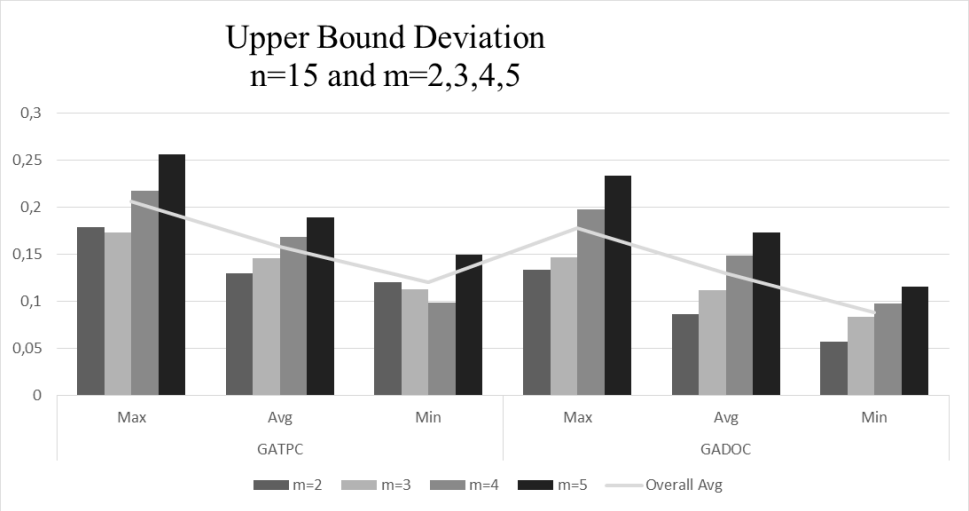
In this section, the upper bound deviation and solution times obtained by solving the data sets developed for systems with 15 orders on 2, 3, 4, and 5 identical parallel machines with 2 different methods The GADOC and GATPC are compared. The desired result is lower upper bound deviation and lower execution time. Lower upper bound deviation means the result is much closed to the total revenue and lower execution time means is solving the problem fast. 5 different data sets were created for each machine number. The data sets with 15 orders can only be solved with the GADOC and GATPC Algorithm.15 and larger instances cannot be solved with the CPLEX algorithm. By solving these data sets, optimum order scheduling and maximum profit results were obtained. Upper Bound Deviation was calculated by proportioning the maximum profit result obtained with the total profit to be obtained when all orders are accepted. Besides, the execution times of solving the 15 order data set with the GADOC and the GATPC. The maximum, minimum, and average deviation of the results were calculated and evaluated. The values obtained as a result of the solution are shown in table 4.1.2.2.1 and also in graph 4.2.1.2.1

Table 4.5. Upper Bound Deviation Performance of GATPC, and GADOC Algorithm for 15 orders on 2, 3, 4, and 5 identical Parallel Machine

UBD Deviation		GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=15	m=2	0,179	0,13	0,121	0,134	0,087	0,057
	m=3	0,173	0,146	0,113	0,147	0,112	0,084
	m=4	0,218	0,169	0,099	0,198	0,149	0,098
	m=5	0,256	0,189	0,15	0,234	0,173	0,116
Overall Avg		0,207	0,159	0,121	0,178	0,130	0,089

In the table above, the solution of the GADOC algorithm for 15 order dataset on 2 identical parallel machines, the average upper bound deviation of the 5 data sets is 0.087, however, 0.13 for the GATPC algorithm. For the same system, the

maximum upper bound deviation of the GADOC algorithm is 0,134 and the GATPC is 0.179. Also, the minimum deviation values are 0,057 for the GADOC and 0.043 for GATPC. Thus, it is seen that the GADOC algorithm has performed better than the GATPC algorithm. On 3 identical parallel machines, the solving performance of the GADOC is an average of 0,112 upper bound deviations and for the GATPC is 0,146. On 4 identical parallel machines, the GADOC has the solving performance with 0,149 average upper bound deviations but GATPC has 0,169. On 5 identical parallel machines, GADOC has the better solving performance with an average of 0.173 upper bound deviations. The GATPC has the worst performance is 0,189 average upper bound deviation. When the average upper bound deviation of the algorithm compares for all machine groups, the GADOC has the best and the lowest upper bound deviation. Also for the maximum and the minimum upper bound deviation for all machine number groups, The GADOC performed better by far than the GATPC with average upper bound deviation for all machine groups respectively 0,130 and 0,159 and also with maximum and minimum upper bound deviation also described in the graph below.

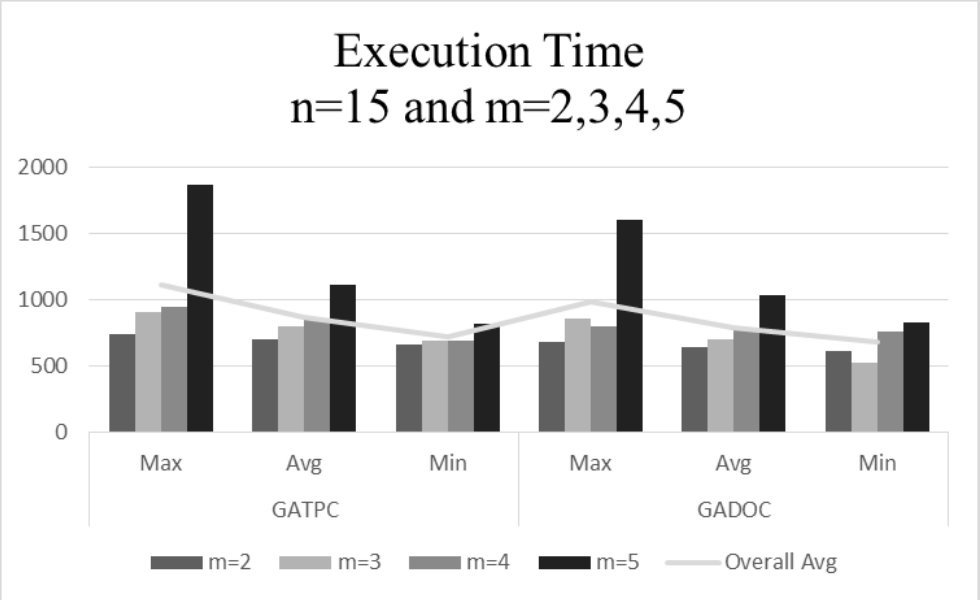


Graph 4.5. Upper Bound Deviation Performance of GATPC, and GADOC Algorithm for 15 orders on 2, 3, 4, and 5 identical Parallel Machine

Table 4.6. Execution Time Performance of GATPC, and GADOC Algorithm for 15 orders on 2, 3, 4, and 5 identical Parallel Machine

Execution Time (sec)		GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=15	m=2	740	696,8	653	678	635,8	610
	m=3	902	798,2	687	855	694	525
	m=4	947	844,8	691	798	772,2	758
	m=5	1862	1112	817	1600	1029,4	829
Overall Avg		1112,75	862,95	712,00	982,75	782,85	680,50

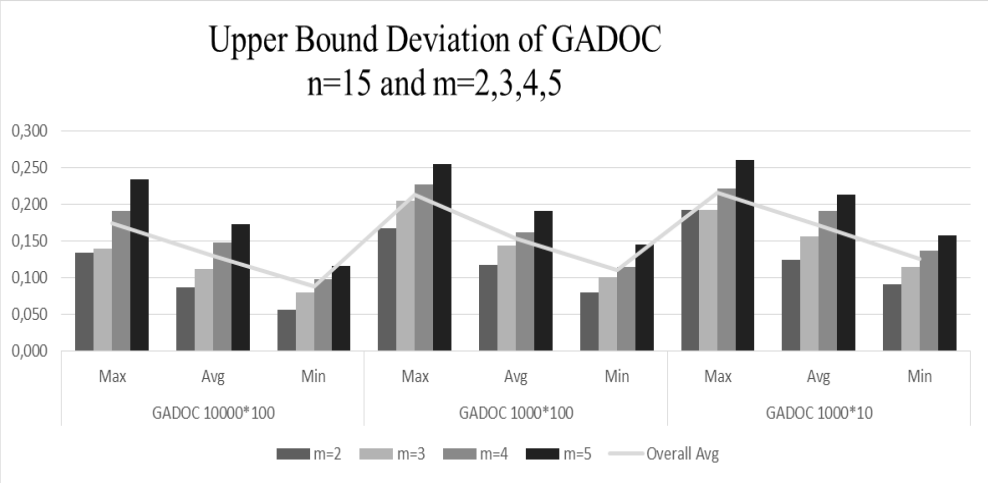
The GADOC has solved 15 order data sets regardless of the number of machines with an average of 782 seconds and for the GATPC is 862 second. For GADOC and GATCP Algorithm, initial population 10000 and 100 iteration number was determined and these algorithms were run only 1 time each. The total number of solutions scanned for GADOC and GATPC is 1000000. It is explained how the number of iterations and the starting population number are determined under the title of execution performance of GADOC. The execution time of the GADOC algorithm has been resolved in an average of 635 seconds for 15 orders on 2 machines, 694 seconds on 3 machines, 772 seconds on 4 machines, and 1029 seconds on 5 machines. The execution time of the GADOC algorithm has been resolved in an average of 696 seconds for 15 orders on 2 machines, 798 seconds on 3 machines, 844 seconds on 4 machines, and 1112 seconds on 5 machines. The GATPC and GADOC algorithms were run only for one recurrence but the probability of finding the correct result is increased with as high as possible initial population and iterations.



Graph 4.6. Execution Time Performance of GATPC, and GADOC Algorithm for 15 orders on 2, 3, 4, and 5 identical Parallel Machine

Table 4.7. Upper Bound Deviation Performance of the GADOC Algorithm for 15 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

UBD		GADOC 10000*100			GADOC 1000*100			GADOC 1000*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=15	m=2	0,134	0,087	0,057	0,167	0,117	0,080	0,192	0,124	0,091
	m=3	0,140	0,112	0,080	0,205	0,143	0,101	0,192	0,156	0,115
	m=4	0,190	0,148	0,098	0,227	0,161	0,114	0,221	0,191	0,137
	m=5	0,234	0,173	0,116	0,255	0,190	0,145	0,260	0,213	0,157
Overall Avg		0,175	0,130	0,088	0,214	0,153	0,110	0,216	0,171	0,125

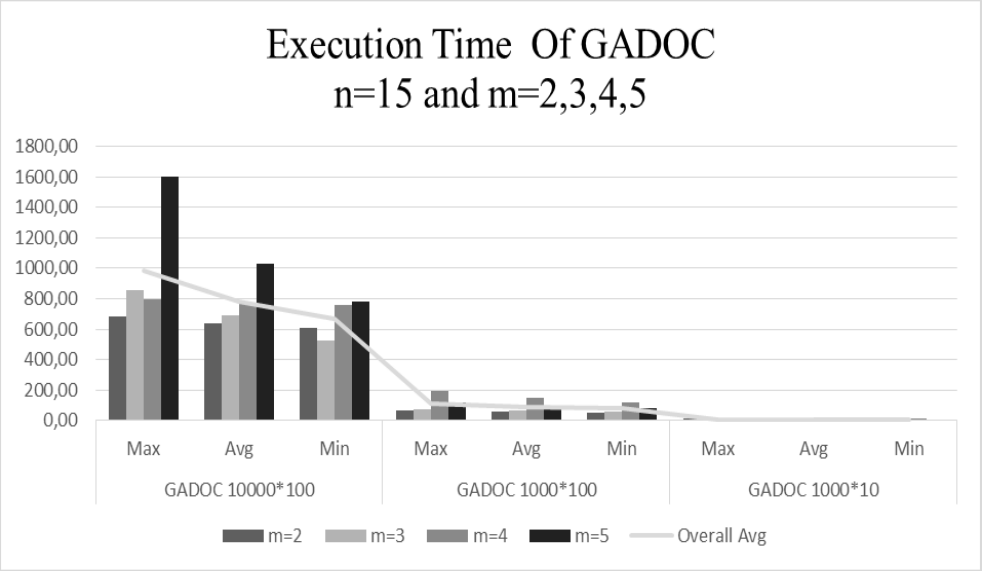


Graph 4.7. Upper Bound Deviation Performance of The GADOC Algorithm for 15 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

The GADOC showed the best performance among Genetic Algorithms. Therefore, when the starting population and the number of iterations were changed, the performance of the GADOC algorithm was compared according to the Upper Bound Deviation (UBD) and the solution or execution time value. When the initial population is decreased from 10000 to 1000 and the number of iterations is kept constant at 100, the average of UBD for all number of machine groups is increased from 0,130 to 0,153. When the initial population stays constant with the value 1000 and the number of iterations is decreased from 100 to 10, the average of UBD for all number of machine groups is increased from 0,153 to 0,171. In general, when the table is examined, it has been observed that the minimum and the maximum upper bound deviation values in all numbers of machines gradually increase like the average values with the decrease of initial and iteration numbers. It is seen also that the maximum and minimum upper bound deviation values increase since the number of searches in the solution space of the problem decrease.

Table 4.8. Execution Time Performance of the GADOC Algorithm for 15 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

Execution Time (sec)		GADOC 10000*100			GADOC 1000*100			GADOC 1000*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=15	m=2	687,00	635,80	610,00	68,00	59,00	54,00	12,00	9,80	9,00
	m=3	855,00	694,00	525,00	73,00	65,60	62,00	9,00	7,20	6,00
	m=4	798,00	772,20	758,00	192,00	147,20	119,00	7,00	9,60	13,00
	m=5	1600,00	1029,40	779,00	117,00	92,40	83,00	8,00	8,00	8,00
Overall Avg		985,00	782,85	668,00	112,50	91,05	79,50	9,00	8,65	9,00



Graph 4.8. Execution Time Performance of The GADOC Algorithm for 15 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

When the execution time performance of the GADOC Algorithm is examined for 15 order instances on 2, 3, 4, and 5 identical parallel machines with different initial populations and iteration numbers, the execution time performance increases. The GADOC algorithm finds the solution in short times. When the initial population is decreases from 10000 to 1000 and the iteration number stays constant

at 100, the average execution time decreases from 782 to 91. And when the initial population stays constant at 100, and the iteration number decreased from 100 to 10, the execution time goes down from 91 to 8. When the execution time is examined, it is observed that in no case the maximum, minimum, and average values are not the same and there is a clear decrease in execution time.

4.1.2. Results for Medium-Sized Instances (n=20, n=25)

In this section, the upper bound deviation and execution time results of GADOC, and GATPC performance of medium-size data sets with 20 and 25 orders are mentioned. Since 15 order and larger size data sets cannot be solved with MILP in the CPLEX algorithm, the upper bound deviation and execution time comparison result of GADOC and GATPC algorithm solution performance of data sets with 20 and 25 orders are compared to analyze how far the solution methods deviated from the upper bound, which algorithm perform better and solution times were compared.

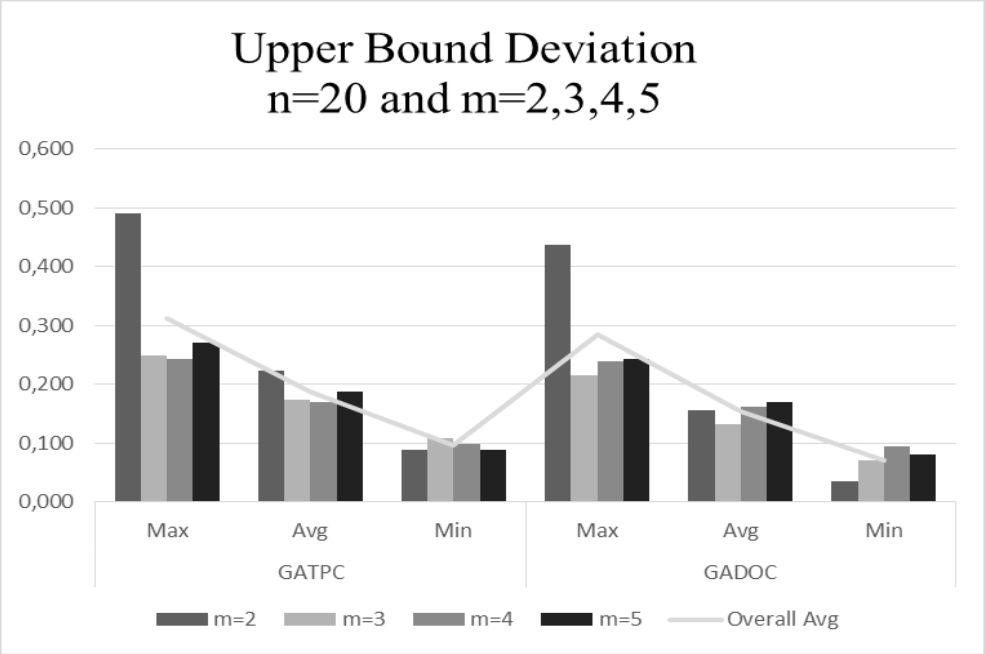
4.1.2.1. Results for 20 Order Sized Instances on 2, 3, 4, and 5 Identical Parallel Machines

Table 4.9. Upper Bound Deviation Performance of GATPC, and GADOC Algorithm for 20 orders on 2, 3, 4, and 5 Identical Parallel Machine

UBD		GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=20	m=2	0,491	0,223	0,088	0,437	0,156	0,035
	m=3	0,248	0,174	0,108	0,216	0,132	0,070
	m=4	0,243	0,170	0,098	0,239	0,161	0,094
	m=5	0,270	0,187	0,089	0,243	0,169	0,081
Overall Avg		0,313	0,189	0,096	0,284	0,155	0,070

In the table above, the solution of the GADOC algorithm for 20 order datasets on 2 identical parallel machines, the average upper bound deviation of the

5 data sets is 0.155, however, 0.189 for the GATPC algorithm. For the same system, the maximum upper bound deviation of the GADOC algorithm is 0,437 and the GATPC is 0.491. Also, the minimum deviation values are 0,035 for the GADOC and 0.088 for GATPC. Thus, it is seen that the GADOC algorithm has performed better than the GATPC algorithm on 2 identical parallel machines. On 3 identical parallel machines, the solving performance of the GADOC is an average of 0,132 upper bound deviations and for the GATPC is 0,174. On 4 identical parallel machines, the GADOC has the solving performance with 0,161 average upper bound deviations but GATPC has 0,170. On 5 identical parallel machines, GADOC has the better solving performance with an average of 0.155 upper bound deviations. The GATPC performance is 0,189 average upper bound deviation. When the average upper bound deviation of the algorithm compares for all machine groups, the GADOC has the best and the lowest upper bound deviation. Also for the maximum and the minimum upper bound deviation for all machine number groups, The GADOC performed better by far than the GATPC with average upper bound deviation for all machine groups respectively 0,155 and 0,189, and also described with maximum and minimum upper bound deviation and also described in the graph below.



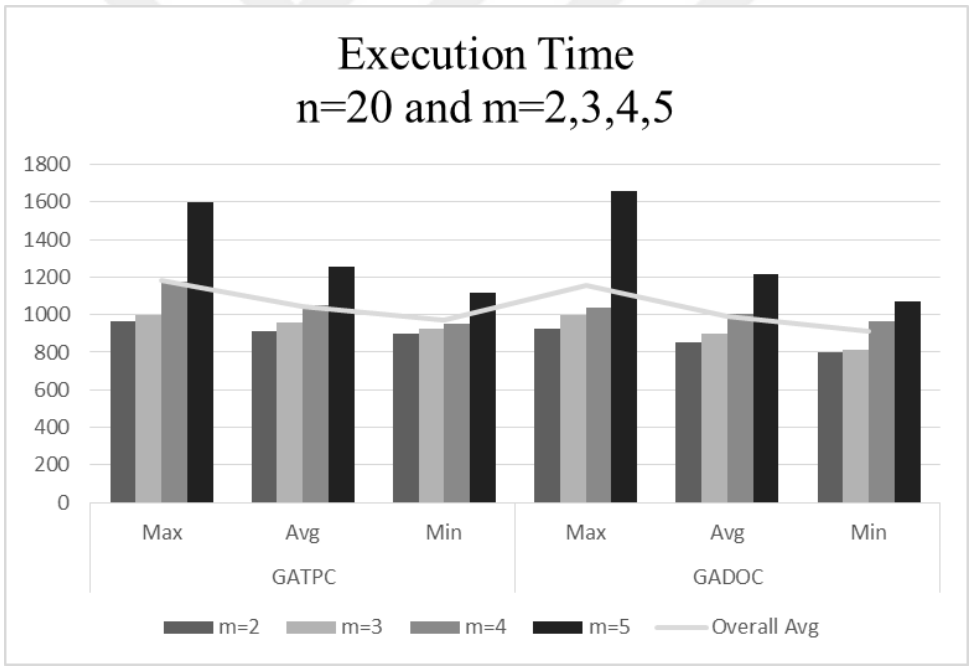
Graph 4.9. Upper Bound Deviation Performance of GATPC, and GADOC Algorithm for 20 orders on 2, 3, 4, and 5 Identical Parallel Machine

Table 4.10. Execution Time Performance of GATPC, and GADOC Algorithm for 20 orders on 2, 3, 4, and 5 Identical Parallel Machine

Execution Time (sec)		GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=20	m=2	964	914,8	897	927	854,4	801
	m=3	998	955,2	924	997	897	814
	m=4	1177	1053,4	951	1035	1007,2	968
	m=5	1596	1253,2	1119	1656	1216,4	1068
Overall Avg		1183,75	1044,15	972,75	1153,75	993,75	912,75

The GADOC has solved 20 order data sets regardless of the number of machines with an average of 993 seconds and for the GATPC is 1044 second. For GADOC and GATCP Algorithm, initial population 10000 and 100 iteration number was determined and these algorithms were run only 1 time each. The total number of solutions scanned for GADOC and GATPC is 1000000. It is explained how the

number of iterations and the starting population number are determined under the title of execution performance of GADOC. The execution time of the GADOC algorithm has been resolved in an average of 854 seconds for 20 orders on 2 machines, 897 seconds on 3 machines, 1007 seconds on 4 machines, and 1216 seconds on 5 machines. The execution time of the GATPC algorithm has been resolved in an average of 914 seconds for 20 orders on 2 machines, 955 seconds on 3 machines, 1053 seconds on 4 machines, and 1253 seconds on 5 machines. The GATPC and GADOC algorithms were run only for one recurrence but the probability of finding the correct result is increased with as high as possible initial population and iterations.

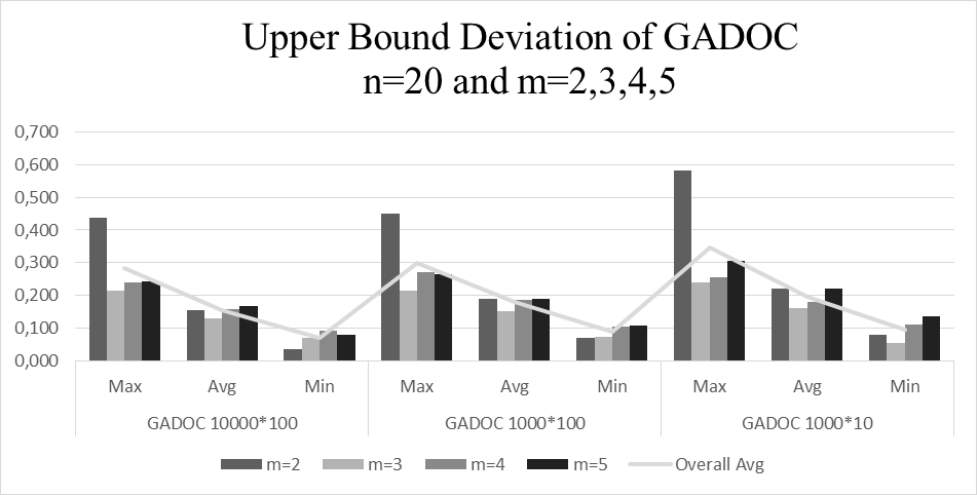


Graph 4.10. Execution Time Performance of GATPC, and GADOC Algorithm for 20 orders on 2, 3, 4, and 5 Identical Parallel Machine

Table 4.11. Upper Bound Deviation Performance of the GADOC Algorithm for 20 orders on 2, 3, 4, and 5 Identical Parallel Machine with Different Initial Population and Iteration Number

UBD		GADOC 10000*100			GADOC 1000*100			GADOC 1000*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=20	m=2	0,437	0,156	0,035	0,450	0,190	0,070	0,191	0,220	0,081
	m=3	0,216	0,132	0,070	0,215	0,151	0,074	0,239	0,161	0,056
	m=4	0,239	0,160	0,094	0,271	0,187	0,104	0,257	0,182	0,112
	m=5	0,243	0,169	0,081	0,266	0,189	0,107	0,306	0,221	0,136
Overall Avg		0,284	0,154	0,070	0,301	0,179	0,089	0,248	0,196	0,096

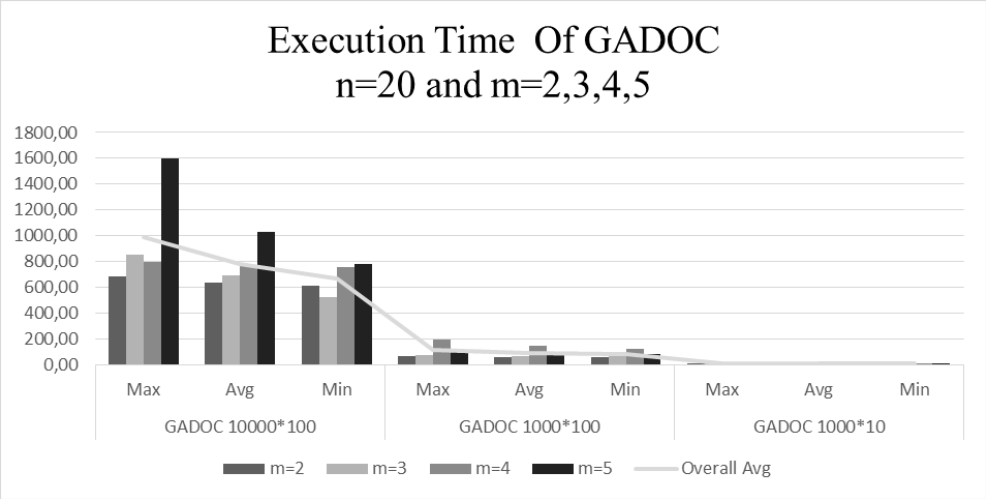
The GADOC showed the best performance among two genetic algorithms. Therefore, when the starting population and the number of iterations were changed, the performance of the GADOC algorithm was compared according to the Upper Bound Deviation (UBD) and the solution or execution time value. When the initial population is decreased from 10000 to 1000 and the number of iterations is kept constant at 100, the average of UBD for all number of machine groups is increased from 0,154 to 0,179. When the initial population stays constant with the value 1000 and the number of iterations is decreased from 100 to 10, the average of UBD for all number of machine groups is increased from 0,179 to 0,196. In general, when the table is examined, it has been observed that the minimum and the maximum upper bound deviation values in all numbers of machines gradually increase like the average values with the decrease of initial and iteration numbers. It is seen also that the maximum and minimum upper bound deviation values increase since the number of searches in the solution space of the problem decrease.



Graph 4.11. Upper Bound Deviation Performance of The GADOC Algorithm for 20 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

Table 4.12. Execution Time Performance of the GADOC Algorithm for 20 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

Execution Time (sec)		GADOC 10000*100			GADOC 1000*100			GADOC 1000*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=20	m=2	927	854,4	801	208,00	120,80	193,50	11,00	11,00	11,00
	m=3	997	897	814	112,00	101,80	88,00	8,00	8,20	9,00
	m=4	1035	1007,2	968	104,00	97,40	91,00	9,00	9,40	11,00
	m=5	1656	1216,4	1068	86,00	82,60	78,00	8,00	8,00	8,00
Overall Avg		1153,75	993,75	912,75	127,50	100,65	112,63	9,00	9,15	9,75



Graph 4.12. Execution Time Performance of The GADOC Algorithm for 20 orders on 2, 3, 4, and 5 Identical Parallel Machine with Different Initial Population and Iteration Number

When the execution time performance of the GADOC Algorithm is examined for instances with 20 orders on 2, 3, 4, and 5 identical parallel machines with different initial populations and iteration numbers, the execution time performance increases. The GADOC algorithm finds the solution in short times. When the initial population is decreases from 10000 to 1000 and the iteration number stays constant at 100, the average execution time decreases from 993 to 100. And when the initial population stays constant at 100, and the iteration number decreased from 100 to 10, the execution time goes down from 100 to 9. When the execution time is examined, it is observed that in no case the maximum, minimum, and average values are not the same and there is a clear decrease in execution time.

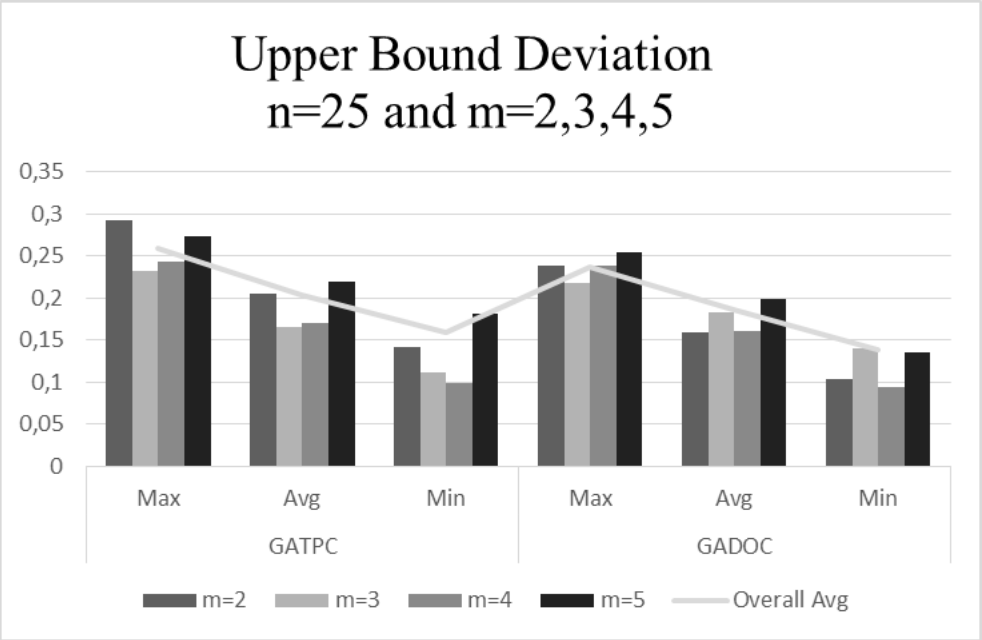
4.1.2.2. Results for 25 Order Sized Instances on 2, 3, 4, and 5 Identical Parallel Machines

Table 4.13. Upper Bound Deviation Performance of GATPC, and GADOC Algorithm for 25 orders on 2, 3, 4, and 5 Identical Parallel Machine

UBD		GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
	m=2	0,293	0,205	0,141	0,238	0,159	0,104
	m=3	0,232	0,166	0,112	0,218	0,183	0,14
n=25	m=4	0,24	0,222	0,2	0,234	0,205	0,175
	m=5	0,274	0,22	0,181	0,255	0,199	0,136
Overall Avg		0,25975	0,20325	0,1585	0,23625	0,1865	0,13875

In the table above, the solution of the GADOC algorithm for 25 order dataset on 2 identical parallel machines, the average upper bound deviation of the 5 data sets is 0.159, however, 0.205 for the GATPC algorithm. For the same system, the maximum upper bound deviation of the GADOC algorithm is 0,238 and the GATPC is 0.293. Also, the minimum deviation values are 0,104 for the GADOC and 0.141 for GATPC. Thus, it is seen that the GADOC algorithm has performed better than the GATPC algorithm on 2 identical parallel machines. On 3 identical parallel machines, the solving performance of the GADOC is an average of 0,183 upper bound deviations and for the GATPC is 0,166. On 4 identical parallel machines, the GADOC has the solving performance with 0,205 average upper bound deviations but GATPC has 0,222. On 5 identical parallel machines, GADOC has the better solving performance with an average of 0.199 upper bound deviations. The GATPC performance is 0, 22 average upper bound deviation. When the average upper bound deviation of the algorithm compares for all machine groups, the GADOC has the best and the lowest upper bound deviation. Also for the maximum and the minimum upper bound deviation for all machine number groups, The GADOC performed better by far than the GATPC with average upper bound deviation for all machine

groups respectively 0,187 and 0,203, and also described with maximum and minimum upper bound deviation and also described in the graph below.



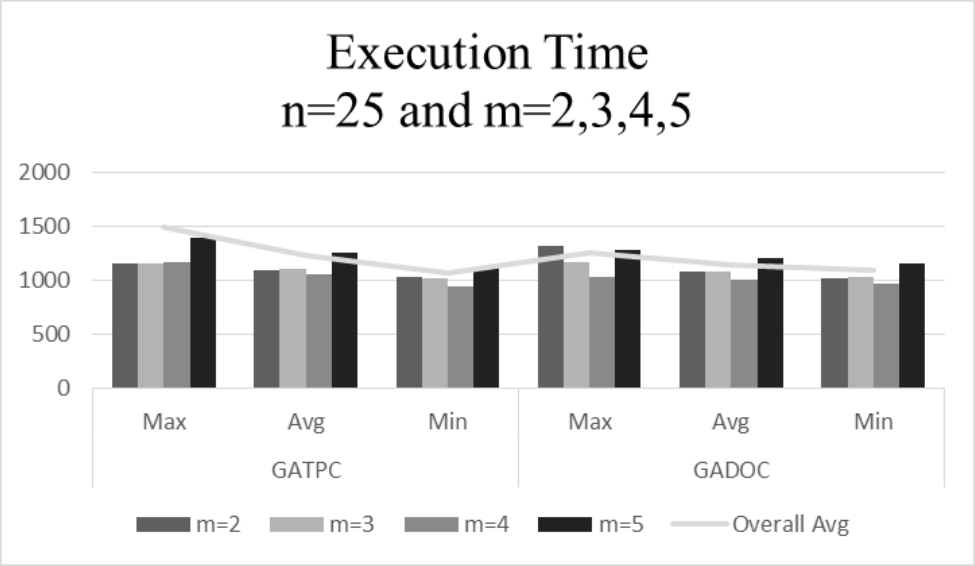
Graph 4.13. Upper Bound Deviation Performance of GATPC, and GADOC Algorithm for 25 orders on 2, 3, 4, and 5 identical Parallel Machine

Table 4.14. Execution Time Performance of GATPC, and GADOC Algorithm for 25 orders on 2, 3, 4, and 5 identical Parallel Machine

Execution Time (sec)		GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
	m=2	1153	1101,2	1033	1321	1083	1016
	m=3	1165	1111,6	1026	1177	1082,2	1031
n=25	m=4	2286	1476,2	1093	1234	1194,2	1161
	m=5	1398	1261	1121	1285	1205,4	1161
Overall Avg		1500,5	1237,5	1068,25	1254,25	1141,2	1092,25

The GADOC has solved the instances with 25 order data sets regardless of the number of machines with an average of 1141 seconds and for the GATPC is 1237

seconds. For GADOC and GATCP Algorithm, initial population 10000 and 100 iteration number was determined and these algorithms were run only 1 time each. The total number of solutions scanned for GADOC and GATPC is 1000000. It is explained how the number of iterations and the starting population number are determined under the title of execution performance of GADOC. The execution time of the GADOC algorithm has been resolved in an average of 1083 seconds for 25 orders on 2 machines, 1082 seconds on 3 machines, 1194 seconds on 4 machines, and 1205 seconds on 5 machines. The execution time of the GATPC algorithm has been resolved in an average of 1101 seconds for 25 orders on 2 machines, 1111 seconds on 3 machines, 1476 seconds on 4 machines, and 1261 seconds on 5 machines. The GATPC and GADOC algorithms were run only for one recurrence but the probability of finding the correct result is increased with as high as possible initial population and iterations.

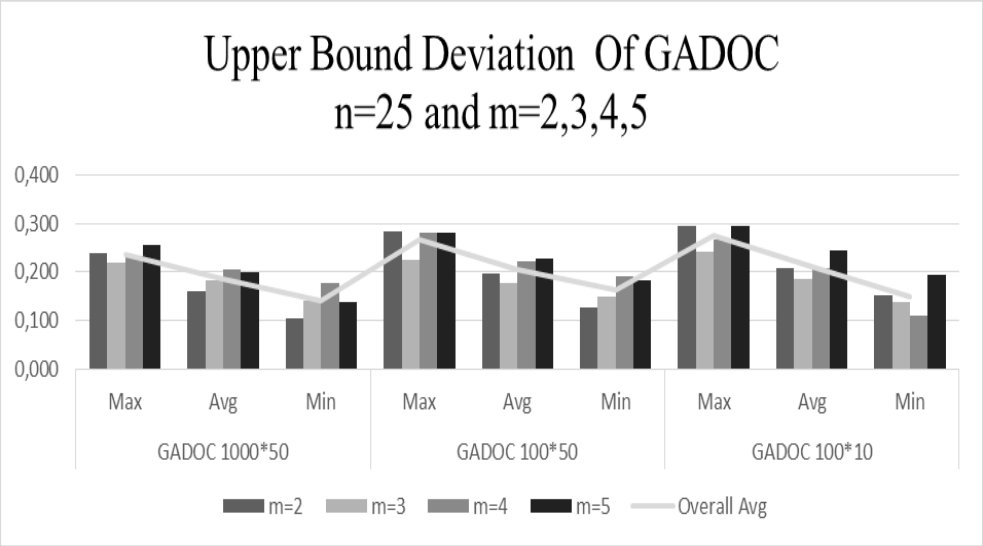


Graph 4.14. Execution Time Performance of GATPC, and GADOC Algorithm for 25 orders on 2, 3, 4, and 5 identical Parallel Machine

Table 4.15. Upper Bound Deviation Performance of the GADOC Algorithm for 25 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

UBD		GADOC 10000*100			GADOC 1000*100			GADOC 1000*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=25	m=2	0,238	0,159	0,104	0,283	0,196	0,126	0,295	0,207	0,152
	m=3	0,218	0,183	0,141	0,226	0,177	0,148	0,241	0,186	0,137
	m=4	0,234	0,205	0,176	0,281	0,221	0,192	0,268	0,207	0,111
	m=5	0,255	0,200	0,138	0,282	0,227	0,184	0,295	0,244	0,194
Overall Avg		0,236	0,187	0,140	0,268	0,205	0,163	0,275	0,211	0,149

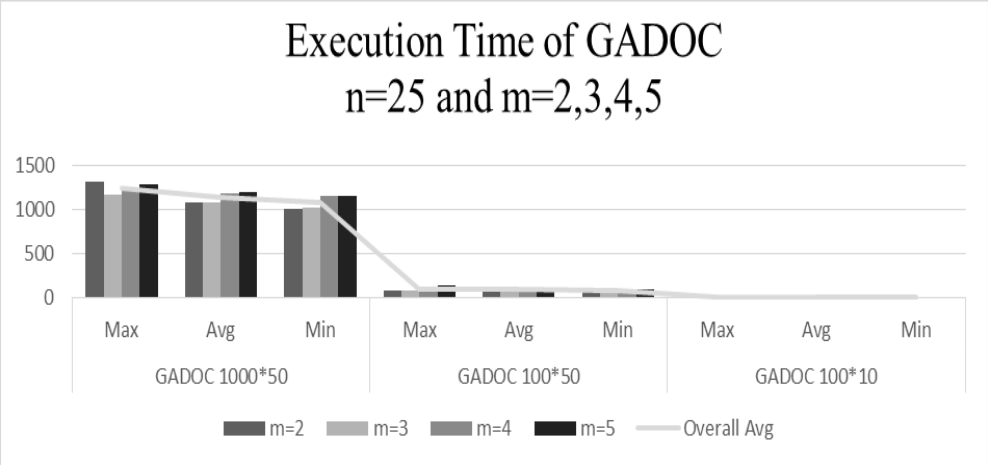
The GADOC showed the best performance among two genetic algorithms. Therefore, when the starting population and the number of iterations were changed, the performance of the GADOC algorithm was compared according to the Upper Bound Deviation (UBD) and the solution or execution time value. When the initial population is decreased from 10000 to 1000 and the number of iterations is kept constant at 100, the average of UBD for all number of machine groups is increased from 0,187 to 0,205. When the initial population stays constant with the value 1000 and the number of iterations is decreased from 100 to 10, the average of UBD for all number of machine groups is increased from 0,205 to 0,211. In general, when the table is examined, it has been observed that the minimum and the maximum upper bound deviation values in all numbers of machines gradually increase like the average values with the decrease of initial and iteration numbers. It is seen also that the maximum and minimum upper bound deviation values increase since the number of searches in the solution space of the problem decrease.



Graph 4.15. Upper Bound Deviation Performance of The GADOC Algorithm for 25 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

Table 4.16. Execution Time Performance of the GADOC Algorithm for 25 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

Execution Time (sec)		GADOC 10000*100			GADOC 1000*100			GADOC 1000*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=25	m=2	1321	1083	1016	81,00	79,60	79,00	8,00	8,00	8,00
	m=3	1177	1082	1031	91,00	88,40	86,00	9,00	9,00	9,00
	m=4	1234	1194	1161	108,00	100,00	94,00	9,00	9,00	9,00
	m=5	1285	1205	1161	139,00	118,00	104,00	10,00	10,00	10,00
Overall Avg		1254,25	1141	1092,25	104,75	96,50	90,75	9,00	9,00	9,00



Graph 4.16. Execution Time Performance of The GADOC Algorithm for 25 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

When the execution time performance of the GADOC Algorithm is examined for instances with 25 orders on 2, 3, 4, and 5 identical parallel machines with different initial populations and iteration numbers, the execution time performance increases. The GADOC algorithm finds the solution in short times. When the initial population is decreases from 10000 to 1000 and the iteration number stays constant at 100, the average execution time decreases from 1141 to 96. And when the initial population stays constant at 100, and the iteration number decreased from 100 to 10, the execution time goes down from 96 to 9. When the execution time is examined, it is observed that in no case the maximum, minimum, and average values are not the same and there is a clear decrease in execution time.

4.1.4. Results of Large-Sized Problems (n=50, n=100)

In this section, the upper bound deviation and execution time results of GADOC, and GATPC performance of large-size data sets with 50 and 100 orders are mentioned. Since 15 order and larger size data sets cannot be solved with MILP in the CPLEX algorithm, the upper bound deviation and execution time comparison

result of GADOC and GATPC algorithm solution performance of data sets with 50 and 100 orders are compared to analyze how far the solution methods deviated from the upper bound, which algorithm perform better and solution times were compared.

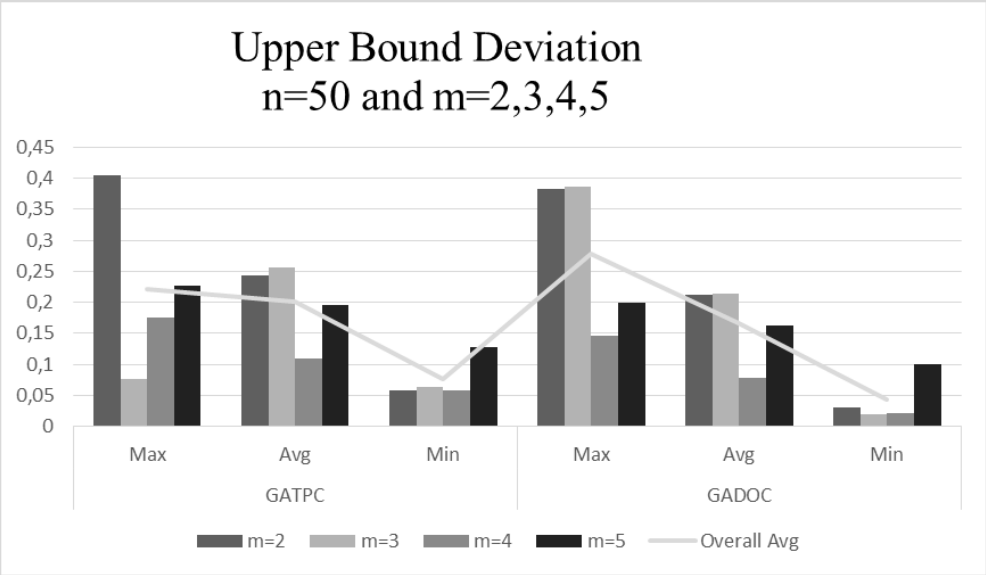
4.1.3.1 Results for 50 Order Sized Instances on 2, 3, 4, and 5 Identical Parallel Machines

Table 4.17. Upper Bound Deviation Performance of GATPC, and GADOC Algorithm for 50 orders on 2, 3, 4, and 5 identical Parallel Machine

UBD Deviation		GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
	m=2	0,404	0,243	0,058	0,382	0,213	0,03
	m=3	0,077	0,256	0,063	0,387	0,214	0,02
n=50	m=4	0,176	0,109	0,058	0,146	0,079	0,021
	m=5	0,226	0,196	0,128	0,2	0,163	0,1
Overall Avg		0,22075	0,201	0,07675	0,27875	0,16725	0,04275

In the table above, the solution of the GADOC algorithm for instances with 50 order on 2 identical parallel machines, the average upper bound deviation of the 5 data sets is 0.213, however, 0.243 for the GATPC algorithm. For the same system, the maximum upper bound deviation of the GADOC algorithm is 0,382 and the GATPC is 0.404. Also, the minimum deviation values are 0,003 for the GADOC and 0.058 for GATPC. Thus, it is seen that the GADOC algorithm has performed better than the GATPC algorithm on 2 identical parallel machines. On 3 identical parallel machines, the solving performance of the GADOC is an average of 0,214 upper bound deviations and for the GATPC is 0,256. On 4 identical parallel machines, the GADOC has the solving performance with 0,079 average upper bound deviations but GATPC has 0,109. On 5 identical parallel machines, GADOC has the better solving performance with an average of 0.163 upper bound deviations. The GATPC performance is 0,196 average upper bound deviation. When the average upper bound

deviation of the algorithm compares for all machine groups, the GADOC has the best and the lowest upper bound deviation. Also for the maximum and the minimum upper bound deviation for all machine number groups, The GADOC performed better by far than the GATPC with average upper bound deviation for all machine groups respectively 0,167 and 0,201, and also described with maximum and minimum upper bound deviation and also described in the graph below.

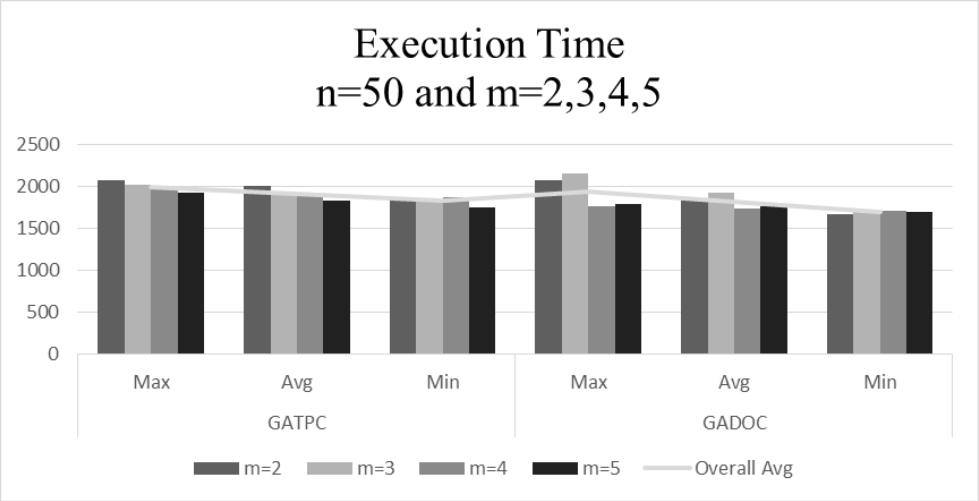


Graph 4.17. Upper Bound Deviation Performance of GATPC, and GADOC Algorithm for 50 orders on 2, 3, 4, and 5 identical Parallel Machine

Table 4.18. Execution Time Performance of GATPC, and GADOC Algorithm for 50 orders on 2, 3, 4, and 5 identical Parallel Machine

Execution Time (sec)		GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
	m=2	2079	2011,8	1875	2072	1859	1675
	m=3	2024	1924,6	1821	2158	1923,8	1702
n=50	m=4	1964	1885,6	1871	1765	1738,4	1709
	m=5	1924	1833,8	1754	1796	1761	1702
Overall Avg		1997,75	1913,95	1830,25	1947,75	1820,55	1697

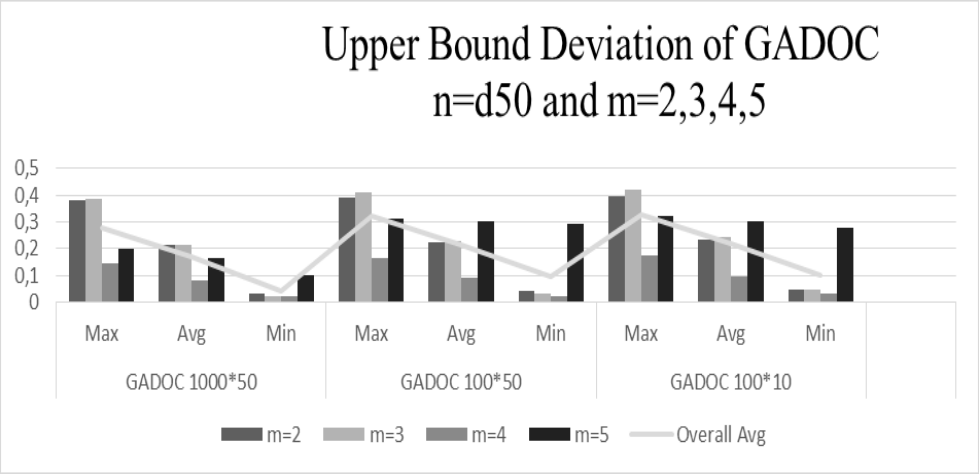
The GADOC has solved the instances with 50 order data sets regardless of the number of machines with an average of 1820 seconds and for the GATPC is 1913 seconds. For GADOC and GATCP Algorithm, initial population 10000 and 100 iteration number was determined and these algorithms were run only 1 time each. The total number of solutions scanned for GADOC and GATPC is 1000000. It is explained how the number of iterations and the starting population number are determined under the title of execution performance of GADOC. The execution time of the GADOC algorithm has been resolved in an average of 1859 seconds for 50 orders on 2 machines, 1923 seconds on 3 machines, 1738 seconds on 4 machines, and 1761 seconds on 5 machines. The execution time of the GATPC algorithm has been resolved in an average of 2011 seconds for 50 orders on 2 machines, 1924 seconds on 3 machines, 1885 seconds on 4 machines, and 1833 seconds on 5 machines. The GATPC and GADOC algorithms were run only for one recurrence but the probability of finding the correct result is increased with as high as possible initial population and iterations.



Graph 4.18. Execution Time Performance of GATPC, and GADOC Algorithm for 50 orders on 2, 3, 4, and 5 identical Parallel Machine

Table 4.19.. Upper Bound Deviation Performance of the GADOC Algorithm for 50 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

UBD		GADOC 10000*100			GADOC 1000*100			GADOC 1000*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=50	m=2	0,382	0,213	0,03	0,390	0,224	0,041	0,396	0,231	0,047
	m=3	0,387	0,214	0,02	0,408	0,229	0,034	0,421	0,242	0,049
	m=4	0,146	0,079	0,021	0,167	0,092	0,020	0,176	0,098	0,033
	m=5	0,2	0,163	0,1	0,314	0,300	0,291	0,322	0,301	0,277
Overall Avg		0,27875	0,16725	0,04275	0,320	0,211	0,097	0,329	0,218	0,102

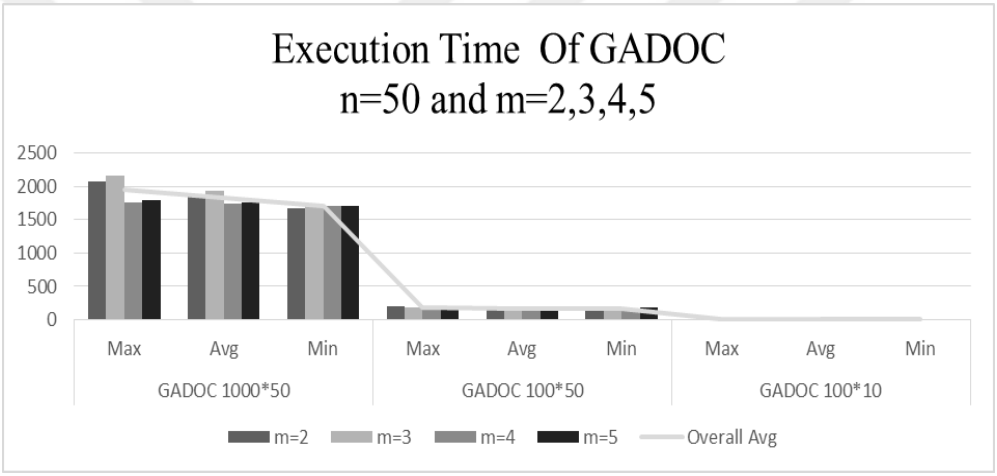


Graph 4.19. Upper Bound Deviation Performance of The GADOC Algorithm for 50 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

The GADOC showed the best performance among two genetic algorithms for instances with 50 orders. Therefore, when the starting population and the number of iterations were changed, the performance of the GADOC algorithm was compared according to the Upper Bound Deviation (UBD) and the solution or execution time value. When the initial population is decreased from 10000 to 1000 and the number of iterations is kept constant at 100, the average of UBD for all number of machine groups is increased from 0,167 to 0,211. When the initial population stays constant with the value 1000 and the number of iterations is decreased from 100 to 10, the average of UBD for all number of machine groups is increased from 0,211 to 0,218. In general, when the table is examined, it has been observed that the minimum and the maximum upper bound deviation values in all numbers of machines gradually increase like the average values with the decrease of initial and iteration numbers. It is seen also that the maximum and minimum upper bound deviation values increase since the number of searches in the solution space of the problem decrease for the instances with 50 orders.

Table 4.20. Execution Time Performance of the GADOC Algorithm for 50 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

Execution Time (sec)		GADOC 10000*100			GADOC 1000*100			GADOC 1000*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=50	m=2	2072	1859	1675	201,00	179,60	168,00	17,00	16,00	15,00
	m=3	2158	1923,8	1702	178,00	168,40	154,00	18,00	16,80	16,00
	m=4	1765	1738,4	1709	165,00	169,60	171,00	18,00	17,20	17,00
	m=5	1796	1761	1702	168,00	172,60	179,00	19,00	18,00	17,00
Overall Avg		1947,75	1820,55	1697	178,00	172,55	168,00	18,00	17,00	16,25



Graph 4.20. Execution Time Performance of The GADOC Algorithm for 50 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

When the execution time performance of the GADOC Algorithm is examined for instances with 50 orders on 2, 3, 4, and 5 identical parallel machines with different initial populations and iteration numbers, the execution time performance increases. The GADOC algorithm finds the solution in a short time. When the initial population is decreases from 10000 to 1000 and the iteration number stays constant at 100, the average execution time decreases from 1820 to 172. And when the initial population stays constant at 100, and the iteration number decreased

from 100 to 10, the execution time goes down from 172 to 17. When the execution time is examined, it is observed that in no case the maximum, minimum, and average values are not the same and there is a clear decrease in execution time.

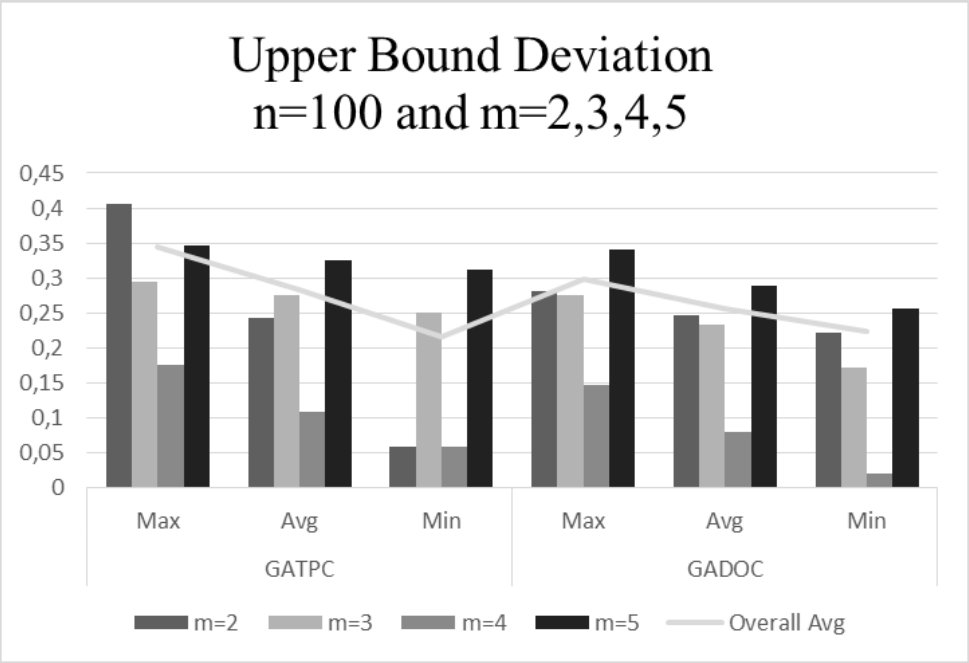
4.1.3.2 Results for 100 Order Sized Instances on 2, 3, 4, and 5 Identical Parallel Machines

Table 4.21. Upper Bound Deviation Performance of GATPC, and GADOC Algorithm for 100 orders on 2, 3, 4, and 5 identical Parallel Machine

UBD Deviation		GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=100	m=2	0,405	0,242	0,058	0,281	0,247	0,222
	m=3	0,295	0,276	0,251	0,276	0,233	0,171
	m=4	0,328	0,287	0,246	0,3	0,259	0,245
	m=5	0,347	0,325	0,311	0,34	0,288	0,256
Overall Avg		0,34375	0,283	0,2165	0,29925	0,257	0,2235

In the table above, the solution of the GADOC algorithm for instances with 100 order dataset on 2 identical parallel machines, the average upper bound deviation of the 5 data sets is 0247, however, 0.247 for the GATPC algorithm. For the same system, the maximum upper bound deviation of the GADOC algorithm is 0,281 and the GATPC is 0.405. Also, the minimum deviation values are 0,222 for the GADOC and 0.141 for GATPC. Thus, it is seen that the GADOC algorithm has performed better than the GATPC algorithm on 2 identical parallel machines. On 3 identical parallel machines, the solving performance of the GADOC is an average of 0,233 upper bound deviations and for the GATPC is 0,276. On 4 identical parallel machines, the GADOC has the solving performance with 0,259 average upper bound deviations but GATPC has 0,287. On 5 identical parallel machines, GADOC has the better solving performance with an average of 0.288 upper bound deviations, and the GATPC performance is 0, 325 average upper bound deviation. When the average upper bound deviation of the algorithm compares for all machine groups, the

GADOC has the best and the lowest upper bound deviation. Also for the maximum and the minimum upper bound deviation for all machine number groups, The GADOC performed better by far than the GATPC with average upper bound deviation for all machine groups respectively 0,257 and 0,283, and also described with maximum and minimum upper bound deviation and also described in the graph below.

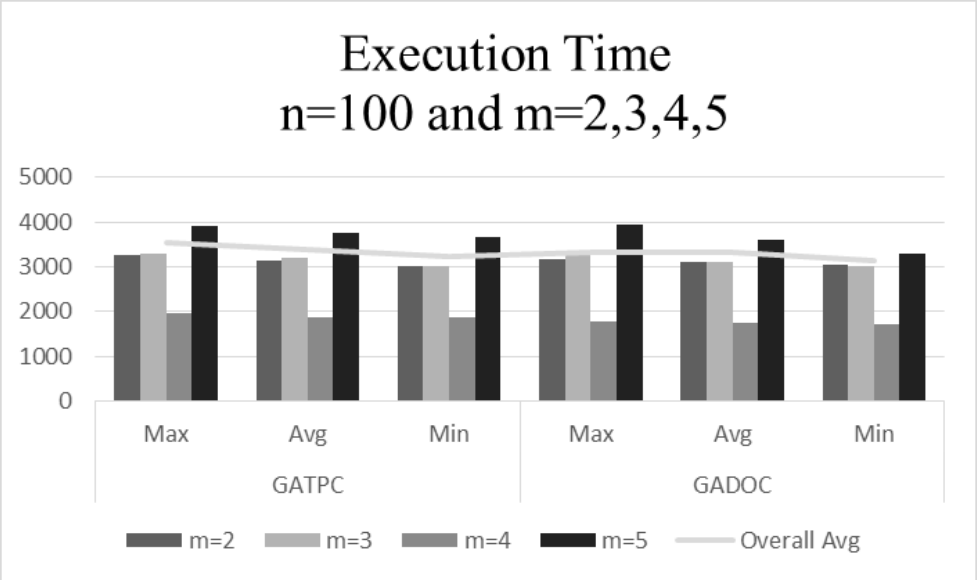


Graph 4.21. Upper Bound Deviation Performance of GATPC, and GADOC Algorithm for 100 orders on 2, 3, 4, and 5 identical Parallel Machine

Table 4.22. Execution Time Performance of GATPC, and GADOC Algorithm for 100 orders on 2, 3, 4, and 5 identical Parallel Machine

Execution Time (sec)		GATPC			GADOC		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=100	m=2	3256	3137,2	3012	3175	3101,6	3058
	m=3	3294	3198,8	3018	3362	3117,4	3020
	m=4	3680	3490,6	3270	3655	3471,4	3250
	m=5	3923	3749	3648	3944	3589,8	3281
Overall Avg		3538,25	3393,90	3237,00	3320,05	3320,05	3152,25

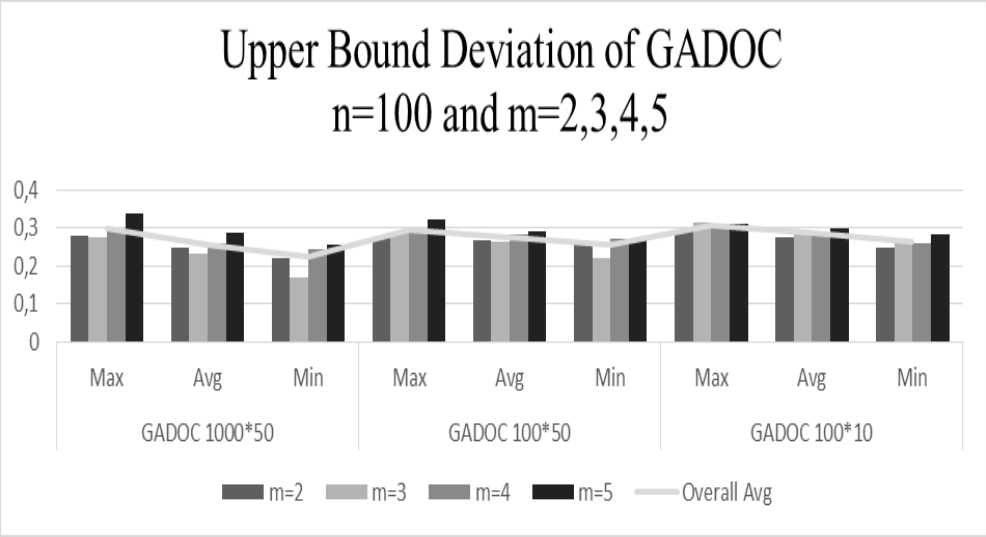
The GADOC has solved the instances with 100 order data sets for all of the number of machines with an average of 3320 seconds and the GATPC is 3393 seconds. For GADOC and GATCP Algorithm, initial population 10000 and 100 iteration number was determined and these algorithms were run only 1 time each. The total number of solutions scanned for GADOC and GATPC is 1000000. It is explained how the number of iterations and the starting population number are determined under the title of execution performance of GADOC. The execution time of the GADOC algorithm has been resolved in an average of 3101 seconds instances with 100 orders on 2 machines, 3117 seconds on 3 machines, 3471 seconds on 4 machines, and 3589 seconds on 5 machines. The execution time of the GATPC algorithm has been resolved in an average of 3137 seconds for 100 orders on 2 machines, 3198 seconds on 3 machines, 3490 seconds on 4 machines, and 3749 seconds on 5 machines. The GATPC and GADOC algorithms were run only for one recurrence but the probability of finding the correct result is increased with as high as possible initial population and iterations.



Graph 4.22. Execution Time Performance of GATPC, and GADOC Algorithm for 100 orders on 2, 3, 4, and 5 identical Parallel Machine

Table 4.23. Upper Bound Deviation Performance of the GADOC Algorithm for 100 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

UBD		GADOC 10000*100			GADOC 1000*100			GADOC 1000*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=100	m=2	0,281	0,247	0,222	0,277	0,267	0,256	0,297	0,277	0,250
	m=3	0,276	0,233	0,171	0,286	0,265	0,220	0,315	0,290	0,262
	m=4	0,3	0,259	0,245	0,297	0,284	0,273	0,309	0,285	0,261
	m=5	0,34	0,288	0,256	0,322	0,292	0,271	0,313	0,300	0,283
Overall Avg		0,29925	0,25675	0,2235	0,2955	0,277	0,255	0,3085	0,288	0,264

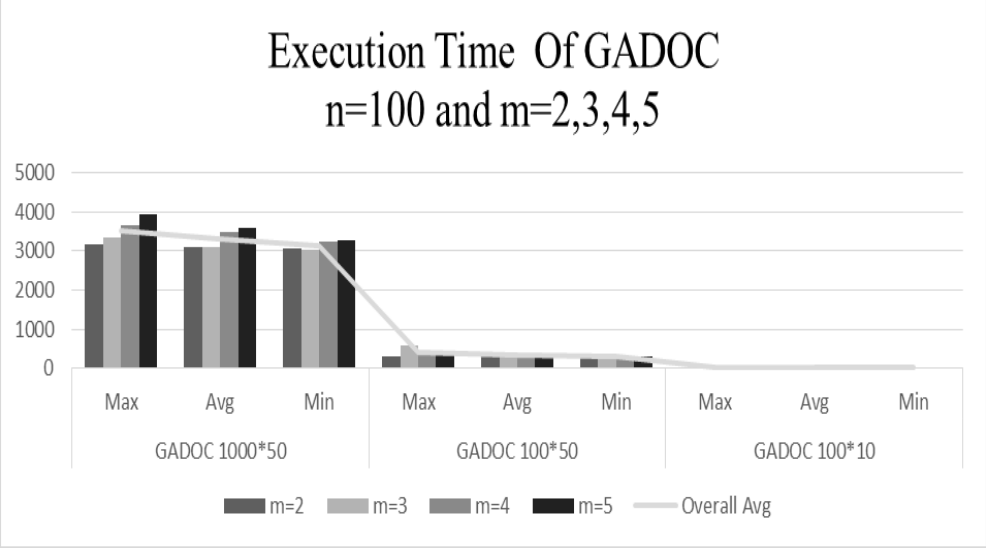


Graph 4.23. Upper Bound Deviation Performance of The GADOC Algorithm for 100 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

The GADOC showed the best performance among the two genetic algorithms. Therefore, when the starting population and the number of iterations were changed, the performance of the GADOC algorithm was compared according to the Upper Bound Deviation (UBD) and the solution or execution time value. When the initial population is decreased from 10000 to 1000 and the number of iterations is kept constant at 100, the average of UBD for all number of machine groups is increased from 0,247 to 0,277. When the initial population stays constant with the value 1000 and the number of iterations is decreased from 100 to 10, the average of UBD for all number of machine groups is increased from 0,277 to 0,288. In general, when the table is examined, it has been observed that the minimum and the maximum upper bound deviation values in all numbers of machines gradually increase like the average values with the decrease of initial and iteration numbers. It is seen also that the maximum and minimum upper bound deviation values increase since the number of searches in the solution space of the problem decrease.

Table 4.24. Execution Time Performance of the GADOC Algorithm for 100 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

Execution Time (sec)		GADOC 10000*100			GADOC 1000*100			GADOC 1000*10		
Order	Machine	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*
n=100	m=2	3175	3101,6	3058	322,00	297,00	286,00	38,00	34,80	33,00
	m=3	3362	3117,4	3020	569,00	385,60	307,00	45,00	40,50	35,00
	m=4	3655	3471,4	3250	352,00	329,20	310,00	38,34	38,21	38,06
	m=5	3944	3589,8	3281	342,00	329,00	319,00	38,50	38,43	38,38
Overall Avg		3534,00	3320,05	3152,25	396,25	335,20	305,50	39,96	37,99	36,11



Graph 4.24. Execution Time Performance of The GADOC Algorithm for 100 orders on 2, 3, 4, and 5 identical Parallel Machine with Different Initial Population and Iteration Number

When the execution time performance of the GADOC Algorithm is examined for instances with 100 orders on 2, 3, 4, and 5 identical parallel machines with different initial populations and iteration numbers, the execution time performance increases. The GADOC algorithm finds the solution in short times. When the initial population is decreases from 10000 to 1000 and the iteration number stays constant at 100, the average execution time decreases from 3320 to 335. And

when the initial population stays constant at 100, and the iteration number decreased from 100 to 10, the execution time goes down from 335 to 38. When the execution time is examined, it is observed that in no case the maximum, minimum, and average values are not the same and there is a clear decrease in execution time.

4.1.4. The Execution Performance of GADOC Algorithm

In this section, information about the solution performance of the developed GADOC algorithm will be given. Since the GADOC algorithm is the best performing algorithm according to the GATPC algorithm, the effect of solution time, initial population size, and the number of iterations on the solution performance of the algorithm is also examined in this study. The first performance examination is made on the first data set of 10 order set on 2 identical parallel machines.

Table 4.25. The GADOC performance with 100000 Total Solution in n=10 and m=2

Initial Population	Iteration Number	Obj	Total Solution	Execution Time(sec)	UBD
100	1000	111	100.000	119,4	0,1048
100	1000	106	100.000	157,06	0,1452
100	1000	108	100.000	172,86	0,129
100	1000	113	100.000	156,51	0,0887
1000	100	113	100.000	102,33	0,0887
2000	50	113	100.000	107,46	0,0887
5000	100	113	100.000	706,83	0,0887

When the initial population of the algorithm is 100 with 1000 iteration, the solution finding performance of the algorithm is not as quite as high according to 1000 initial population and 10 iterations. When the initial population is 1000 or more in 10 iterations, it can find the optimum solution which is also found by the CPLEX algorithm, very quickly for 10 order 2-machine system data set. When the initial population number is small, it is not enough to increase the number of iterations for correct results. The number of recurrences should also be increased for correct

results. But keeping the number of iterations higher than 1000 for 10 order data sets is just an unnecessary waste of time as seen in table 4.1.5.1.

As seen in table 4.1.5.2, choosing the initial population of 200 and above is the best choice for 10 order data sets. In the starting population of 200 and above, when the number of iterations is 100 and above, it reaches the optimum result only in 1 recurrence. When the initial population is 500 and above, the number of iterations of 100 and below is sufficient to find the best result in one recurrence in a 10 order dataset. When the initial population number is 1000 and above, even 10 iterations are sufficient to reach the optimal result. The problem-solving time of the program increases in direct proportion to the initial population and the number of iterations as shown in graph 4.1.5.1. For example, while the initial population number is 1000 and 2000, the time to make 1000 iterations is 11 minutes and 25 minutes, respectively. Likewise, while the initial population is 1000, the solution time for 10, 50, and 100 iterations are 13 seconds, 52 seconds, and 102 seconds respectively.

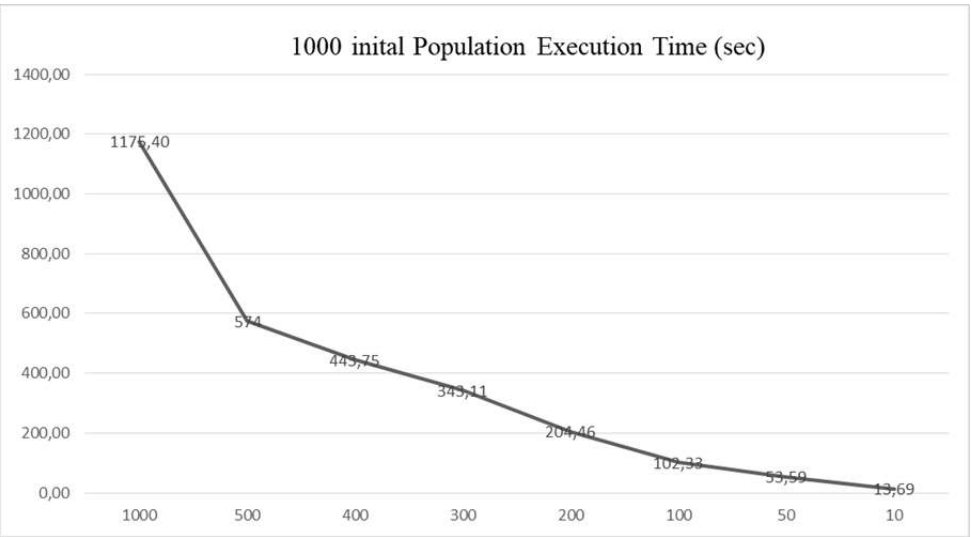


Figure 4.1. The GADOC performance on n=10 and m=2

If the initial population is chosen small as 100, the number of recurrence increases. For example, if 100 is selected for the initial population and the number of iterations is 1000, GADOC finds the optimum value in 4 recurrences. When the initial population is selected 1000, the number of iterations is selected 100 and a total of 100000 solutions are scanned with the same example, but the optimum solution is reached only in the first recurrence. On the other hand, when the initial population is chosen high and low iteration is selected, the program finds a faster solution even though it scans the same amount of solution. For this reason, for 10 data sets, taking the initial population as high as 1000 and iterating as low as 10 maybe a bit more is the best option to reach the best solution quality and time in one recurrence.

Table 4. 26. The GADOC performance with n=10 and m=2

10 Order on 2 Unrelated Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	102	1000	1,32	0,1774
100	20	105	2000	3,54	0,1532
100	50	113	5000	7,56	0,0887
100	100	113	10.000	12,74	0,0887
100	500	113	50.000	72,09	0,0887
100	1000	111	100.000	119,4	0,1048
100	2000	113	200.000	331,29	0,1452
200	100	113	20.000	21,6	0,129
200	1000	113	200.000	306,75	0,0887
500	100	113	50.000	54,45	0,0887
500	1000	113	500.000	766,91	0,1048
1000	1000	113	1.000.000	1175,40	0,1452
1000	500	113	500.000	574	0,129
1000	400	113	400.000	443,75	0,0887
1000	300	113	300.000	343,11	0,0887
1000	200	113	200.000	204,46	0,0887
1000	100	113	100.000	102,33	0,0887
1000	50	113	50.000	53,59	0,0887
1000	10	113	10.000	13,69	0,0887
2000	10	113	20.000	24,78	0,0887

On the other hand, another important point is that the situation examining is when the order number is small, about 10. However, when the number of orders increases and the number of machines changes, it is necessary to examine how the solution performance of the algorithm changes. The performance of the GADOC algorithm was examined when the number of machines remained the same as the previous example and was 2, but the order number increased to 15. It is expected that the performance of the program will decrease a little more due to the expansion of the space sought for a solution.

Table 4.27. The GADOC performance with n=15 and m=2

15 Order on 2 Unrelated Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	1000	156,5	100.000	187,54	0.1006
1000	100	155,75	100.000	169,47	0.1049
1000	100	158	100.000	225,74	0.0920
2000	50	156,5	100.000	163,43	0.1006
2000	100	158	200.000	305,84	0.0920
2000	100	158	200.000	441,84	0.0920
2000	500	158	1.000.000	1594,50	0.0920
2000	1000	158	2.000.000	3077,70	0.0920
3000	100	158	300.000	536,46	0.0920
5000	100	158	500.000	936,45	0.0920

As seen in the Table above the smallest UBD value is obtained when the initial population number is too large as 1000 or bigger. The UBD value increases when the initial population number decreases to 100. By looking at these values, keeping the initial population number high is more important than keeping the number of iterations is to obtain the correct result.

Table 4.28. The GADOC performance with n=50 and m=2

50 Orders on 2 Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	1000	398	100.000	463,64	0.2737
1000	100	407	100.000	491,51	0.2573
1000	1000	415	1.000.000	4863,67	0.2427
2000	50	402	100.000	430,10	0.2664
2000	100	407	200.000	860,80	0.2573
2000	500	417	1.000.000	4304,36	0.2391
2000	1000	417	2.000.000	9208,50	0.2391
20.000	100	417	2.000.000	8383,10	0.2391

As seen in the Table above the smallest UBD value is obtained when the initial population number is too large as 2000 or bigger. The UBD value and the recurrence increase when the initial population number decreases to 100. By looking

at these values, keeping the initial population number high is more important than keeping the number of iterations is to obtain the correct result.

The values of UBD values and solution times for the solution of 10, 15, and 50 order data sets of the GADOC algorithm in 2, 3, 4, and 5 parallel machine systems, in which initial population and in which the number of iterations is given in more detail in the Appendix-A section.

In conclusion, in this study, the initial population number is taken to be quite high for the GADOC algorithm to find the optimum value for all data sets. In the 10-order data sets, the initial population number was taken as 10000 and the number of iterations was taken as 50. However, to ensure the optimum result in medium and large data sets, the initial population number is 10000 and the number of iterations is 100. Thus, it is easier and more guaranteed to reach the optimum solution of the algorithm in the solution of 15,20,25,50, and 100 order data sets.

4.1.5. Sensitivity Analyses

Scheduling problems are crucial for firms to handle their production tasks to deliver on time to the customers. Firms' main aim is the maximization of profit. Maximization of profit is related to any field in the company like marketing, manufacturing, financial, and all the other department of the company. For this reason, a company should decide by gathering the targets of all departments while making a decision. For the production department, the purpose in the production planning may be the minimization of production time, intermediate stocks, or other losses, while the main purpose in the marketing department can be customer satisfaction maximization. However, the most fundamental main purpose for a firm as a whole is to make a profit. For the company to make a profit, it is very important to have common planning in which the purpose of each department is taken into account. For this reason, order acceptance and scheduling is a study that meets both the objective of the marketing department and the objective of production scheduling in production planning.

The optimum result found may sometimes not be sufficient for decision-making managers. Scheduling problems are different from other optimization problems due to their nature. The optimum value of the problem will likely change as a result of the changes in existing parameters in scheduling problems. For example, the sequence of orders in the scheduling will affect many results, such as the completion time of other orders, the makespan, and net profit. Therefore, the parameter change that keeps the optimum value constant is quite difficult in scheduling problems. Maybe other solutions can be found close to the optimum solution or have the same result as the optimum solution with a different order sequence. The existence of other solutions makes it easier for the decision-maker. Or, prices and other parameters are changing dynamically in today's world. (Nicholas Hall, 2004)A change that does not exist during modeling may occur later. Parameters such as changes in profits of products, changes in delay penalties of products, or penalties for outsourcing may change. For this reason, sensitivity analysis has been made in this section and it is revealed what can happen in case of change and how the results can be affected.

Real-life data and system model parameters are highly variable. Problems generally seek answers according to one or several scenarios. After finding answers to the problems, results are sought by decision-making managers for different scenarios or small changes. Sensitivity analysis is also used in these cases. Sensitivity analysis is performed by decision-makers to help them make decisions in different scenario situations. Available optimum for sensitivity analysis with linear programming problems the optimum solutions must be found. (Yılmaz, 2010) With sensitivity analysis, the cases are studied when the optimum solution remains unchanged. With the possible changes in the coefficients of the model and if there has been a change, the new to determine how to affect an optimum solution.

In scheduling problems, it is very difficult for the objective function value to remain constant with parameters or sequence of order changing. For this reason, this study, it is tried to find the other order sequences that give the same best optimum

value. A genetic algorithm was used while finding sequences giving the same objective function value. A separate large-scale study is required to find the sequences that give the same objective function value. Since scheduling problems are the difficult problem type, it is preferred to work with the smallest data set. The smallest working group in this study is the data set with 10 orders and 2 machines. The objective function value of net profit found with the MILP is 113. The other order sequences with the net profit value of 113 were found with the Genetic Algorithm. (Table 4.30)

When making decisions, managers generally need different scheduling options. However, this is not possible with a single schedule. Having more than one schedule that gives the same best objective function value provides convenience and variety for production planning managers. The more alternatives offered to the managers while making decisions, the more effective decisions can be done in real life.

Generally, there are 2 main situations in a sensitivity analysis. The first is what will happen when the optimum objective function value is constant and other parameters change, the second is what the new objective function value will be as a result of the changes in the equations. In scheduling problems, when the sequence of orders changes, generally the objective function value also changes. For the first, the sequences of orders with the same best objective value are shown in this study with the same optimum solution. (Table 4.30) In the second part, it is shown how the value of the objective function is affected by the changes in the objective function coefficients.

Sensitivity Analyze on the Constant Optimum Objective Function Value

In scheduling problems, sensitivity analysis with constant optimum objective function value can be made with different sequences of orders if it is possible. Different order sequences with the same net profit will provide flexibility

in decision-making. In this study, different order sequences with the same optimum objective function value were obtained with the help of a Genetic Algorithm.

In this section, the optimum solution and other feasible solutions with a value of 113 obtained by solving the first data set with 10 orders on 2 identical parallel machines with the MILP will be shown.

The optimum solution value that is found by the MILP for the first data set with 10 orders on 2 identical parallel machines is 113 given in tables 4.30,4.31 and 4.32

Table 4.29. The best possible sequence of orders and optimum solution

Machine 1	**3,9,2**
Machine 2	**4,5,6,8,10**
Outsourced Orders	**1 and 7**
Orders with Delay Punishment	-
Objective Function Value	**113**

Table 4.30. The other possible sequence of orders with the same objective functions

Machine 1	*3,2,9,10*
Machine 2	*6,8,4,5*
Outsourced Orders	*7 and 1*
Orders with Delay Punishment	-
Objective Function Value	*113*
Machine 1	**3,4,10,5**
Machine 2	**9,2,8,6**
Outsourced Orders	**7 and 1**
Orders with Delay Punishment	-
Objective Function Value	**113**
Machine 1	*4,8,10,3,6*
Machine 2	*5,2,9*
Outsourced Orders	*1 and 7*
Orders with Delay Punishment	-
Objective Function Value	*113*
Machine 1	**9,2,3,10**
Machine 2	**5,6,4,8**
Outsourced Orders	**7 and 1**
Orders with Delay Punishment	-
Objective Function Value	**113**
Machine 1	*3,2,8,10*
Machine 2	*9,4,5,6*
Outsourced Orders	*1 and 7*
Orders with Delay Punishment	-
Objective Function Value	*113*

Table 4.31. The other possible sequence of orders with the same objective functions

Machine 1	**3,2,4,10**
Machine 2	**9,8,6,1,5**
Outsourced Orders	**7 and 1**
Orders with Delay Punishment	**-**
Objective Function Value	**113**
Machine 1	***8,4,5,6***
Machine 2	***3,2,9,10***
Outsourced Orders	***1 and 7***
Orders with Delay Punishment	**-**
Objective Function Value	***113***
Machine 1	**3,9,2,4**
Machine 2	**8,6,10,5**
Outsourced Orders	**7 and 1**
Orders with Delay Punishment	**-**
Objective Function Value	**113**
Machine 1	***4,6,5,8,10***
Machine 2	***9,3,2***
Outsourced Orders	***1 and 7***
Orders with Delay Punishment	**-**
Objective Function Value	***113***
Machine 1	**8,5,9,2**
Machine 2	**4,6,3,10**
Outsourced Orders	**7 and 1**
Orders with Delay Punishment	**-**
Objective Function Value	**113**

The optimum solution on the first data set of 10 orders with 2 identical parallel machines is 113. In all the possible feasible solutions in the scheduling, it is observed that the order numbered 1 and 7 were rejected. Data sets were checked whether this decision was correct or not. Among the orders in this dataset, the two orders with the lowest return are 1 and 7. Therefore, this problem model aiming the maximization of total net revenue, it is best to reject the orders that have the lowest profit. On the other hand, the outsourcing penalty for orders numbered 1, 7, 9, and 10 is also lower than the other orders in the data set. For this reason, it is seen that the results given by the program for this problem dataset are appropriate with the decisions to be made for the desired goal.

Sensitivity Analyze on Objective Function Coefficients

In this study, the objective function equation depends on three main parameters which are revenue of each order, delay punishment pay if it occurs, and outsourcing cost if the product doesn't include in the accepted order scheduling list. So in this section, there is an experiment on the coefficients of the objective function. This experiment is applied to understand what value the objective function takes when the equation parameters change. The instances in the previous section are chosen for an example data set. This instance is with 10 orders on 2 identical parallel machines. The coefficients of each objective function parameter multiply with the numbers from 1 to 40 to be able to analyze which values take the objective function.

Table 4.32. k-fold increase in outsourcing penalty (k*Co) and new objective function value (OBJ) for n=10 on 2 identical parallel machine

Co*k	1	1,5	2	2,5	3	3,5	4	4,5	5	10	20	30	40
OBJ	113	111,5	110	108,5	107	105,5	104	102,5	101	86	56	26	-4

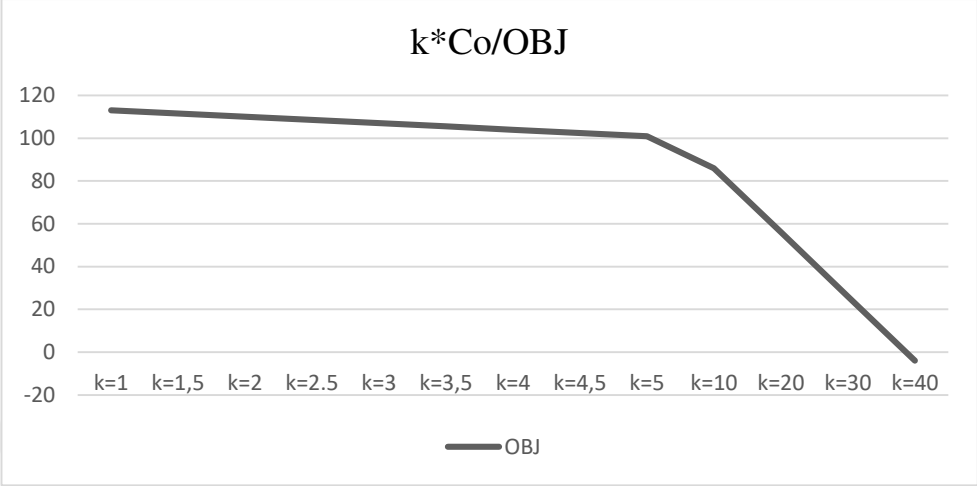


Figure 4.2. K-fold increase in outsourcing penalty (k*Co) and new objective function value (OBJ) for n=10 on 2 identical parallel machine

In Table 4.30, the relation between outsourcing parameter and objective function value is studied. When the outsourcing penalty increases, the objective function value decreases. It has been shown also in Figure 4.27.

Table 4.33. k-fold increase in Total Revenue (k*E) and new objective function value (OBJ) for n=10 on 2 identical parallel machine

E*k	1	1,5	2	2,5	3	3,5	4	4,5	5	10	20	30	40
OBJ	113	171	229	287	345	403	461	519	577	1161	2388	3628	4868

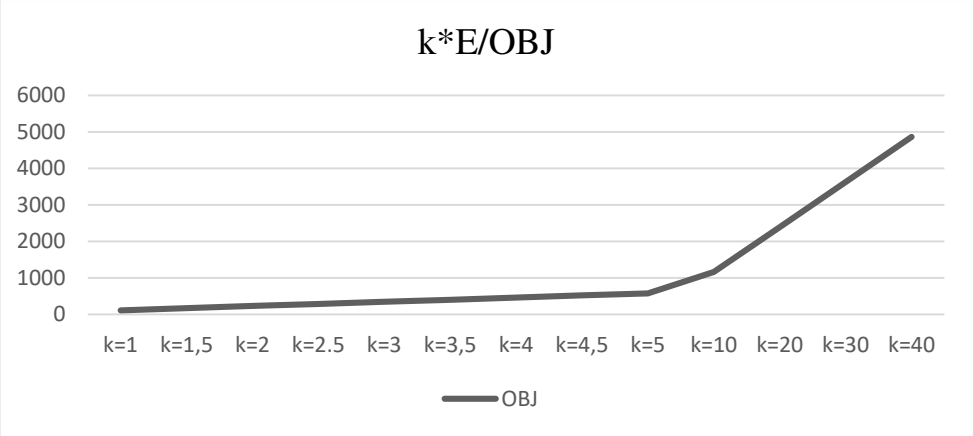


Figure 4.3. K-fold increase in Total Revenue (k*E) and new objective function value (OBJ) for n=10 on 2 identical parallel machine

In table 4.34, the relation between revenue parameter and objective function value is studied. When the revenue increases, the objective function value increases. These values have been shown in also Figure 4.3.

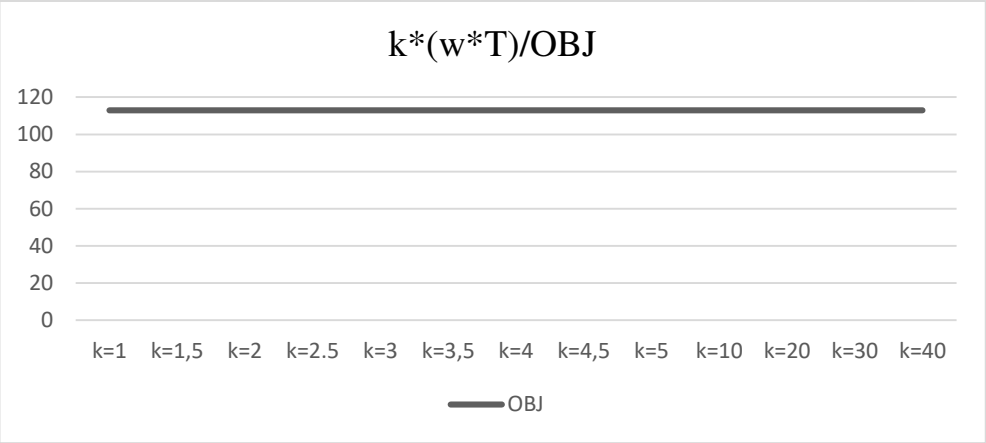


Figure 4.4. K-fold increase in Delay Penalty (k*w*T) and new objective function value (OBJ) for n=10 on 2 identical parallel machine

Table 4.34. k-fold increase in Delay Penalty (k*w*T) and new objective function value (OBJ) for n=10 on 2 identical parallel machine

W*T*k	1	1,5	2	2,5	3	3,5	4	4,5	5	10	20	30	40
OBJ	113	113	113	113	113	113	113	113	113	113	113	113	113

In figure 4.4 and Table 4.35, the relation between delay parameter and objective function value is studied. When the delay increases, the objective function value stays constant because in this instances example, in the optimum result scheduling, there is no order with delay. The reason the value for the objective function remains constant is that there is no delay in scheduling the orders in the data set. In other words, the products that are accepted into production scheduling have always made a full profit without delay cost and some orders are not included in the schedule due to outsourcing. Therefore, the change in the delay penalty coefficient did not affect the objective function. These values have been shown in also Figure 4.4.

4.1.6. General Result of FOAS-IPM and Discussion

Order acceptance and scheduling problem is an NP-Hard problem that is difficult to solve in multi-machine parallel systems as well as in single-machine systems. In this study, outsourcing, fuzzy production set-up, and delivery times were addressed to the problem of order acceptance and scheduling in multiple identical parallel machine systems. Firstly, Fuzzy Order Acceptance and Scheduling Problem on Identical Parallel Machine System (FOAS-IPM) problem is formulated as a mixed-integer linear mathematical model. 5 data sets were created for each order number and machine number. 2, 3, 4 orders were created for the small dataset 10 and 15, and 500 orders for the 5-machine. Likewise, 20, 25 medium-size data sets were created in the same way. 50 and 100 large datasets were created in the same way. A total of 4400 orders have been analyzed.

Small instances with 10 orders can be solved by the exact method in a reasonable time with the CPLEX algorithm in the GAMS program. CPLEX algorithm can solve the FOAS-IPM problem up to 14 orders regardless of machine number. For this reason, the problem has been tried to be solved by using the genetic algorithm, which is one of the metaheuristic methods, to solve the larger data sets belonging to the problem.

The GADOC and GATPC genetic algorithms with 2 different crossover methods have been developed to solve the problem. Both genetic algorithms offered solutions close to the solution of the CPLEX algorithm in small data sets of 10. Two genetic algorithms, in the solution of some small data sets, found an optimum result faster than the CPLEX algorithm. However, the CPLEX algorithm has reached all small 10 data sets short and optimum results. The GADOC algorithm has found better results than the GATPC algorithm in the solution of data sets with orders of 10 or more. With the increasing number of orders of large data sets, the performance of the GADOC algorithm is revealed more clearly. Also, the GADOC algorithm solves the problem faster than the GATPC algorithm.

The order acceptance and scheduling problem is a very important scheduling problem for companies that have capacity constraints. Customer satisfaction is very important in order-oriented production. On-time delivery is critical for customer satisfaction. With this scheduling, timely delivery of the order and maximization of profit, which is the main purpose for the company, are provided.

This study is an important study that can shed light on future studies. The uncertainties that exist in real life are also included in this problem as fuzzy time parameters. On the other hand, in cases where the order cannot be accepted, the order can be delivered to the customer on time by outsourcing. Unlike other production planning schedules, it is a very important scheduling study because it deals with both time and profit.

It has provided very successful results to the problem in the developed algorithms. How the initial population number, randomness, and number of

iterations affect the solution in the developed algorithms has also been demonstrated effectively.

5. CONCLUSION AND FUTURE STUDIES

The thesis provides optimization and scheduling on the fuzzy order acceptance problem on identical parallel machines. Firstly, an order acceptance and scheduling problem is formulized as mixed-integer linear programming to have the maximum net profit with the scheduling. In this problem, a developed mathematical model enables optimization in terms of both profit and time in the production schedules developed for real-life production planning. On the other hand, it also includes outsourcing costs of orders that have to be rejected. Fuzzy times that exist in real-life systems are handled in this model. For this reason, modeling that is more up-to-date and close to real-life systems has been made than the order acceptance and scheduling studies examined in the literature to date. Besides, as in real life, having more than one machine in the system makes the model more realistic.

The mathematical model developed has found optimum scheduling successfully. The scheduling found gave the optimum result to achieve the highest possible profit. However, this has been possible for CPLEX in solving small data sets with 10 orders. To cope with this situation, the genetic algorithm, which is a heuristic algorithm, has been used. 2 different genetic algorithms using 2 different crossover methods have been developed. Three solution methods were tested with small data sets and they were compared according to their solving performance and execution time. The upper bound deviation is used to compare the solving performance. The upper bound deviation is the portion between the result and the total revenue of each order set. The upper bound deviation calculation is calculated by proportioning the sum of the returns of the orders in the data set and the difference in the results of the algorithm to the total revenue. As a result of this proportioning, the closest to the desired total revenue and the highest net profit as a result of this, the smallest upper bound deviation is desired. Orders of 10 have been resolved with CPLEX, GADOC, and GATCP algorithm, and 3 algorithm performances are compared. The CPLEX algorithm is the best solution for data sets with a slight

difference from the GADOC algorithm. GATPC is an algorithm that solves the instances the worst. In large data sets, GADOC and GATPC were compared and the GADOC algorithm performs better. Thus GADOC algorithm with Davis Order Crossover is more successful than the GATPC algorithm with a two-point crossover in finding correct results in a multi-machine system. Although the number of orders and number of machines in the data set varies, GADOC works better and faster than the GATPC algorithm in terms of execution speed and upper bound deviation.

Since the GADOC algorithm offers a better solution than the GATPC algorithm, a more detailed performance study has been carried out for the GADOC algorithm. The GADOC performance in the number of different iterations and the starting population has been studied in more detail. The optimum result can be achieved with the higher starting population and low iteration number and it performs better than the high number of iterations and low initial population number. Therefore, to increase the probability of GADOC and GATPC algorithms to reach optimum results, a high initial population number and appropriate iteration number were determined.

Sensitivity analysis, which offers alternatives for decision-makers, was also carried out in this study. Sensitivity analysis has been performed to examine optimum results in different scenarios. In sensitivity analysis, the effect of changing the objective function coefficients on the objective function. Also after the optimum result is found, different sequencing alternatives are presented by keeping the value of the objective function constant.

This paper also presents both theoretical and practical implications of production scheduling. This study deals with scheduling, which is the most important point for production planning, in terms of net profit maximization, which is the main objective of the firm. Uncertainties in real life have been included in this study and the durations have been studied as indefinite. On the other hand, unlike other orders and acceptance and scheduling, multiple machine systems have been handled. Also, an outsourcing policy has been considered ed for rejected orders.

5.1. Future Research Directions

In this study, the problem of order acceptance and scheduling has been addressed by dealing with fuzzy logic and outsourcing. While establishing the model in this study, some assumptions and problems in some systems could not be included. The system is modeled as close to reality. However, of course, more realistic system parameters can be added to this problem in future studies.

In this study, the factors that extend the production time in real-life systems are not discussed. In future studies, machine breakdown times, idle times, and other periods in the production process can be taken into account to make more realistic studies. On the other hand, the revenue of the product has been taken constantly in a period. However, even at the end of the manufacturing period of the product in real life, the return is not the same due to reasons such as fluctuations in the markets, exchange rate changes, or inflation. More realistic model studies can be handled in future studies. It can also be added to the objective function as other factors that reduce profit in costs such as inventory or holding cost.

The mathematical model can be set up differently for the solution of the problem outside of its content. With a different mathematical model, a higher number of orders can be analyzed. Or it can be resolved better with a different metaheuristic. There are many different kinds of heuristic algorithms in the literature. This can be solved by heuristics or a combination of heuristics. These may produce better results for the solution.

REFERENCES

Akyurt, D. Ö. Ü. İ. Z. 2012. Üretİm Sİstemlerİnİn Planlanmasi. İstanbul Üniversitesi: İstanbul Üniversitesi Açik Ve Uzaktan Eğitim Fakültesi.

Barak, M. Z. E. 2015. Sipariş Kabul Ve Çizelgeleme Probleminin Parçacik Sürü Optimizasyonu İle Çözülmesi. Çukurova Üniveristesi, Adana.

Bilginturk, Z. 2007. Order Acceptance And Scheduling Decisions In Make-To-Order Systems. Unpublished Master of Science, Koc University, İstanbul.

Buckley, J. E., Esfandiar. 2002. An Introduction to Fuzzy Logic and Fuzzy Sets.

Cesaret, B. O., Ceyda;Sibel Salman, F. 2012. A tabu search algorithm for order acceptance and scheduling. Computers & Operations Research, 39(6): 1197-1205.

Chakraborty, S. P. M., G;Sarkar B. 2006. An Approach To Manage Constraint Resource And Outsourcing Decision. Journal of Scientific and Industrial Research 65: 625-631.

Chaurasia, S. N., & Kim, J. H. 2019. An Artificial Bee Colony Based Hyper-heuristic for the Single Machine Order Acceptance and Scheduling Problem, Decision Science in Action: 51-63.

Chen, S.-H., Liou, Y.-C., Chen, Y.-H., & Wang, K.-C. 2019. Order Acceptance and Scheduling Problem with Carbon Emission Reduction and Electricity Tariffs on a Single Machine. Sustainability, 11(19).

Cihan, C. K., Panyi; Hungaryakos, Varga. 2017. Try Not To Be Late! – The Importance Of Delivery Service In Online Shopping. Organizations And Markets in Emerging Economies,, 8.

Çelik , Y. Y., İlker Karadeniz, Alper Talha. 2019. Son Üç Yılda Geliştirilen Metasezgisel Algoritmalar Hakkında Kısa Bir İnceleme. European Journal of Science and Technology, Special Issue: 463-477.

Çiftçi, H. Ö. 2020. Sipariş Tipi Üretim Sistemlerinde Üretim Planlama Ve Bir İşletmede Örnek Uygulama. T.C. Maltepe Üniversitesi.

Diwakar Gupta, A. S. B. 2004. Make-to-order, make-to-stock, or delay product differentiation? A common framework for modeling and analysis. IIE Transactions: 529–546.

Dr. Öznur İşçi, P. D. S. K. 2003. Genetik Algoritma Yaklasim ve Yöneylem Arastirmasinda Bir Uygulama. Yönetim Ve Ekonomi, 10(2).

Emami, S., Sabbagh, M., & Moslehi, G. 2015. A Lagrangian relaxation algorithm for order acceptance and scheduling problem: a globalised robust optimisation approach. International Journal of Computer Integrated Manufacturing, 29(5): 535-560.

Emrah Akyar, H. A., Serkan Ali Duzce. 2012 A new method for ranking triangular fuzzy numbers. International Journal of Uncertainty Fuzziness and Knowledge-Based Systems 13.

Erer, Ö. G. D. B. 2018. Reasons And Risks Of Outsourcing In Business. Social Sciences Studies Journal, 4(27): 6114-6124.

Feng Lin, X. Z., Zengxia Cai. 2015. Approximation Algorithms for Scheduling with Rejection on Two Unrelated Parallel Machines. International Journal of Advanced Computer Science and Applications, 6.

Göksu, N. Ö. 2015. Bulanık Ortamda Ortak Teslim Tarihli Tek Makine Çizelgeleme Problemleri için Bir Karar Destek Sistemi. Unpublished Master of Science, Gazi Üniversitesi, Ankara.

He, L., Guijt, A., De Weerdt, M., Xing, L., & Yorke-Smith, N. 2019. Order acceptance and scheduling with sequence-dependent setup times: A new memetic algorithm and benchmark of the state of the art. Computers & Industrial Engineering, 138.

Jiang, D., & Tan, J. 2016. Scheduling with job rejection and nonsimultaneous machine available time on unrelated parallel machines. Theoretical Computer Science, 616: 94-99.

Kahraman, C. G., M.;Kabak, Ö. . 2006. Applications of fuzzy sets in industrial engineering: A topical classification. StudFuzz, 201: 55.

Karakaş, E., & Özpalamutçu, H. 2019. A genetic algorithm for fuzzy order acceptance and scheduling problem. An International Journal of Optimization and Control: Theories & Applications (IJOCTA), 9(2).

Kong, M., Pei, J., Liu, X., Lai, P.-C., & Pardalos, P. M. 2020. Green manufacturing: Order acceptance and scheduling subject to the budgets of energy consumption and machine launch. Journal of Cleaner Production, 248.

Koyuncu, E. 2017. A Fuzzy Mathematical Model for Order Acceptance and Scheduling Problem. International Journal of Mathematical and Computational Sciences, 11.

Lei, D., & Guo, X. 2015. A parallel neighborhood search for order acceptance and scheduling in flow shop environment. International Journal of Production Economics, 165: 12-18.

Lei He, A. G., Mathijs De Weerdt, Lining Xing, Neil Yorke-Smith. 2019. Order Acceptance and Scheduling with Sequence-dependent Setup Times: A New Memetic Algorithm and Benchmark of the State of the Art. Computers & Industrial Engineering.

Li, X., Ventura, J. A., & Bunn, K. A. 2020. A joint order acceptance and scheduling problem with earliness and tardiness penalties considering overtime. Journal of Scheduling: 1-20.

Liu, P., & Lu, X. 2020. New approximation algorithms for machine scheduling with rejection on single and parallel machine. Journal of Combinatorial Optimization, 40(4): 929-952.

Mitchell, M. 1998. L.D. Davis, Handbook of Genetic Algorithms. Artificial Intelligence, 100.

Naderi, B., & Roshanaei, V. 2020. Branch-Relax-and-Check: A tractable decomposition method for order acceptance and identical parallel machine scheduling. European Journal of Operational Research, 286(3): 811-827.

Nenad Perši, V. D. 2008. Conceptual Modelling Of Continuous Discrete Production Systems. 42000 Varaždin, Pavlinska 2, Croatia: University of Zagreb, Faculty of Organization and Informatics.

Nicholas Hall, M. E. P. 2004. Learning sensitivity of schedules by analyzing the search process Journal of Scheduling.

Oguz, C. S. S., F.; Bilgintürk Yalçın, Zehra. 2010. Order acceptance and scheduling decisions in make-to-order systems. International Journal of Production Economics, 125(1): 200-211.

P., C. L. O. 2003. An Elitist Genetic Algorithm For Multiobjective Optimization, Metaheuristics: Computer Decision-Making: 217-236: Kluwer Academic Publishers B.V.

Paksoy, S. 2007. Genetik Algoritma İle Proje Çizelgeleme. Cukurova University, Adana.

Palakiti, V. P. M., Usha. 2018. Order Acceptance and scheduling parallel machine environment with weighted completion Time. European Journal Industrial Engineering, 12.

Rakesh Kumar, G. G., Rajesh Kumar. 2013. Novel Crossover Operator for Genetic Algorithm for Permutation Problems. International Journal of Soft Computing and Engineering (IJSCE), 3(2).

Rom, W. O., & Slotnick, S. A. 2009. Order acceptance using genetic algorithms. Computers & Operations Research, 36(6): 1758-1767.

Sahebe Esfandiari, H. M. a. S. E. 2019. Coordination of Order Acceptance, Scheduling, and Pricing Decisions in Unrelated Parallel Machine Scheduling. International Journal of Industrial Engineering & Production Research (2019) 30: 195-205.

Shabtay, D., Gaspar, N., & Kaspi, M. 2012. A survey on offline scheduling with rejection. Journal of Scheduling, 16(1): 3-28.

Sleit, A. S., Maha; Almobaideen, Wesam. 2016. A Two Phase Fuzzy System For Edge Detection, Int'l Conf. IP, Comp. Vision, and Pattern Recognition CSREA Press.

Slotnick, S. A. 2011. Order acceptance and scheduling: A taxonomy and review. European Journal of Operational Research, 212(1): 1-11.

Thevenin, S., Zufferey, N., & Widmer, M. 2016. Order acceptance and scheduling with earliness and tardiness penalties. Journal of Heuristics, 22(6): 849-890.

Tuş, A. 2006. Bulanik Dogrusal Programlama Ve Bir Üretim Planlamasinda Uygulama Örneği. Unpublished Master of Science, Pamukkale Üniversitesi, Denizli.

Wang, B., & Wang, H. 2018. Multiobjective Order Acceptance and Scheduling on Unrelated Parallel Machines with Machine Eligibility Constraints. Mathematical Problems in Engineering, 2018: 1-12.

Wang, S., & Ye, B. 2018. Exact methods for order acceptance and scheduling on unrelated parallel machines. Computers & Operations Research, 104: 159-173.

Wang, X., Huang, G., Hu, X., & Edwin Cheng, T. C. 2015. Order acceptance and scheduling on two identical parallel machines. Journal of the Operational Research Society, 66(10): 1755-1767.

Wang, X., Xie, X., & Cheng, T. C. E. 2013. Order acceptance and scheduling in a two-machine flowshop. International Journal of Production Economics, 141(1): 366-376.

Wu, G.-H., Cheng, C.-Y., Yang, H.-I., & Chena, C.-T. 2018. An improved water flow-like algorithm for order acceptance and scheduling with identical parallel machines. Applied Soft Computing, 71: 1072-1084.

Yao, J.-S., & Wu, K. 2000. Ranking fuzzy numbers based on decomposition principle and signed distance. Fuzzy Sets and Systems, 116(2): 275-288.

Yapici, M. M. 2012. Genetik Algoritma Kullanilarak Ders Çizelgeleme Yaziliminin Geliştirilmesi. Unpublished Yüksek Lisans Tezi, Gazi Üniversitesi, Ankara.

Yavari, M., Marvi, M., & Akbari, A. H. 2019. Semi-permutation-based genetic algorithm for order acceptance and scheduling in two-stage assembly problem. Neural Computing and Applications, 32(8): 2989-3003.

Yılmaz, H. 2010. Doğrusal Programlama Tekniği ile Üretim Planlamasinin Mobilya Sektöründe Uygulanmasi. Unpublished Yüksek Lisans Tezi, Süleyman Demirel Üniversitesi, Isparta.

Zadeh, L. A. 1965. Fuzzy sets. Information and Control, 8(3): 338-353.

Zandieh, M., & Roumani, M. 2017. A biogeography-based optimization algorithm for order acceptance and scheduling. Journal of Industrial and Production Engineering, 34(4): 312-321.

Zhong, X., Ou, J., & Wang, G. 2014. Order acceptance and scheduling with machine availability constraints. European journal of operational research, 232(3): 435-441.

CURRICULUM VITAE

M.Zühal E. BARAK took her Bachelor's Degree in Industrial Engineering from Galatasaray University, in İstanbul. She graduated from Çukurova University with a Master of Science in Industrial Engineering. Presently, she is a candidate for Doctor of Philosophy in Industrial Engineering Department at Çukurova University, in Adana. She has also 10 years of work experience in the Metal Recycle Business.

APPENDICES

Appendix A The GADOC Execution Performance with Different Initial Population and Iteration Number

Appendix A-1 For n=10 m=2 The GADOC Execution Performance with Different Initial Population and Iteration Number

10 Order on 2 Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	102	1000	1,32	0,1774
100	20	105	2000	3,54	0,1532
100	50	113	5000	7,56	0,0887
100	100	113	10.000	12,74	0,0887
100	500	113	50.000	72,09	0,0887
100	1000	111	100.000	119,4	0,1048
100	2000	113	200.000	331,29	0,1452
200	100	113	20.000	21,6	0,129
200	1000	113	200.000	306,75	0,0887
500	100	113	50.000	54,45	0,0887
500	1000	113	500.000	766,91	0,1048
1000	1000	113	1.000.000	1175,40	0,1452
1000	500	113	500.000	574	0,129
1000	400	113	400.000	443,75	0,0887
1000	300	113	300.000	343,11	0,0887
1000	200	113	200.000	204,46	0,0887
1000	100	113	100.000	102,33	0,0887
1000	50	113	50.000	53,59	0,0887
1000	10	113	10.000	13,69	0,0887
2000	10	113	20.000	24,78	0,0887
2000	20	113	40.000	41,04	0,0887
2000	30	113	60.000	80,98	0,0887
2000	50	113	100.000	107,46	0,0887
2000	100	113	200.000	302,740	0,0887
2000	100	113	200.000	276,51	0,0887
2000	500	113	1.000.000	1039,400	0,0887
2000	1000	113	2.000.000	2567,100	0,0887
3000	100	113	300.000	416,08	0,0887
5000	100	113	500.000	698,83	0,0887

Appendix A-2 For n=15 m=2 GADOC Execution Performance with Different Initial Population and Iteration Number

15 Order on 2 Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	150	1000	1,99	0.1379
100	20	150	2000	4,44	0.1379
100	50	153	5000	9,57	0.1207
100	100	153	10.000	19,63	0.1207
100	100	153	10.000	21,95	0.1207
100	500	155,75	50.000	92,67	0.1049
100	1000	156,5	100.000	187,54	0.1006
100	2000	156,5	200.000	376,45	0.1006
200	100	153,37	20.000	47,33	0.1185
200	1000	158	200.000	342,64	0.0920
300	1000	158	300.000	532,46	0.0920
400	1000	158	400.000	756,21	0.0920
500	1000	158	500.000	964,64	0.0920
500	100	158	50.000	114,2	0.0920
1000	10	153	10.000	18,01	0.1207
1000	50	156,5	50.000	92,79	0.1006
1000	100	155,75	100.000	169,47	0.1049
1000	100	158	100.000	225,74	0.0920
1000	200	158	200.000	345,51	0.0920
1000	300	158	300.000	523,64	0.0920
1000	400	158	400.000	676,21	0.0920
1000	500	158	500.000	847,30	0.0920
1000	1000	158	1.000.000	1794,50	0.0920
2000	10	156	20.000	49,61	0.1034
2000	20	156	40.000	102,61	0.1034
2000	30	156,5	60.000	134,64	0.1006
2000	50	156,5	100.000	163,43	0.1006
2000	100	158	200.000	305,84	0.0920
2000	100	158	200.000	441,84	0.0920
2000	500	158	1.000.000	1594,50	0.0920
2000	1000	158	2.000.000	3077,70	0.0920
3000	100	158	300.000	536,46	0.0920
5000	100	158	500.000	936,45	0.0920

Appendix A-3 For n=50 m=2 GADOC Execution Performance with Different Initial Population and Iteration Number

50 Order on 2 Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	386	1000	3,88	0.2956
100	20	388	2000	4,49	0.2920
100	50	392	5000	21,82	0.2847
100	100	396	10.000	46,97	0.2773
100	500	398	50.000	236,45	0.2737
100	1000	398	100.000	463,64	0.2737
100	2000	402	200.000	835,54	0.2664
200	100	398	20.000	91,96	0.2737
200	1000	402	200.000	835,67	0.2664
300	1000	404	300.000	1136,70	0.2628
400	1000	405	400.000	1496,39	0.2609
500	100	402	50.000	233,70	0.2664
500	1000	409	500.000	2556,88	0.2536
1000	10	404	10.000	30,38	0.2628
1000	50	405	50.000	279,99	0.2609
1000	100	407	100.000	491,51	0.2573
1000	200	409	200.000	865,65	0.2536
1000	300	410,5	300.000	1021,63	0.2509
1000	400	412,5	400.000	1125,45	0.2477
1000	500	415	500.000	1433,41	0.2427
1000	1000	415	1.000.000	4863,67	0.2427
2000	10	394	20.000	90,71	0.2810
2000	20	399,06	40.000	168,48	0.2718
2000	30	399,06	60.000	271,04	0.2718
2000	50	402	100.000	430,10	0.2664
2000	100	407	200.000	860,80	0.2573
2000	500	417	1.000.000	4304,36	0.2391
2000	1000	417	2.000.000	9208,50	0.2391
3000	100	410,5	300.000	1398,32	0.2509
5000	100	415	500.000	2486,60	0.2477
20.000	100	417	2.000.000	8383,10	0.2391

Appendix A-4 For n=10 m=3 GADOC Execution Performance with Different Initial Population and Iteration Number

10 Order on 3 Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	106	1000	3,03	0.1492
100	20	109	2000	3,19	0.1210
100	50	111	5000	10,32	0.1048
100	100	113	10.000	19,37	0.0887
100	500	113	50.000	87,16	0.0887
100	1000	113	100.000	174,33	0.0887
100	2000	113	200.000	366,84	0.0887
200	100	113	20.000	37,45	0.0887
200	1000	113	200.000	348,66	0.0887
300	1000	113	300.000	524,90	0.0887
400	1000	113	400.000	686,12	0.0887
500	100	113	50.000	78,65	0.0887
500	1000	113	500.000	877,20	0.0887
1000	10	113	10.000	14,99	0.0887
1000	50	113	50.000	72,36	0.0887
1000	100	113	100.000	165,23	0.0887
1000	200	113	200.000	328,82	0.0887
1000	300	113	300.000	475,00	0.0887
1000	400	113	400.000	619,70	0.0887
1000	500	113	500.000	764,26	0.0887
1000	1000	113	1.000.000	1874,40	0.0887
2000	10	113	20.000	33,80	0.0887
2000	20	113	40.000	61,35	0.0887
2000	30	113	60.000	76,55	0.0887
2000	50	113	100.000	146,12	0.0887
2000	100	113	200.000	263,65	0.0887
2000	100	113	200.000	321,35	0.0887
2000	500	113	1.000.000	1098,46	0.0887
2000	1000	113	2.000.000	2263,19	0.0887
2000	1000	113	2.000.000	3480,91	0.0887
3000	100	113	300.000	654,89	0.0887
5000	100	113	500.000	819,23	0.0887

Appendix A-5 For n=15 m=3 GADOC Execution Performance with Different Initial Population and Iteration Number

15 Order on 3 Parallel Machine					
Initial Populatio	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	143	1000	2,43	0.1782
100	20	146	2000	3,33	0.1609
100	50	148	5000	9,57	0.1494
100	100	148	10.000	12,42	0.1494
100	500	148	50.000	119,75	0.1494
100	1000	148	100.000	245,50	0.1494
100	2000	149	200.000	491,66	0.1437
200	100	148	20.000	30,62	0.1494
200	1000	148	200.000	333,85	0.1494
300	1000	148	300.000	346,50	0.1494
400	1000	149	400.000	401,89	0.1437
500	100	150	50.000	85,08	0.1379
500	1000	150	500.000	454,67	0.1379
1000	10	146	10.000	13,70	0.1609
1000	50	148	50.000	68,11	0.1494
1000	100	150	100.000	172,77	0.1379
1000	200	150	200.000	326,80	0.1379
1000	300	150	300.000	533,20	0.1379
1000	400	150	400.000	670,80	0.1379
1000	500	152	500.000	744,00	0.1264
1000	1000	150	1.000.000	948,24	0.1379
2000	10	150	20.000	40,62	0.1379
2000	20	150	40.000	61,05	0.1379
2000	30	150	60.000	133,65	0.1379
2000	50	152	100.000	175,21	0.1264
2000	100	152	200.000	367,50	0.1264
2000	500	153	1.000.000	1761,60	0.1207
2000	1000	153	2.000.000	3681,10	0.1207
3000	100	155,75	300.000	517,86	0.1049
5000	100	158	500.000	714,77	0.0920
10.000	50	156	500.000	736,62	0.1034
20.000	100	158	2.000.000	3239,80	0.0920

Appendix A-6 For n=50 m=3 GADOC Execution Performance with Different Initial Population and Iteration Number

50 Order on 3 Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	369,5	1000	3,88	0.3257
100	20	369,62	2000	4,49	0.3255
100	50	370	5000	21,82	0.3248
100	100	371,12	10.000	46,97	0.3227
100	500	372	50.000	236,45	0.3211
100	1000	386	100.000	463,64	0.2956
100	2000	388	200.000	835,54	0.2919
200	100	386	20.000	91,96	0.2956
200	1000	388	200.000	835,67	0.2919
300	1000	391	300.000	1136,70	0.2865
400	1000	402	400.000	1496,39	0.2664
500	100	402	50.000	233,70	0.2664
500	1000	407	500.000	2556,88	0.2573
1000	10	386	10.000	77,55	0.2956
1000	50	388	50.000	206,16	0.2920
1000	100	407	100.000	491,51	0.2573
1000	100	388	100.000	370,80	0.2920
1000	200	389	200.000	704,52	0.2901
1000	300	391	300.000	1073,00	0.2865
1000	400	391	400.000	1478,40	0.2865
1000	500	402	500.000	2827,50	0.2664
1000	1000	415	1.000.000	4863,67	0.2427
2000	10	394	20.000	90,71	0.2810
2000	20	399,06	40.000	168,48	0.2718
2000	30	399,06	60.000	271,04	0.2718
2000	50	402	100.000	430,10	0.2664
2000	100	407	200.000	860,80	0.2573
2000	500	417	1.000.000	4304,36	0.2391
2000	1000	417	2.000.000	9208,50	0.2391
3000	100	410,5	300.000	1398,32	0.2509
5000	100	415	500.000	2486,60	0.2477

Appendix A-7 For n=10 m=4 GADOC Execution Performance with Different Initial Population and Iteration Number

10 Order on 4 Parallel Machine					
Initial Population	Iteration Number	Obj	Total Solution	Execution Time(sec)	UBD
100	10	102	1000	3,32	0.1774
100	20	105	2000	5,54	0.1532
100	50	105	5000	9,56	0.1532
100	100	108	10.000	18,74	0.1290
100	100	112	10.000	15,45	0,0968
100	500	113	50.000	86,09	0.0887
100	1000	113	100.000	162,51	0.0887
100	1000	113	100.000	165,51	0.0887
100	2000	113	200.000	348,29	0.0887
200	100	113	20.000	27,60	0.0887
200	1000	113	200.000	128,40	0.0887
300	1000	113	300.000	169,06	0.0887
400	1000	111	400.000	181,86	0.0887
500	100	113	50.000	63,45	0.0887
500	1000	113	500.000	314,75	0.0887
1000	10	113	10.000	13,69	0.0887
1000	50	113	50.000	53,59	0.0887
1000	100	113	100.000	102,33	0.0887
1000	100	113	100.000	148,39	0.0887
1000	200	113	200.000	204,46	0.0887
1000	300	113	300.000	343,11	0.0887
1000	400	113	400.000	443,75	0.0887
1000	500	113	500.000	574,28	0.0887
1000	1000	113	1.000.000	770,91	0.0887
2000	10	113	20.000	24,78	0.0887
2000	20	113	40.000	41,04	0.0887
2000	30	113	60.000	80,98	0.0887
2000	50	113	100.000	107,46	0.0887
2000	100	113	200.000	302,74	0.0887
2000	100	113	200.000	289,51	0.0887
2000	500	113	1.000.000	1039,40	0.0887
2000	1000	113	2.000.000	2567,10	0.0887
2000	1000	113	2.000.000	1200,40	0.0887
3000	100	113	300.000	428,08	0.0887
5000	100	113	500.000	706,83	0.0887

Appendix A-8 For n=15 m=4 GADOC Execution Performance with Different Initial Population and Iteration Numbe

15 Order on 4 Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	132,75	1000	1,99	0.2370
100	20	132,75	2000	4,44	0.2370
100	50	133,37	5000	9,57	0.2335
100	100	133,37	10.000	18,63	0.2335
100	100	133,37	10.000	19,95	0.2335
100	500	137,2	50.000	87,67	0.2114
100	1000	137,2	100.000	187,54	0.2114
100	1000	140,57	100.000	186,54	0.1921
100	2000	140,57	200.000	374,45	0.1921
200	100	133,37	20.000	47,33	0.2335
200	1000	142	200.000	342,64	0.1839
300	1000	142	300.000	528,46	0.1839
400	1000	150	400.000	747,21	0.1379
500	100	137,2	50.000	109,20	0.2114
500	1000	150	500.000	923,64	0.1379
1000	10	150	10.000	10,01	0.1379
1000	50	150	50.000	86,79	0.1379
1000	100	150	100.000	134,47	0.1379
1000	100	150	100.000	223,74	0.1379
1000	200	153	200.000	313,51	0.1207
1000	300	153	300.000	503,64	0.1207
1000	400	153	400.000	598,21	0.1207
1000	500	153	500.000	786,30	0.1207
1000	1000	156,5	1.000.000	1671,50	0.1006
2000	10	153	20.000	49,61	0.1207
2000	20	156	40.000	101,61	0.1034
2000	30	156	60.000	132,64	0.1034
2000	50	156,5	100.000	156,43	0.1006
2000	100	156,5	200.000	431,84	0.1006
2000	100	156,5	200.000	307,84	0.1006
2000	500	156,5	1.000.000	1596,50	0.1006
2000	1000	158	2.000.000	3016,70	0.0920
2000	1000	158	2.000.000	2979,70	0.0920
3000	100	158	300.000	513,46	0.0920
5000	100	158	500.000	939,45	0.0920

Appendix A-9 For n=50 m=4 GADOC Execution Performance with Different Initial Population and Iteration Number

50 Order on 4 Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	357,4	1000	3,88	0.3478
100	20	368	2000	4,49	0.3285
100	50	369	5000	21,82	0.3266
100	100	369	10.000	46,97	0.3266
100	500	370,8	50.000	236,45	0.3233
100	1000	373,7	100.000	463,64	0.3180
100	2000	375	200.000	835,54	0.3156
200	100	373,7	20.000	91,96	0.3180
200	1000	375	200.000	835,67	0.3156
300	1000	375	300.000	1136,70	0.3156
400	1000	382	400.000	1496,39	0.3029
500	1000	382	500.000	2556,88	0.3029
500	100	375	50.000	233,70	0.3156
1000	10	382	10.000	30,38	0.3029
1000	50	382	50.000	279,99	0.3029
1000	100	383,4	100.000	491,51	0.3004
1000	100	382	100.000	491,51	0.3029
1000	200	383,4	200.000	865,65	0.3004
1000	300	395	300.000	1021,63	0.2792
1000	400	388,7	400.000	1125,45	0.2907
1000	500	389	500.000	1433,41	0.2901
1000	1000	399,06	1.000.000	4863,67	0.2717
2000	10	399,06	20.000	90,71	0.2717
2000	20	399,06	40.000	168,48	0.2717
2000	30	399,06	60.000	271,04	0.2717
2000	50	402	100.000	430,10	0.2664
2000	100	402	200.000	860,80	0.2664
2000	100	399,06	200.000	860,80	0.2717
2000	500	407	1.000.000	4304,36	0.2573
2000	1000	417	2.000.000	9208,50	0.2391
2000	1000	417	2.000.000	9208,50	0.2391
3000	100	402	300.000	1398,32	0.2664
5000	100	407	500.000	2486,60	0.2572

Appendix A-10 For n=10 m=5 GADOC Execution Performance with Different Initial Population and Iteration Number

10 Order on 5 Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	102	1000	3,32	0.1774
100	20	105	2000	5,54	0.1532
100	50	105	5000	9,56	0.1532
100	100	112	10.000	15,45	0,0968
100	100	108	10.000	18,74	0.1290
100	500	113	50.000	86,09	0.0887
100	1000	113	100.000	165,51	0.0887
100	1000	113	100.000	162,51	0.0887
100	2000	113	200.000	348,29	0.0887
200	100	113	20.000	27,60	0.0887
200	1000	113	200.000	128,40	0.0887
300	1000	113	300.000	169,06	0.0887
400	1000	111	400.000	181,86	0.0887
500	100	113	50.000	63,45	0.0887
500	1000	113	500.000	314,75	0.0887
1000	10	113	10.000	17,69	0.0887
1000	50	113	50.000	63,59	0.0887
1000	100	113	100.000	113,33	0.0887
1000	100	113	100.000	148,39	0.0887
1000	200	113	200.000	217,46	0.0887
1000	300	113	300.000	352,11	0.0887
1000	400	113	400.000	453,75	0.0887
1000	500	113	500.000	580,28	0.0887
1000	1000	113	1.000.000	770,91	0.0887
2000	10	113	20.000	27,78	0.0887
2000	20	113	40.000	48,04	0.0887
2000	30	113	60.000	87,98	0.0887
2000	50	113	100.000	114,46	0.0887
2000	100	113	200.000	316,74	0.0887
2000	100	113	200.000	289,51	0.0887
2000	500	113	1.000.000	1043,40	0.0887
2000	1000	113	2.000.000	2585,10	0.0887
2000	1000	113	2.000.000	1200,40	0.0887
3000	100	113	300.000	428,08	0.0887
5000	100	113	500.000	706,83	0.0887

Appendix A-10 For n=15 m=5 GADOC Execution Performance with Different Initial Population and Iteration Number

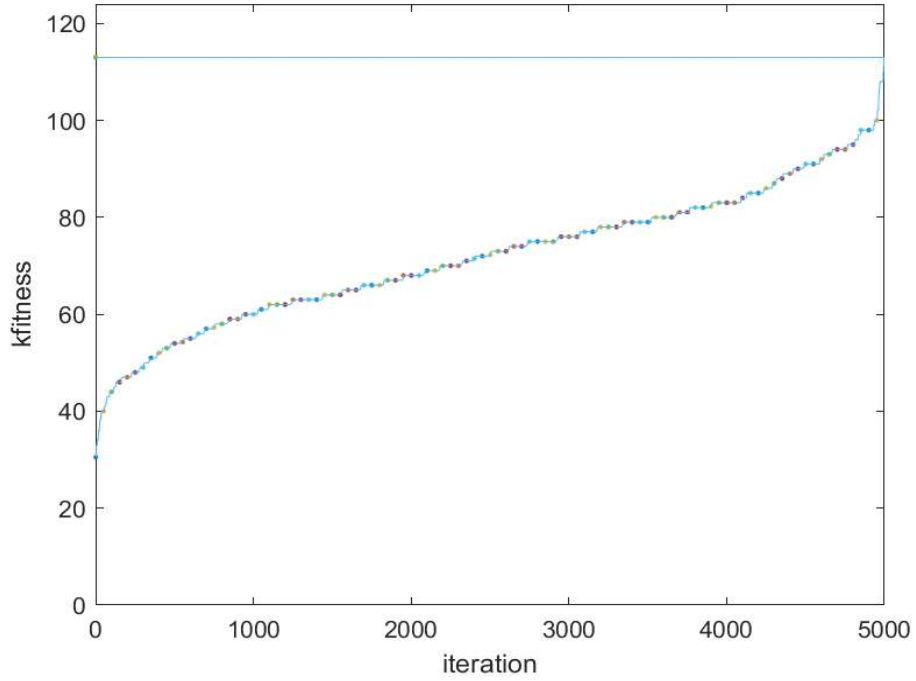
15 Order on 5 Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	132,75	1000	1,99	0.2370
100	20	132,75	2000	5,44	0.2370
100	50	133,37	5000	10,57	0.2335
100	100	133,37	10.000	23,95	0.2335
100	100	133,37	10.000	21,63	0.2335
100	500	137,2	50.000	99,67	0.2114
100	1000	137,2	100.000	195,54	0.2114
100	1000	140,57	100.000	192,54	0.1921
100	2000	140,57	200.000	387,45	0.1921
200	100	133,37	20.000	56,33	0.2335
200	1000	142	200.000	353,64	0.1839
300	1000	142	300.000	544,46	0.1839
400	1000	150	400.000	767,21	0.1379
500	100	137,2	50.000	127,20	0.2114
500	1000	150	500.000	979,64	0.1379
1000	100	150	100.000	238,74	0.1379
1000	1000	156,5	1.000.000	1853,50	0.1006
1000	10	150	10.000	15,01	0.1379
1000	50	150	50.000	98,79	0.1379
1000	100	150	100.000	159,47	0.1379
1000	200	153	200.000	346,51	0.1207
1000	300	153	300.000	553,64	0.1207
1000	400	153	400.000	666,21	0.1207
1000	500	153	500.000	827,30	0.1207
2000	10	153	20.000	53,61	0.1207
2000	20	156	40.000	111,61	0.1034
2000	30	156	60.000	141,64	0.1034
2000	50	156,5	100.000	168,43	0.1006
2000	100	156,5	200.000	454,84	0.1006
2000	100	156,5	200.000	313,84	0.1006
2000	500	156,5	1.000.000	1603,50	0.1006
2000	1000	158	2.000.000	3176,70	0.0920
2000	1000	158	2.000.000	3096,70	0.0920
3000	100	158	300.000	542,46	0.0920
5000	100	158	500.000	950,45	0.0920

Appendix A-10 For n=50 m=5 GADOC Execution Performance with Different Initial Population and Iteration Number

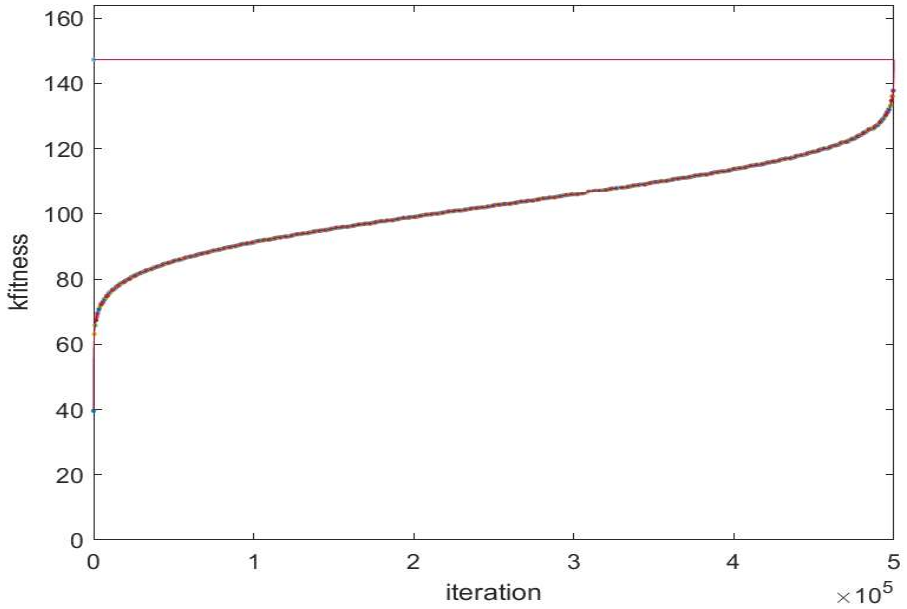
50 Order on 5 Parallel Machine					
Initial Population	**Iteration Number**	**Obj**	**Total Solution**	**Execution Time(sec)**	**UBD**
100	10	357,4	1000	3,88	0,3478
100	20	368	2000	4,49	0,3285
100	50	369	5000	21,82	0,3266
100	100	369	10.000	46,97	0,3266
100	100	369	10.000	46,9700	0,3266
100	500	370,8	50.000	236,45	0,3233
100	1000	373,7	100.000	463,64	0,318
100	1000	373,7	100.000	463,64000	0,3180
100	2000	375	200.000	835,54	0,3156
200	100	373,7	20.000	91,9600	0,318
200	1000	375	200.000	835,67000	0,3156
300	1000	375	300.000	1136,70000	0,3156
400	1000	382	400.000	1496,39000	0,3029
500	100	375	50.000	233,7000	0,3156
500	1000	382	500.000	2556,88000	0,3029
1000	10	381	10.000	30,38	0,3047
1000	50	382	50.000	279,99	0,3029
1000	100	382	100.000	491,51	0,3029
1000	100	382	100.000	491,5100	0,3029
1000	200	383,4	200.000	865,65	0,3004
1000	300	383,4	300.000	1021,63	0,3004
1000	400	388,7	400.000	1125,45	0,2907
1000	500	389	500.000	1.433,400	0,2901
1000	1000	399,06	1.000.000	4863,67000	0,2717
2000	10	395	20.000	90,7100	0,2792
2000	20	396	40.000	168,4848	0,2773
2000	30	396	60.000	271,0398	0,2773
2000	50	397	100.000	430,1000	0,2755
2000	100	399,06	200.000	860,8000	0,2717
2000	100	399,06	200.000	860,8000	0,2717
2000	500	402	1.000.000	4304,3600	0,2664
2000	1000	417	2.000.000	9208,5000	0,2391
2000	1000	417	2.000.000	9208,50000	0,2391
3000	100	402	300.000	1398,3200	0,2664
5000	100	407	500.000	2486,6000	0,2572

Appendix B- The GADOC Algorithm Fitness Function and Iteration Matlab Graphs

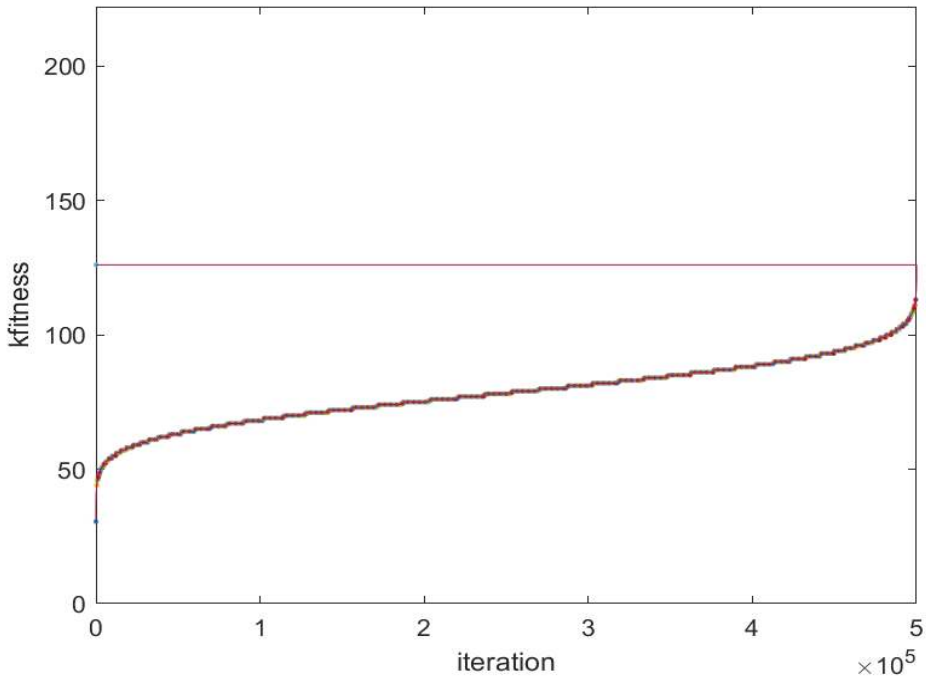
Appendix B-1 10 orders&2 machines Fitness Function and Iteration Matlab Graph



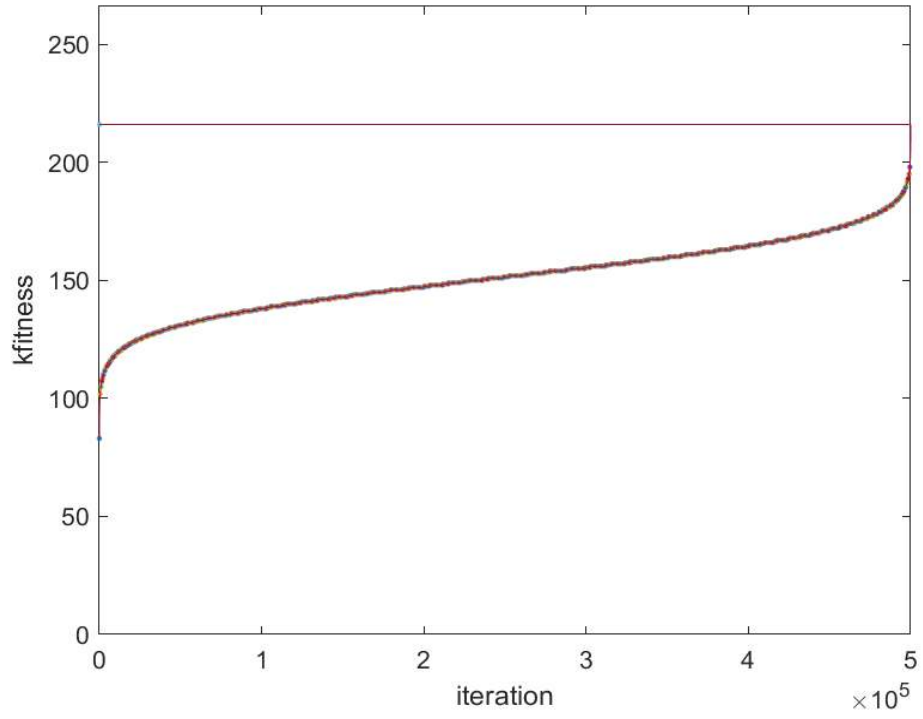
Appendix B-2 15 orders&2 machines Fitness Function and Iteration Matlab Graph



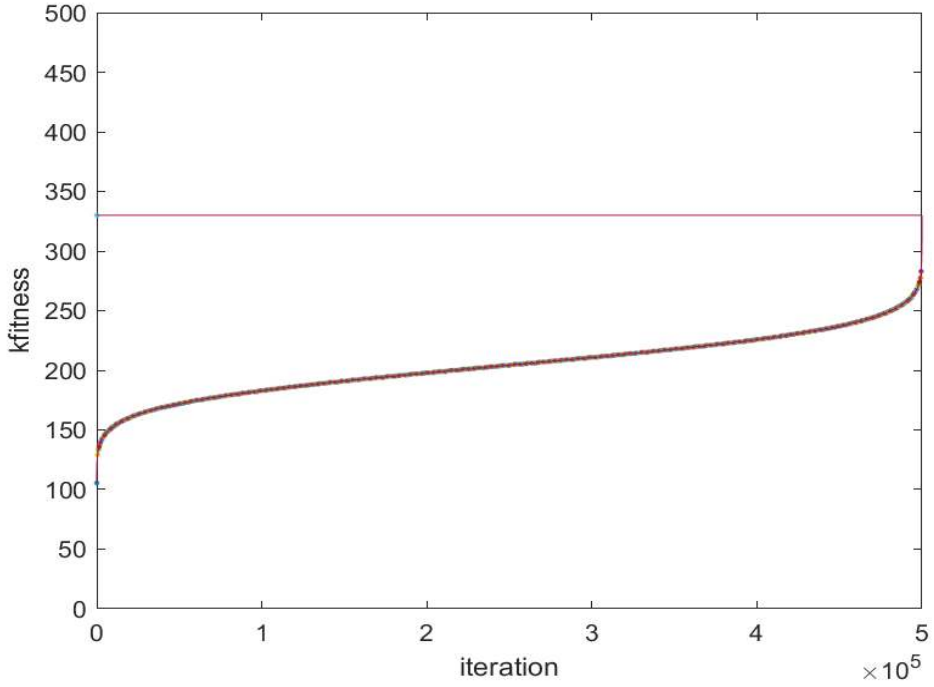
Appendix B-3 20 orders&2 machines Fitness Function and Iteration Matlab Graph



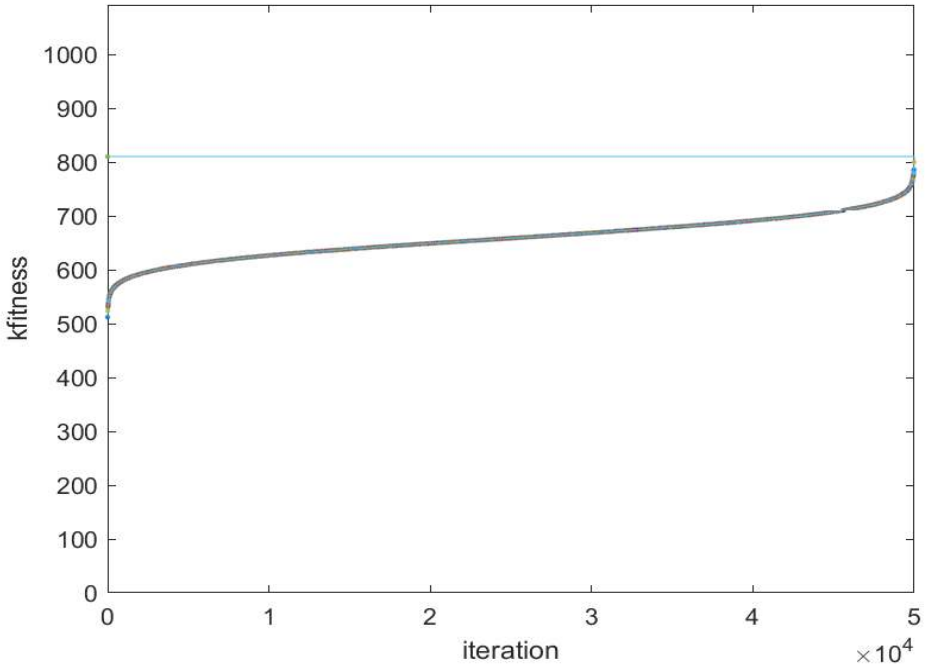
Appendix B-4 25 orders&2 machines Fitness Function and Iteration Matlab Graph



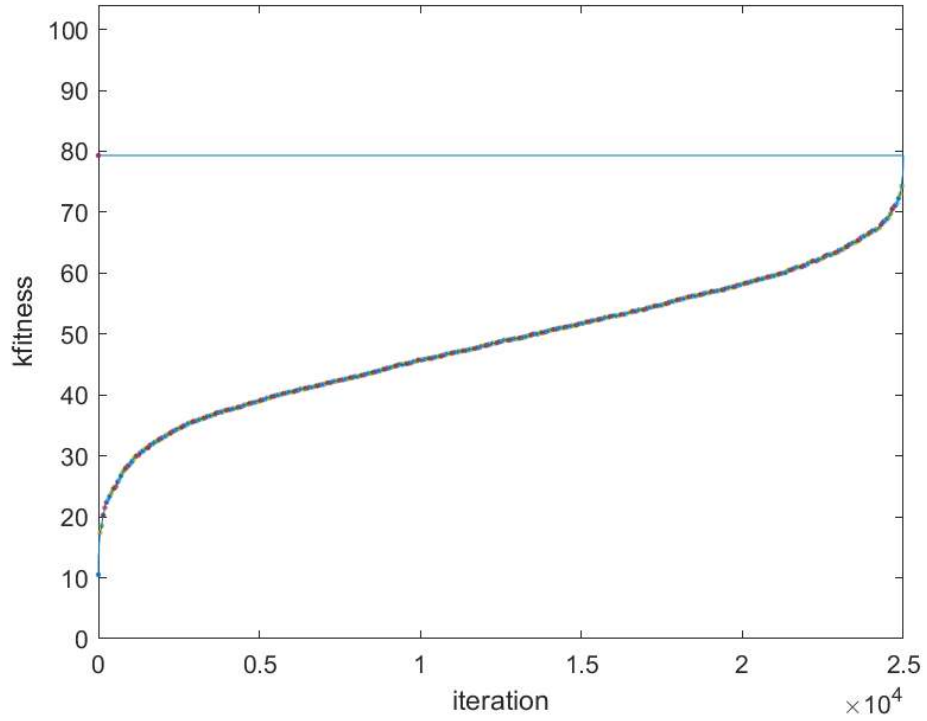
Appendix B-5 50 orders&2 machines Fitness Function and Iteration Matlab Graph



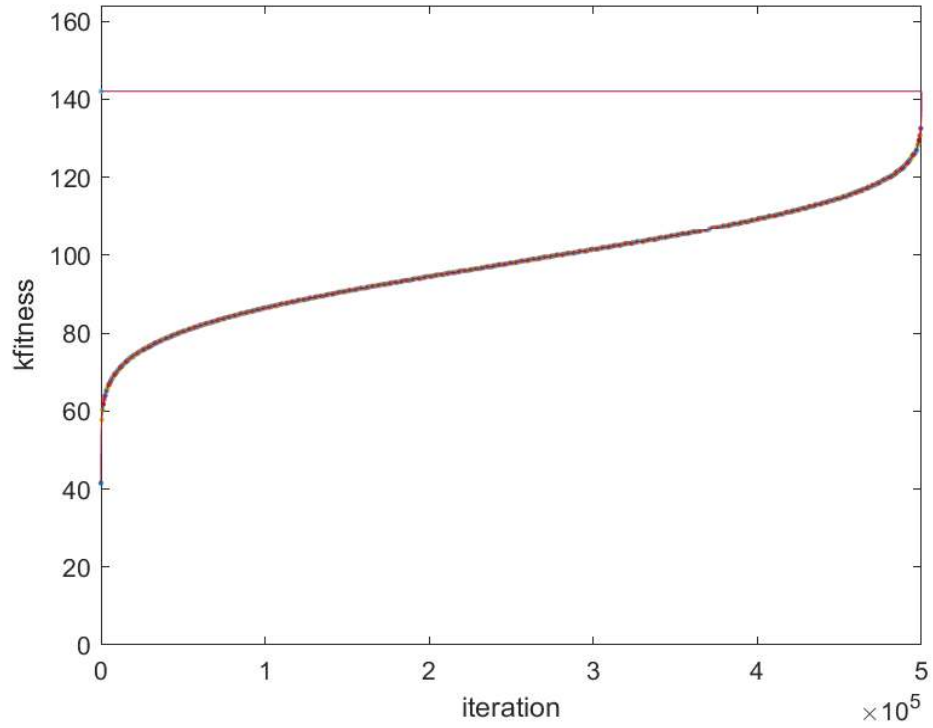
Appendix B-6 100 orders&2 machines Fitness Function and Iteration Matlab Graph



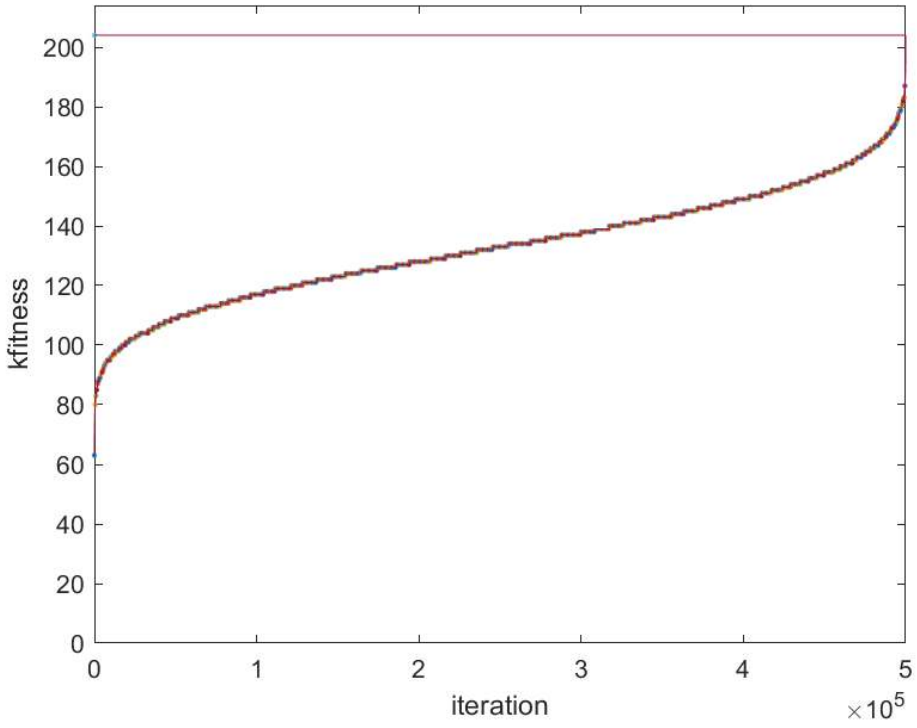
Appendix B-7 10 orders&3 machines Fitness Function and Iteration Matlab Graph



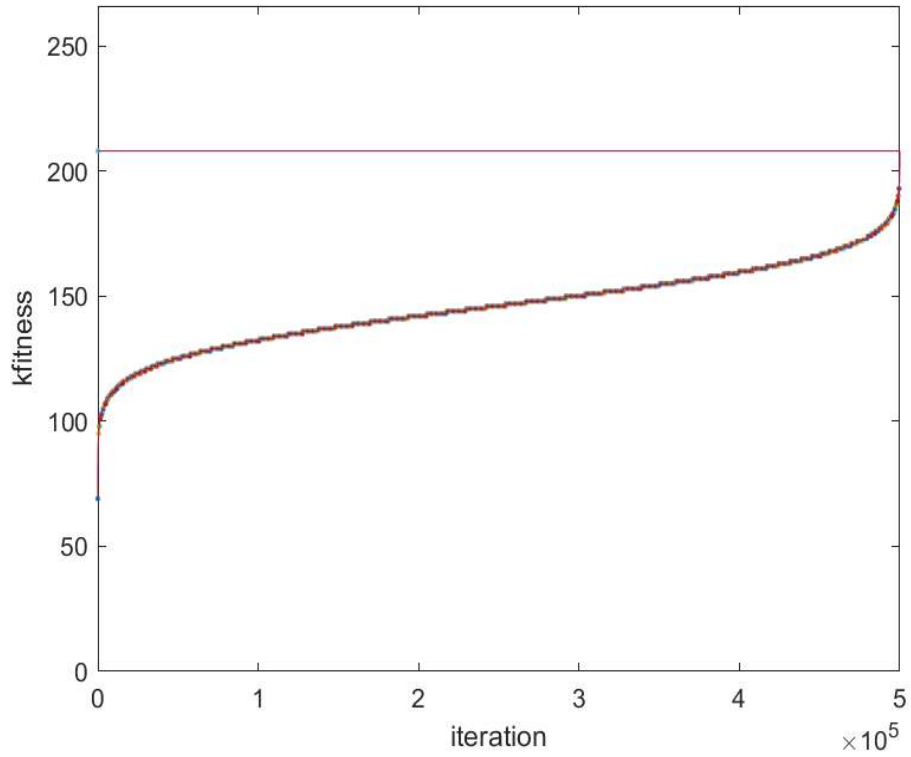
Appendix B-8 15 orders&3 machines Fitness Function and Iteration Matlab Graph



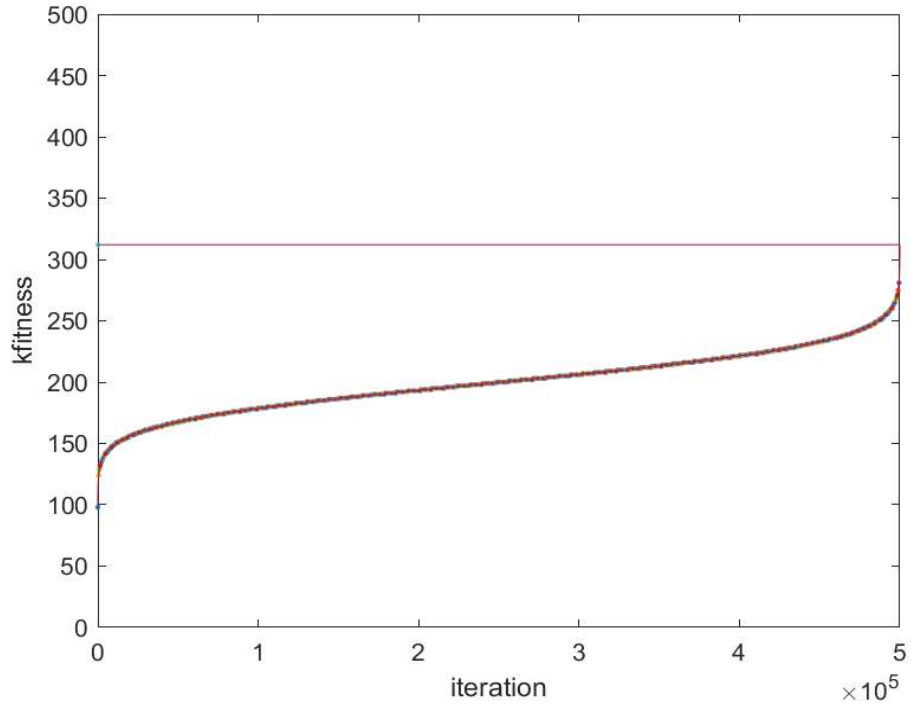
Appendix B-9 20 orders&3 machines Fitness Function and Iteration Matlab Graph



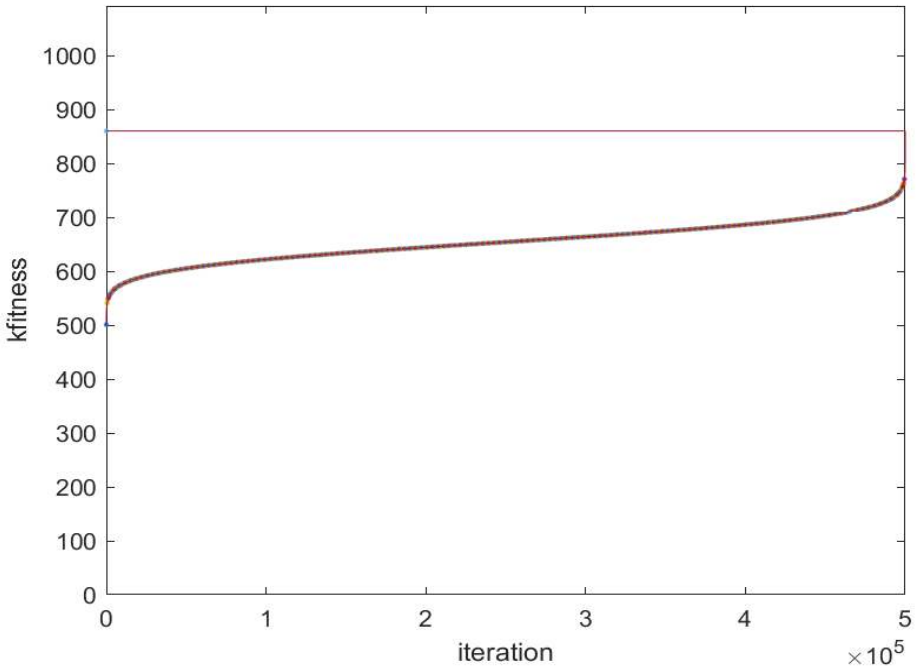
Appendix B-10 25 orders&3 machines Fitness Function and Iteration Matlab Graph



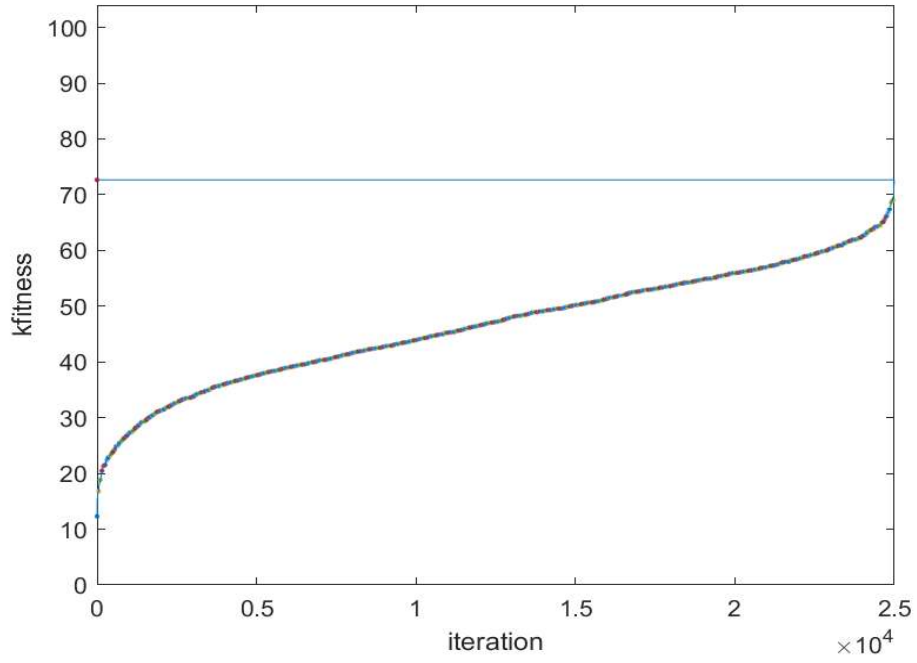
Appendix B-11 50 orders&3 machines Fitness Function and Iteration Matlab Graph



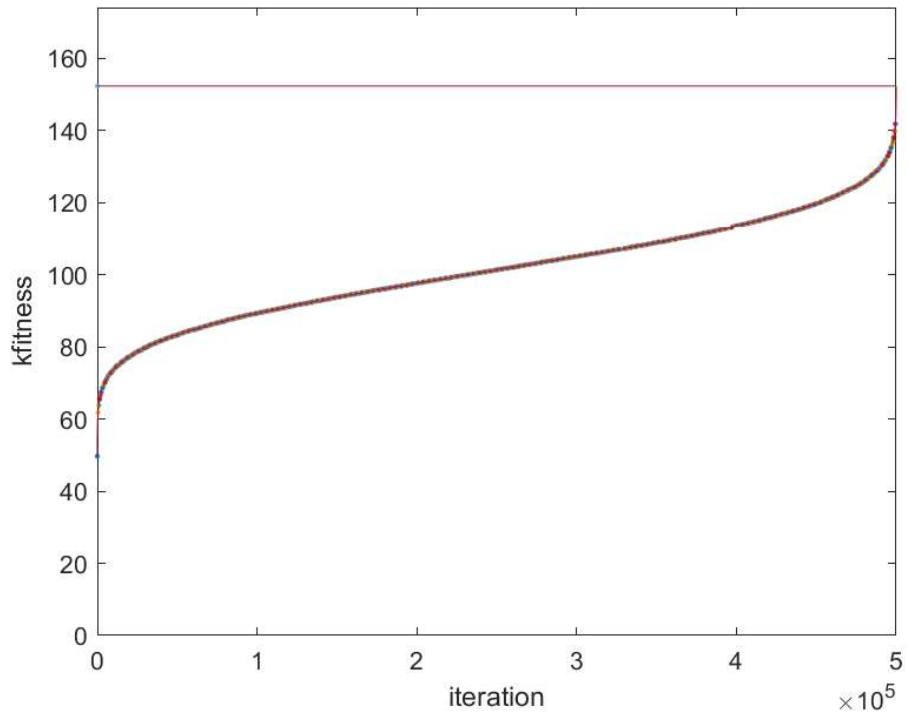
Appendix B-12 100 orders&3 machines Fitness Function and Iteration Matlab Graph



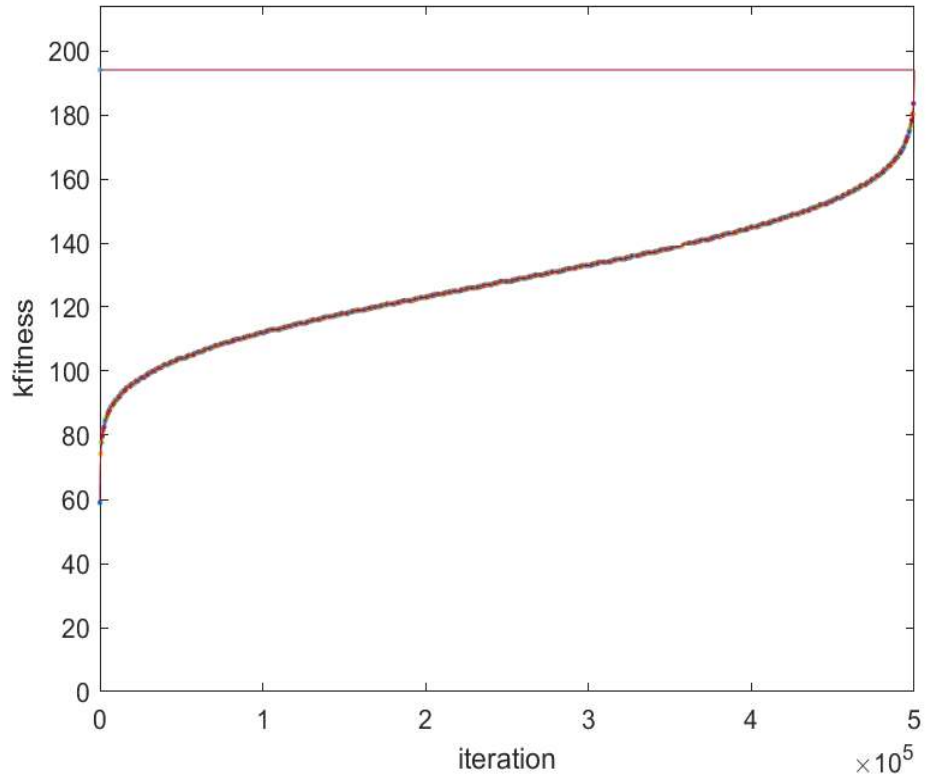
Appendix B-13 10 orders&4 machines Fitness Function and Iteration Matlab Graph



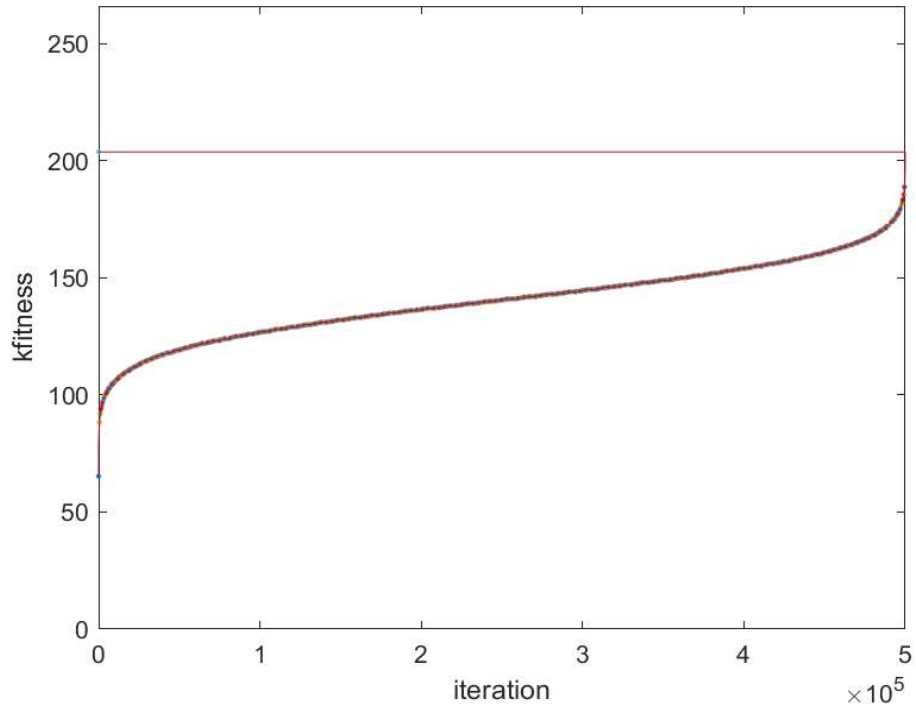
Appendix B-14 15 orders&4 machines Fitness Function and Iteration Matlab Graph



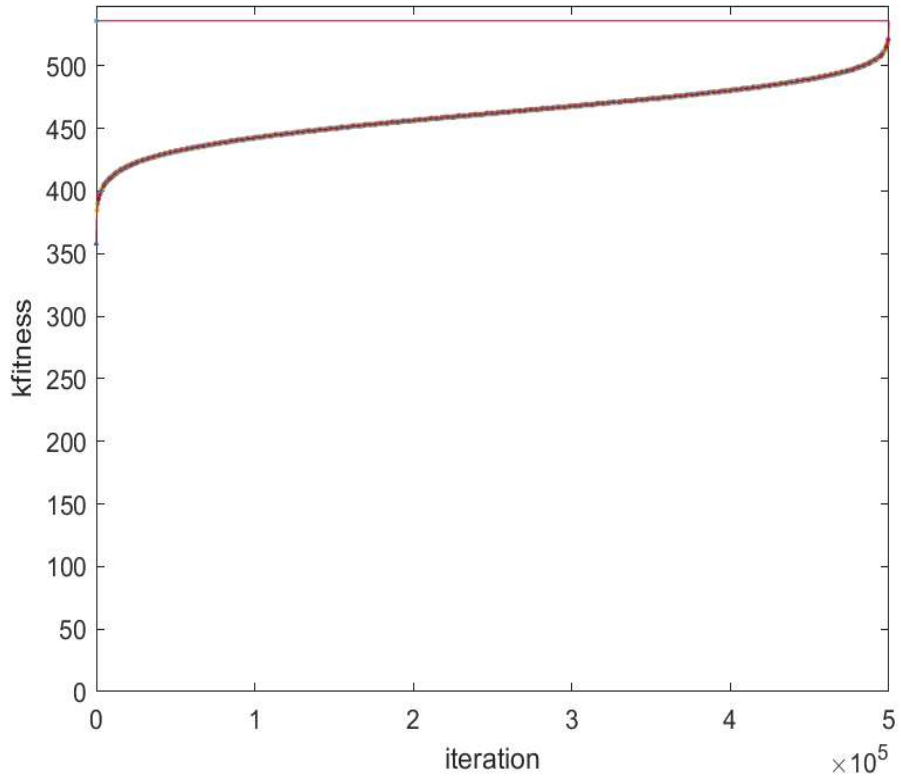
Appendix B-15 20 orders&4 machines Fitness Function and Iteration Matlab Graph



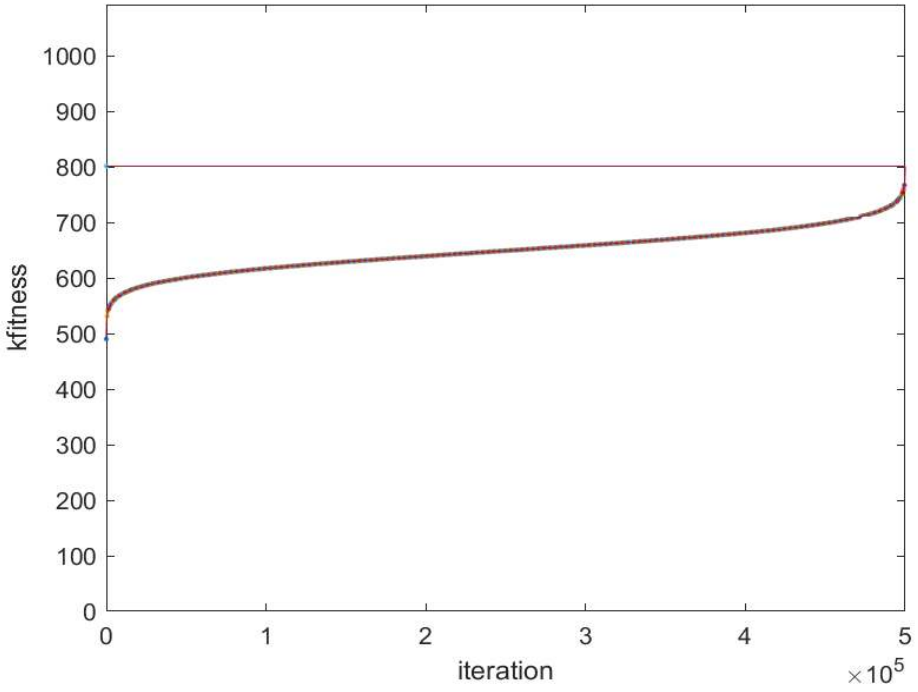
Appendix B-16 25 orders&4 machines Fitness Function and Iteration Matlab Graph



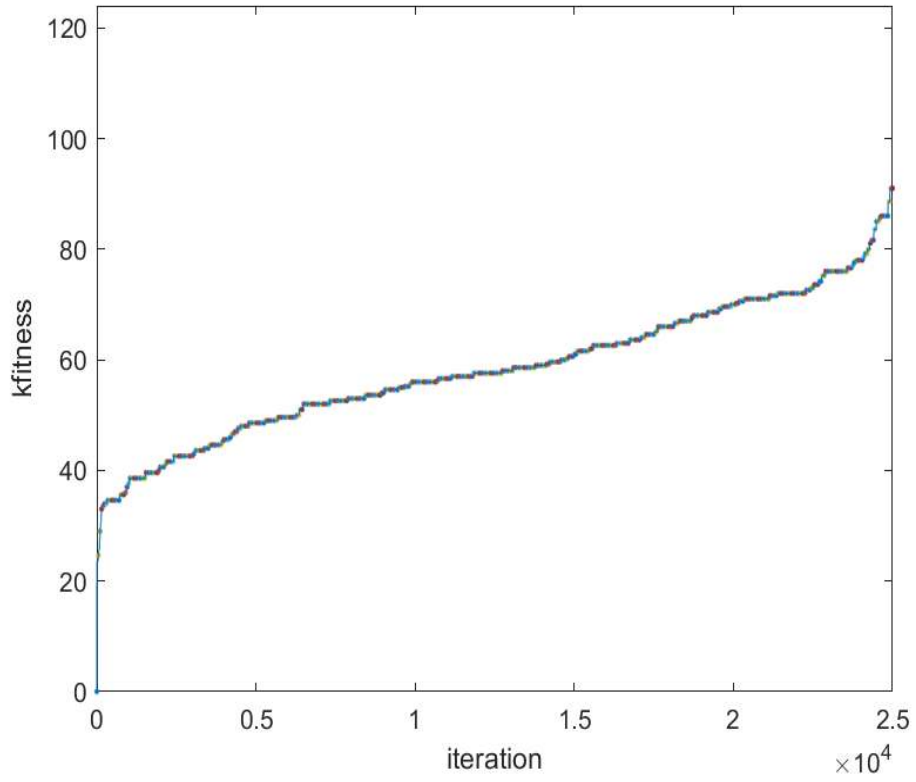
Appendix B-17 50 orders&4 machines Fitness Function and Iteration Matlab Graph



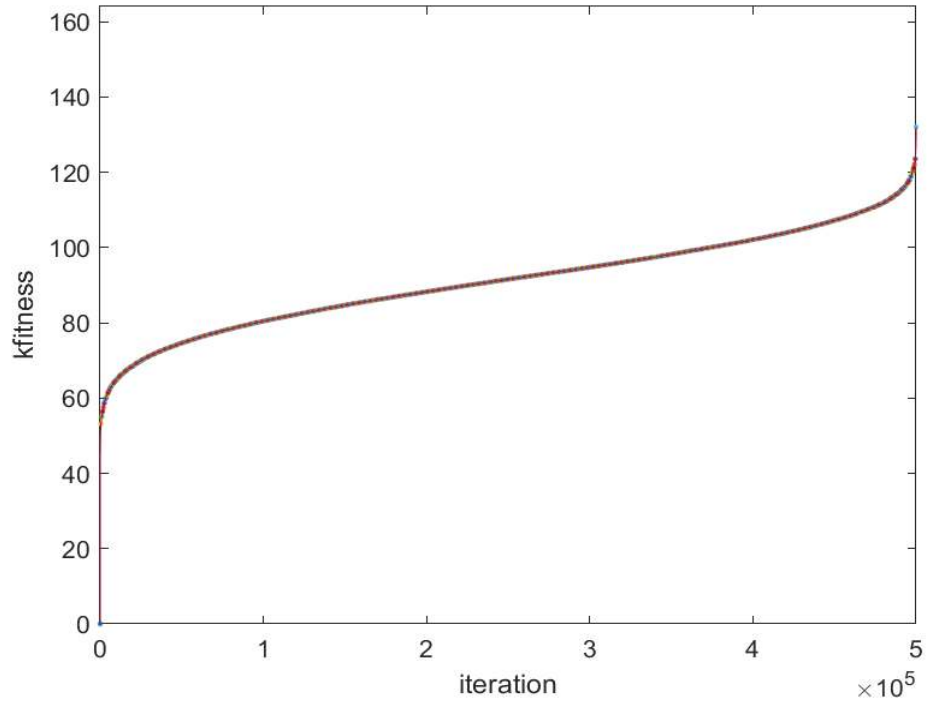
Appendix B-18 100 orders&4 machines Fitness Function and Iteration Matlab Graph



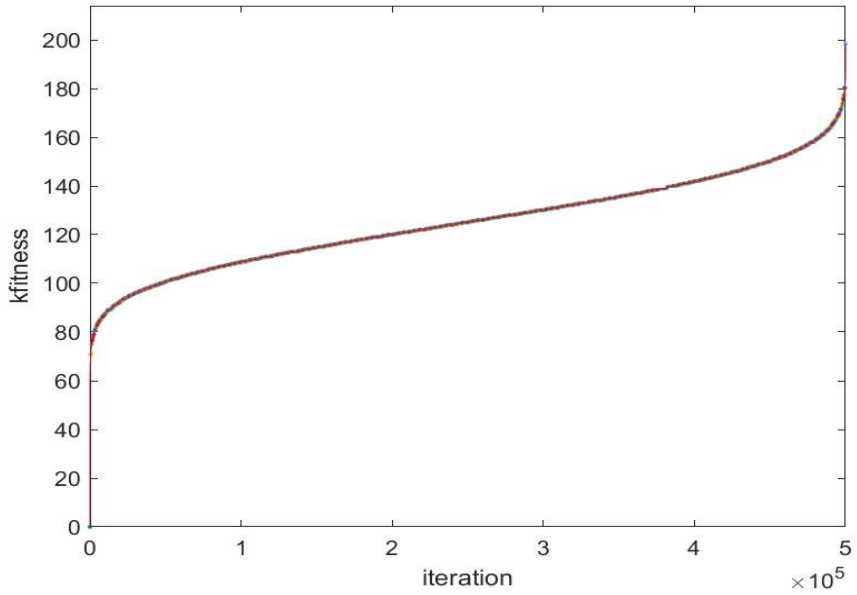
Appendix B-19 10 orders&5 machines Fitness Function and Iteration Matlab Graph



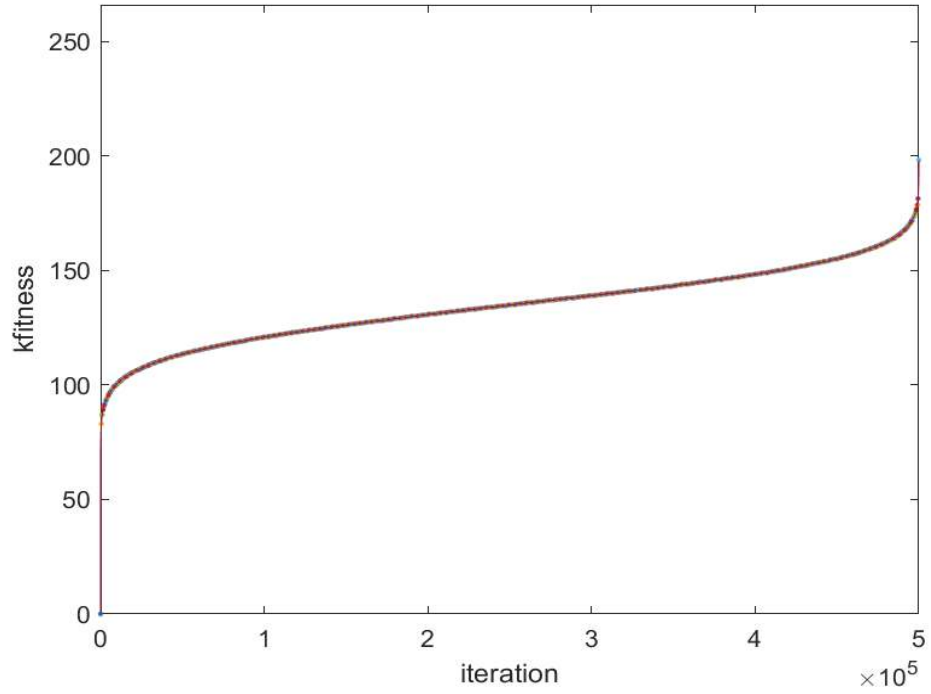
Appendix B-20 15 orders&5 machines Fitness Function and Iteration Matlab Graph



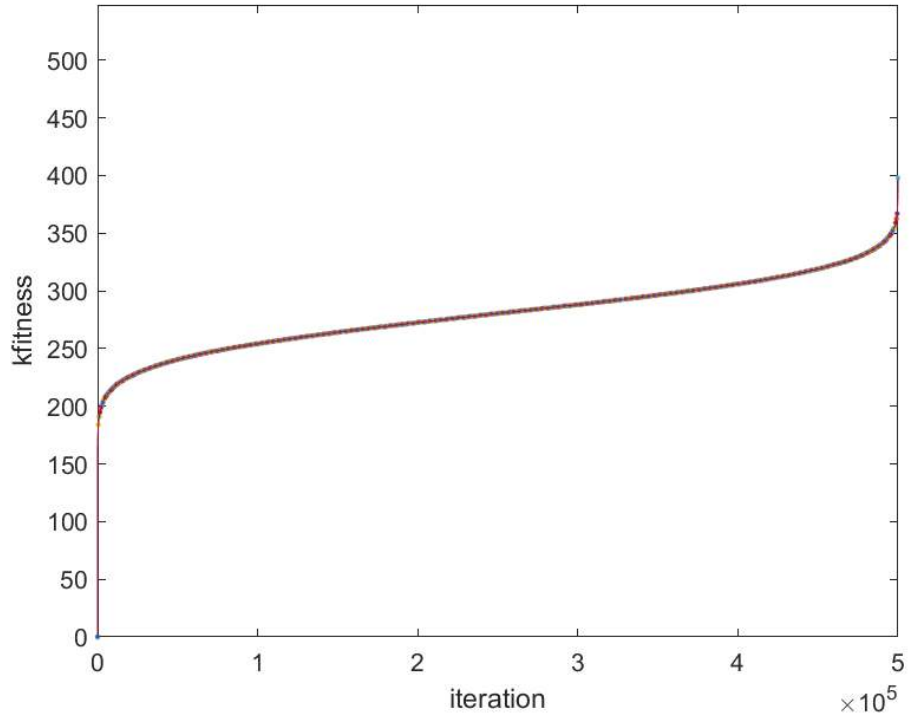
Appendix B-21 20 orders&5 machines Fitness Function and Iteration Matlab Graph



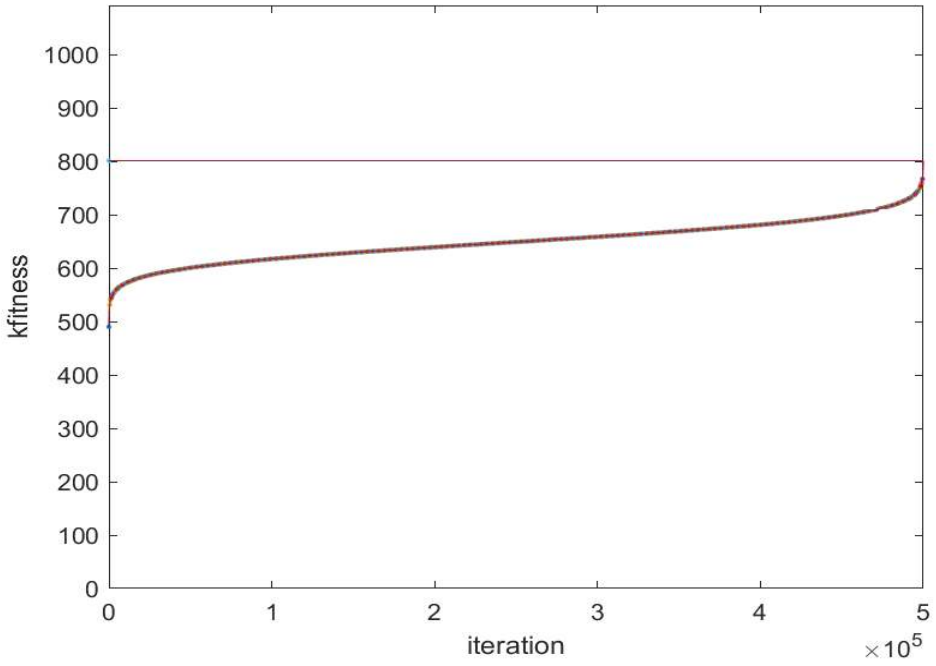
Appendix B-22 25 orders&5 machines Fitness Function and Iteration Matlab Graph



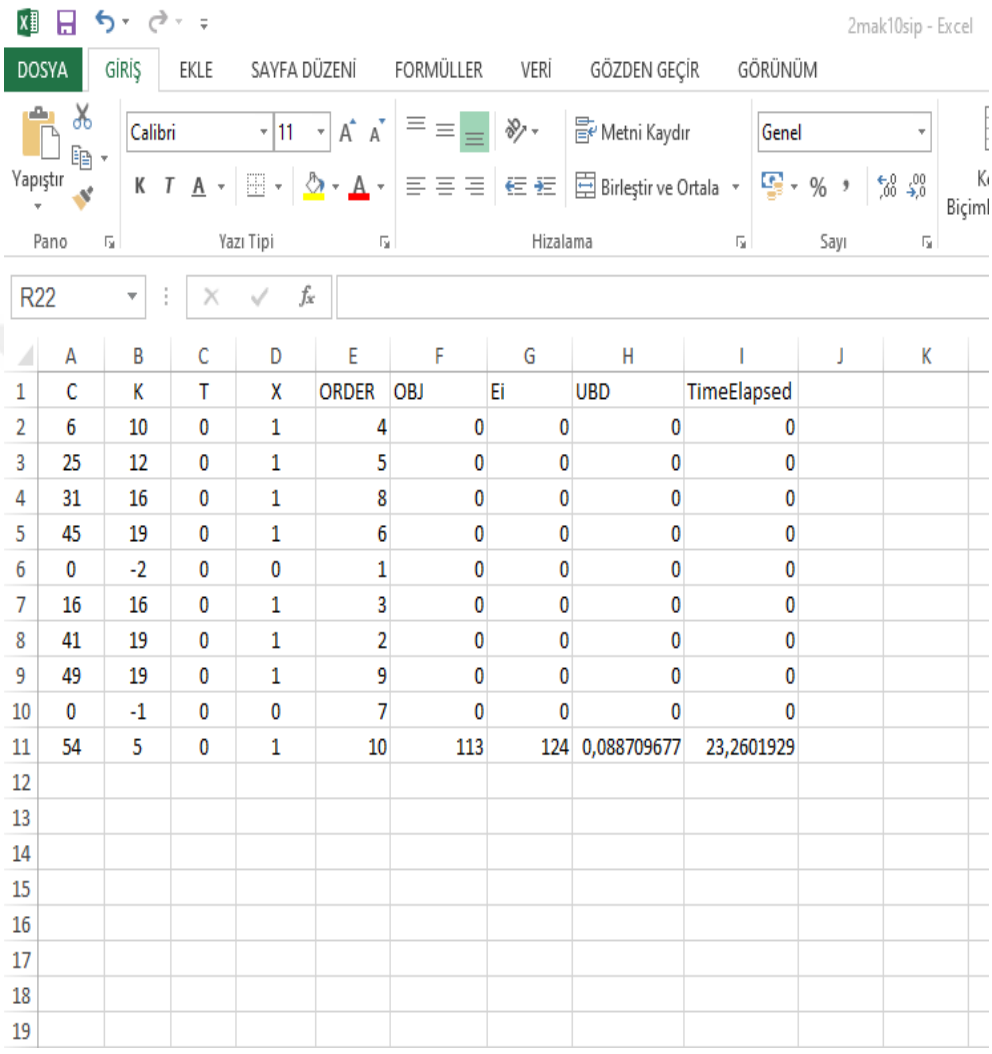
Appendix B-23 50 orders&5 machines Fitness Function and Iteration Matlab Graph



Appendix B-24 100 orders&5 machines Fitness Function and Iteration Matlab Graph



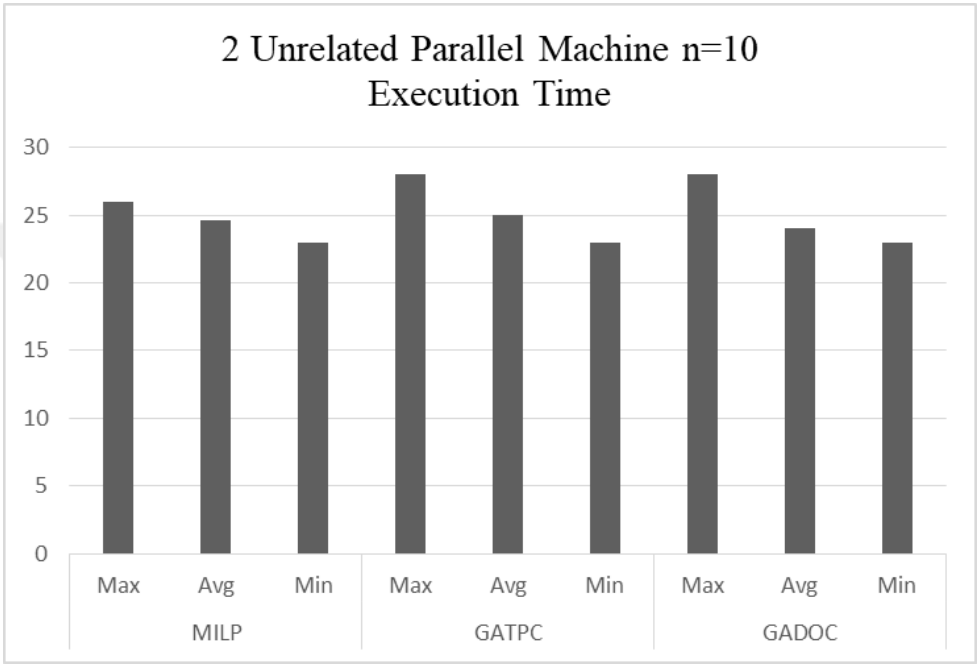
Appendix C- Matlab Result Excel File Example For 10 Order on 2 Machine



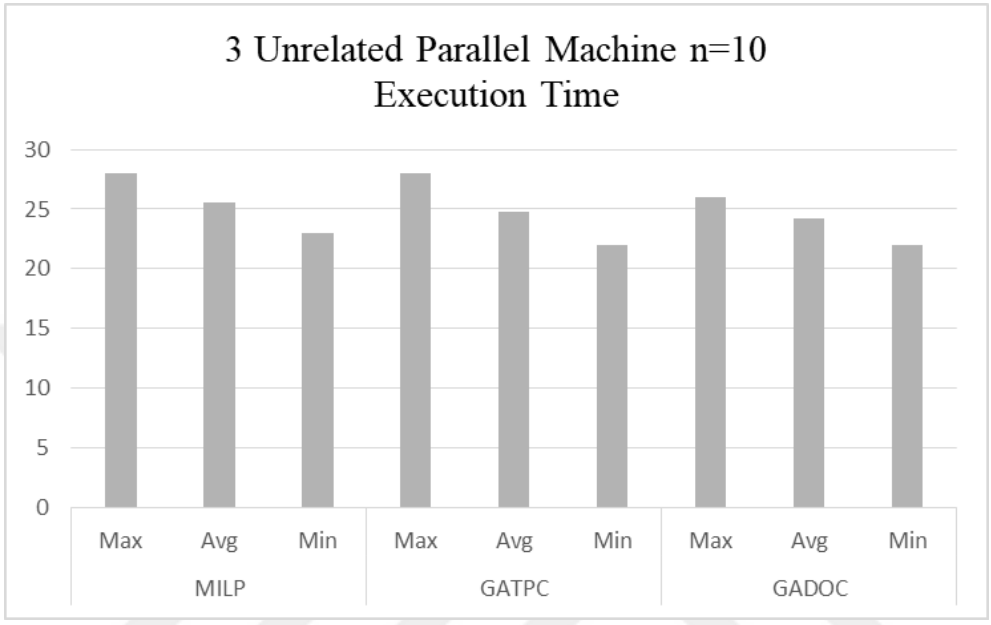
	A	B	C	D	E	F	G	H	I	J	K
1	C	K	T	X	ORDER	OBJ	Ei	UBD	TimeElapsed		
2	6	10	0	1	4	0	0	0	0		
3	25	12	0	1	5	0	0	0	0		
4	31	16	0	1	8	0	0	0	0		
5	45	19	0	1	6	0	0	0	0		
6	0	-2	0	0	1	0	0	0	0		
7	16	16	0	1	3	0	0	0	0		
8	41	19	0	1	2	0	0	0	0		
9	49	19	0	1	9	0	0	0	0		
10	0	-1	0	0	7	0	0	0	0		
11	54	5	0	1	10	113	124	0,088709677	23,2601929		
12											
13											
14											
15											
16											
17											
18											
19											

Appendix D- Execution Time Graphs of MILP, GATPC, and GADOC Algorithm

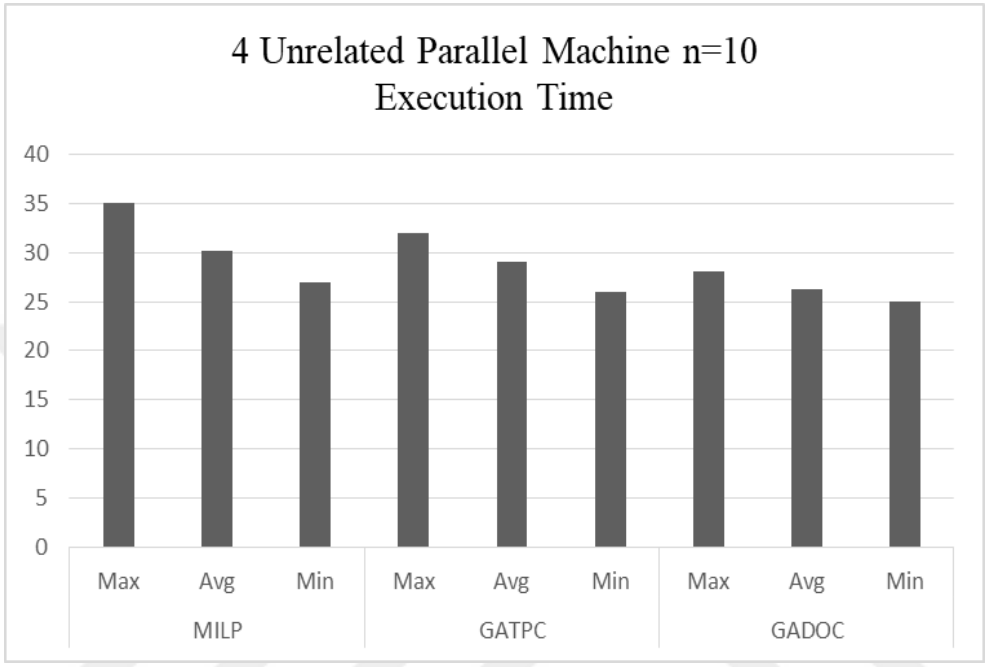
Appendix D-1 Execution Time Graph of MILP, GATPC, and GADOC n=10 m=2



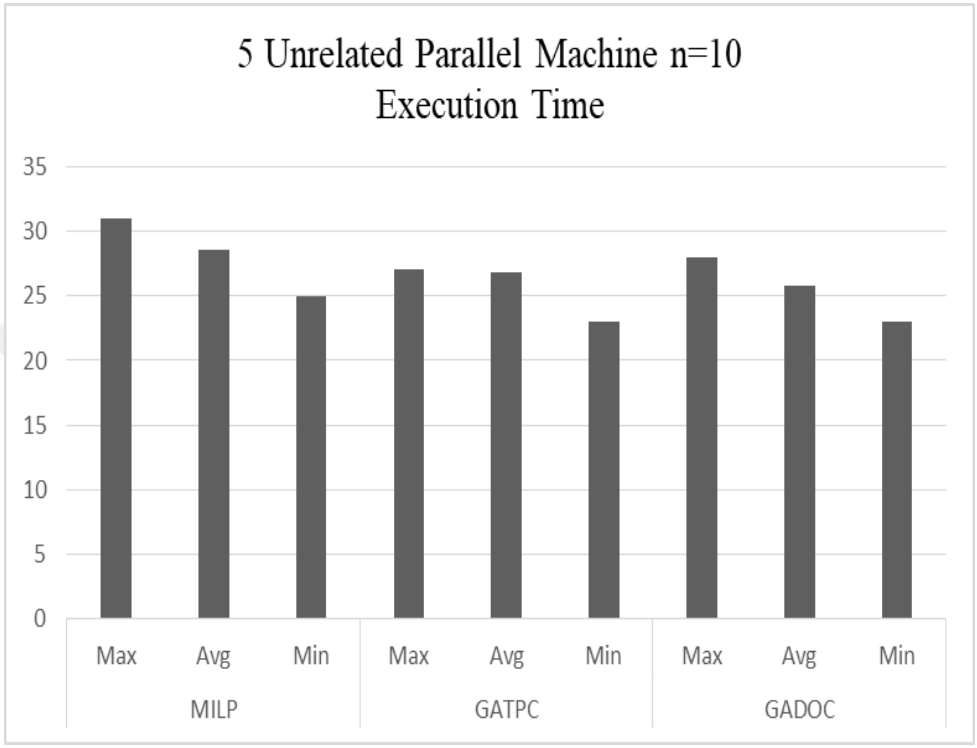
Appendix D-2 Execution Time Graph of MILP, GATPC and GADOC n=10 m=3



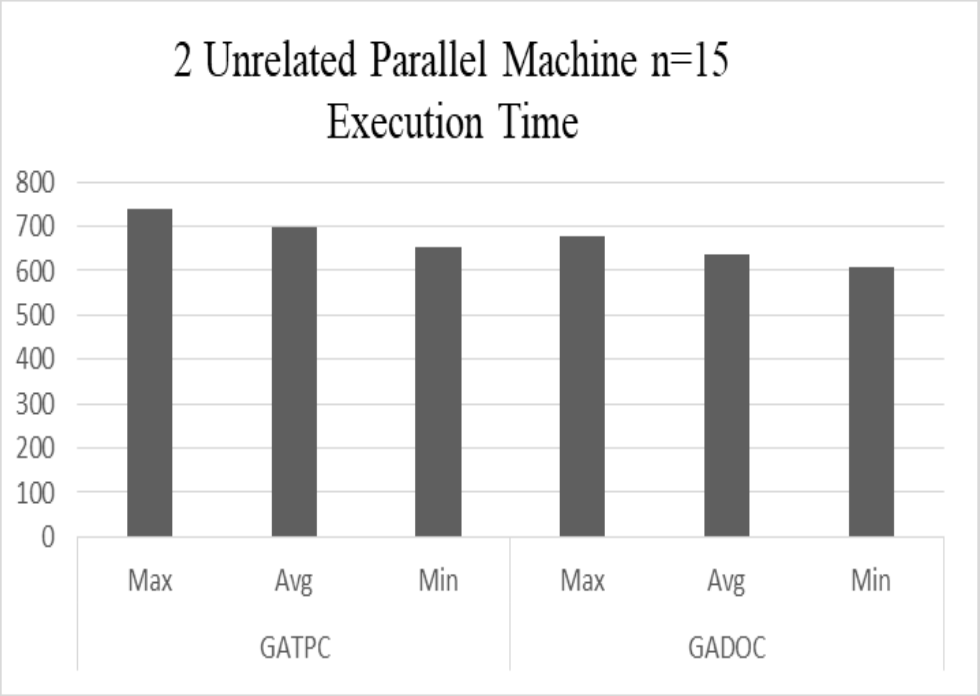
Appendix D-3 Execution Time Graph of MILP, GATPC and GADOC n=10 m=4



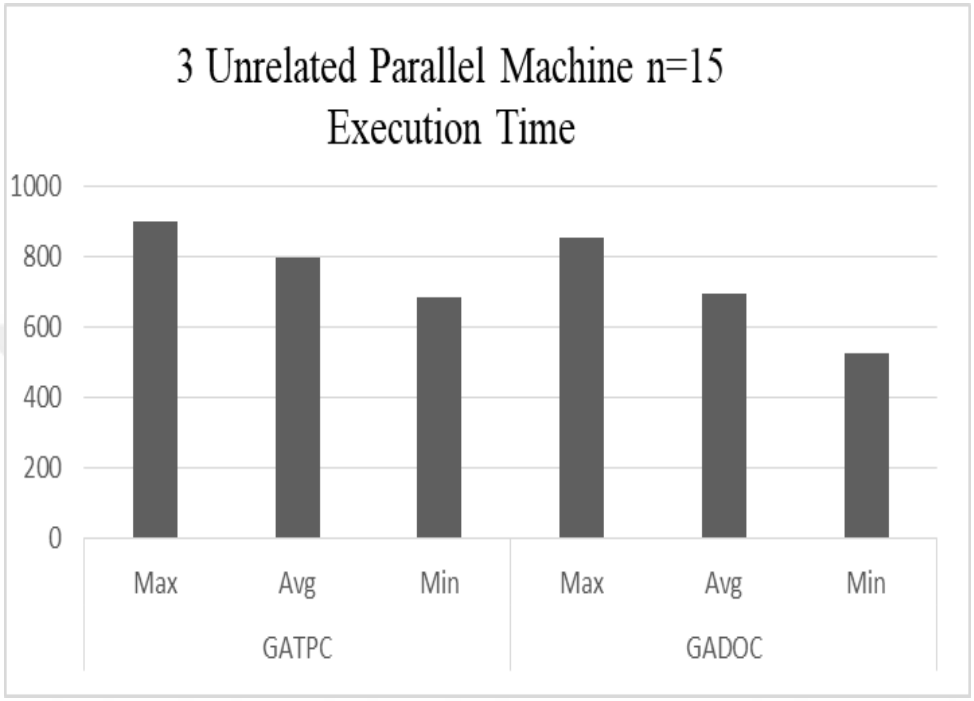
Appendix D-4 Execution Time Graph of MILP, GATPC and GADOC n=10 m=5



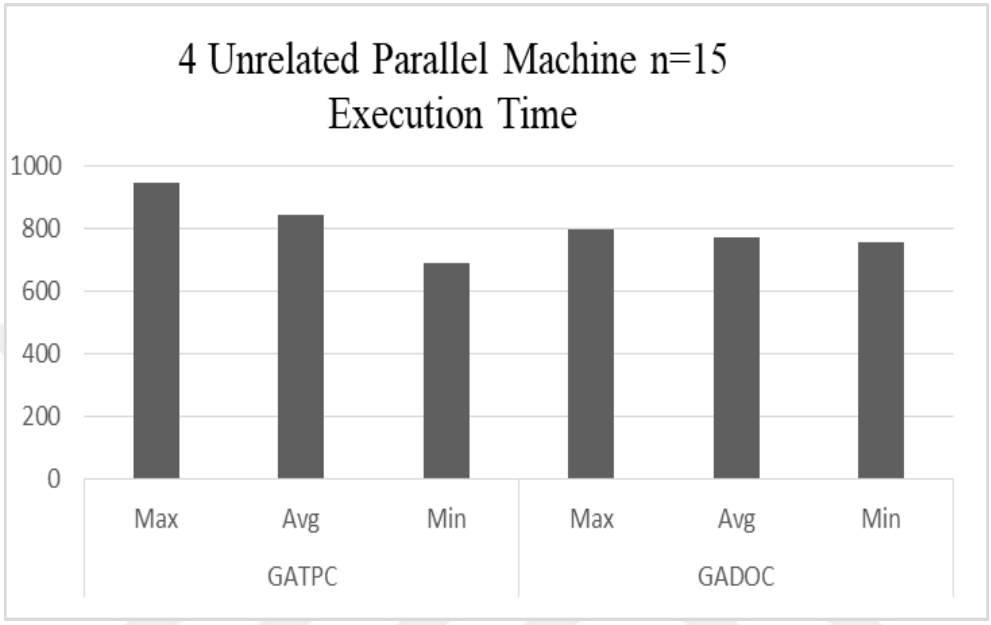
Appendix D-5 Execution Time Graph of GATPC and GADOC n=15 m=2



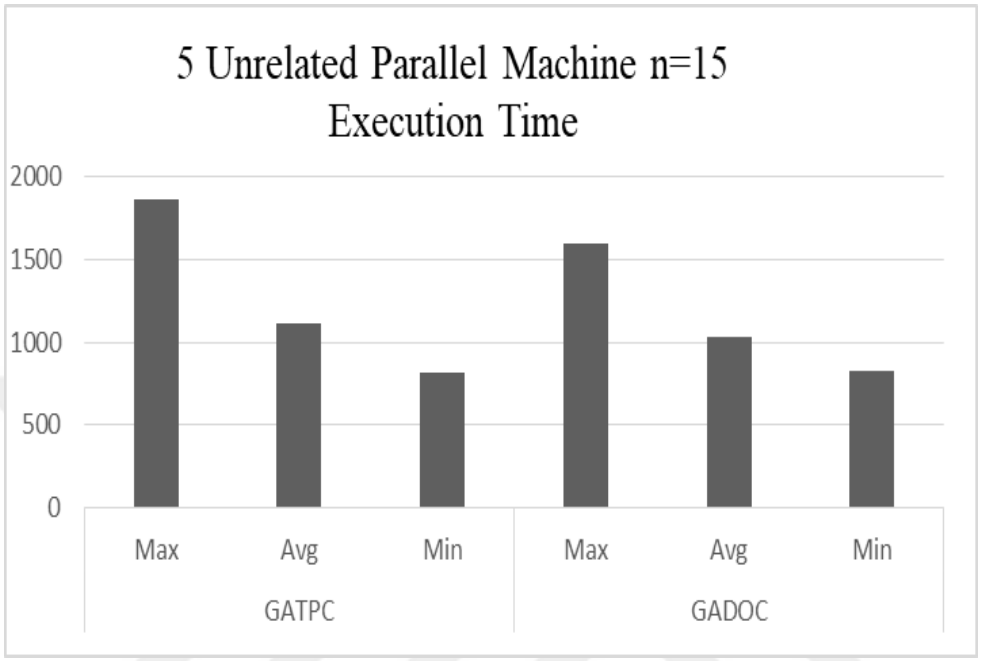
Appendix D-6 Execution Time Graph of GATPC and GADOC n=15 m=3



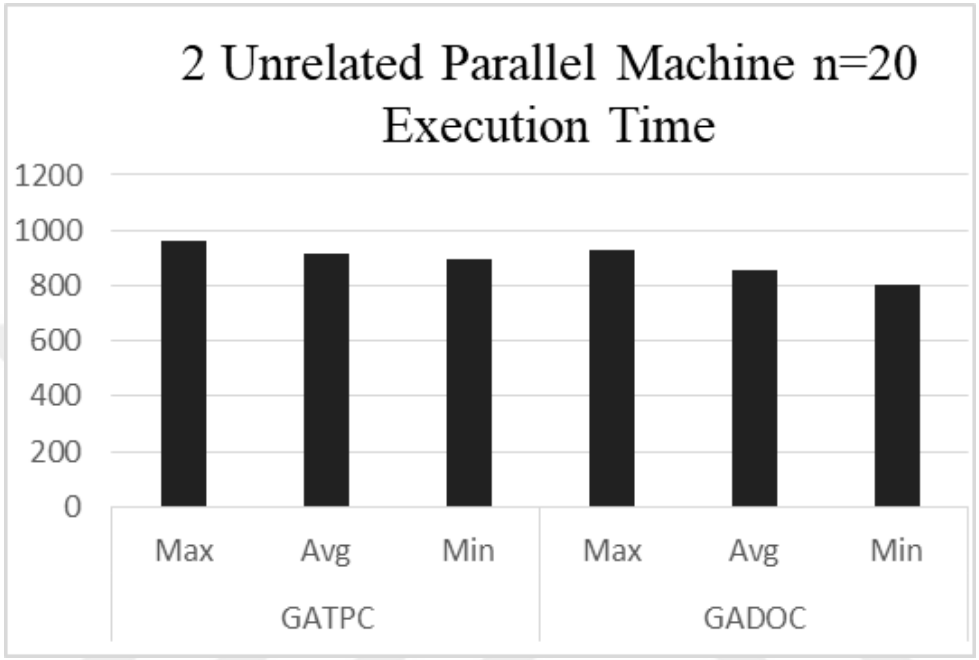
Appendix D-7 Execution Time Graph of GATPC and GADOC n=15 m=4



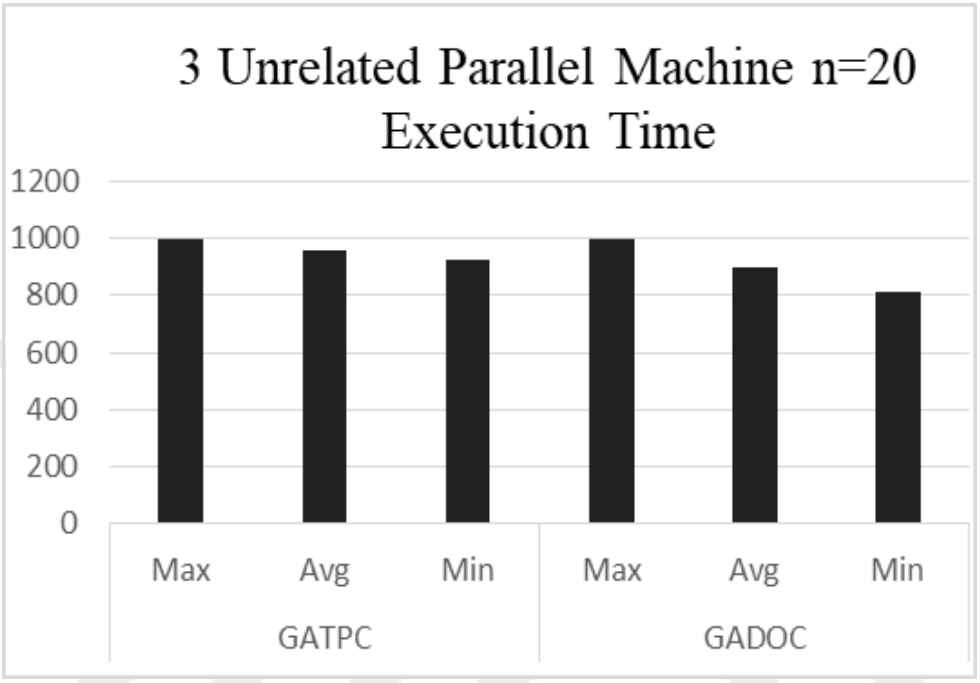
Appendix D-8 Execution Time Graph of GATPC and GADOC n=15 m=5



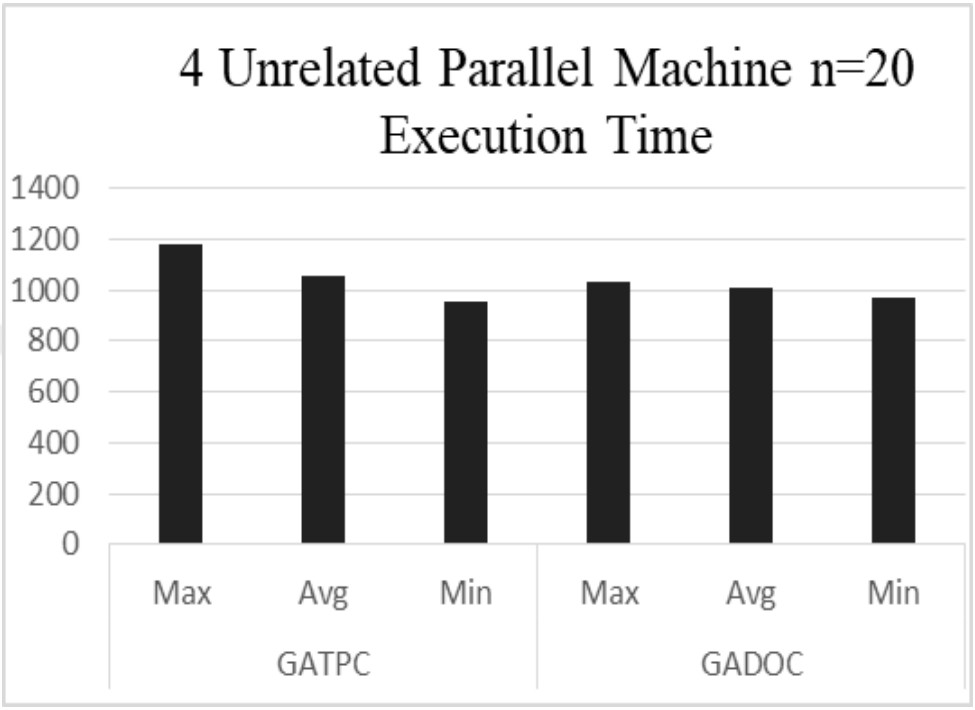
Appendix D-9 Execution Time Graph of GATPC and GADOC n=20 m=2



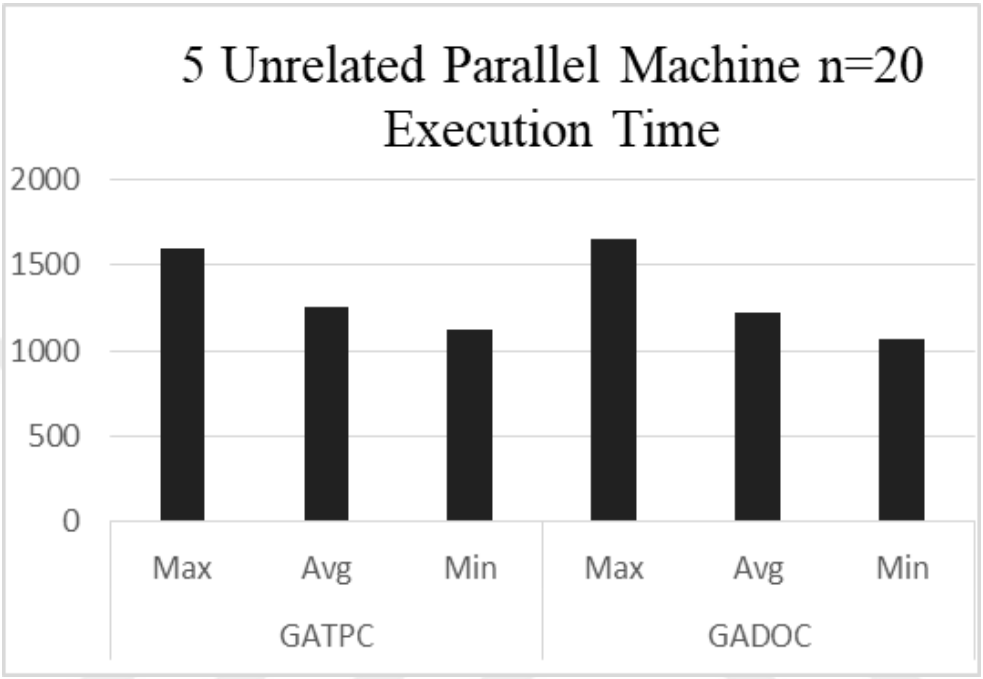
Appendix D-10 Execution Time Graph of GATPC and GADOC n=20 m=3



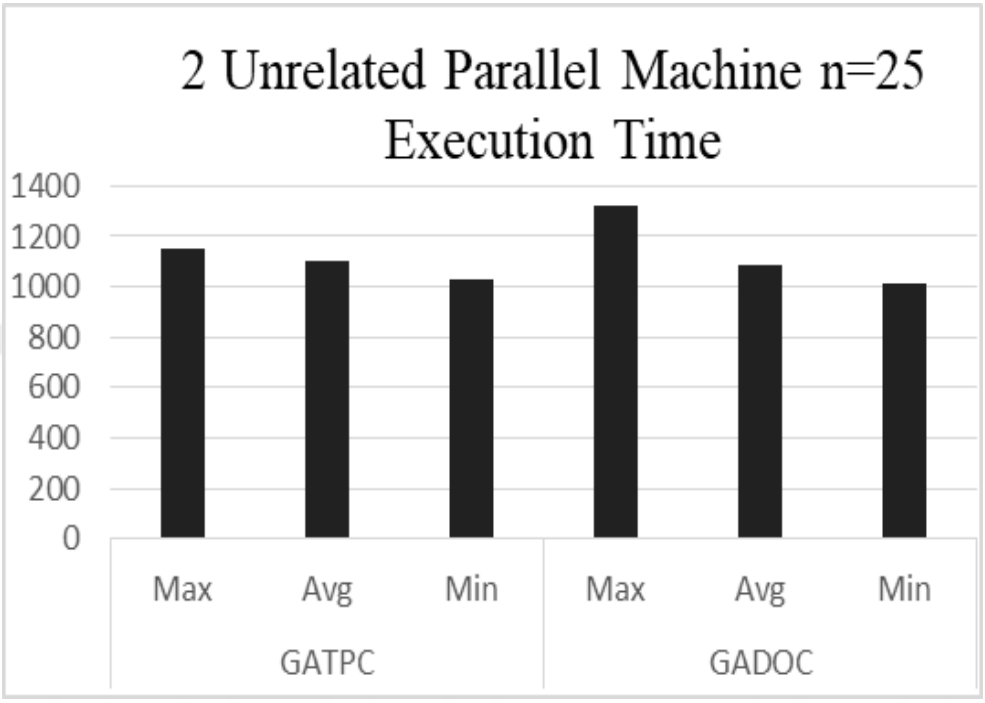
Appendix D-11 Execution Time Graph of GATPC and GADOC n=20 m=4



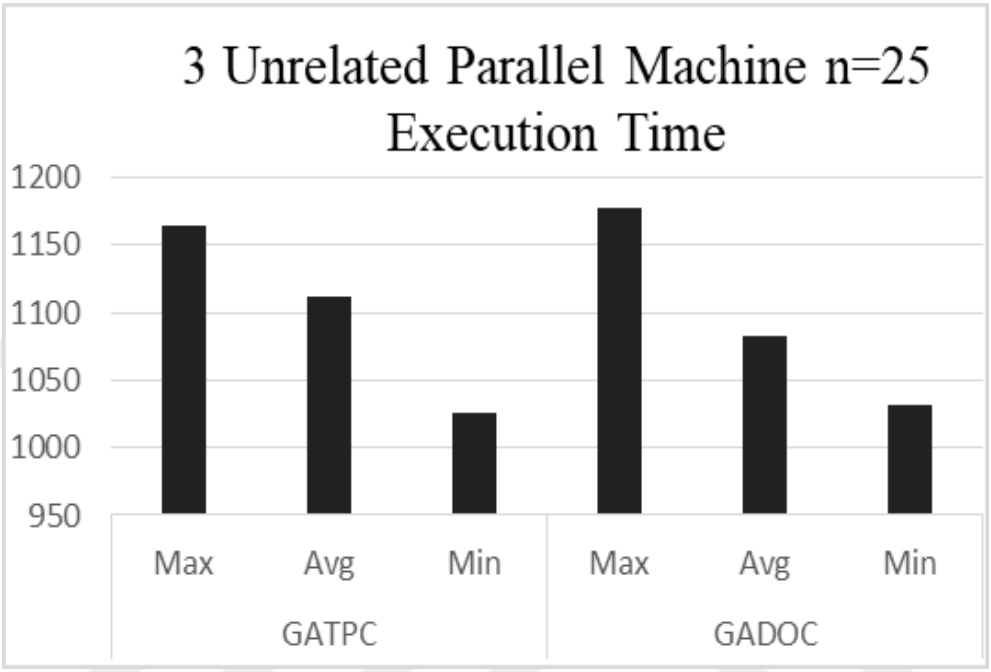
Appendix D-12 Execution Time Graph of GATPC and GADOC n=20 m=5



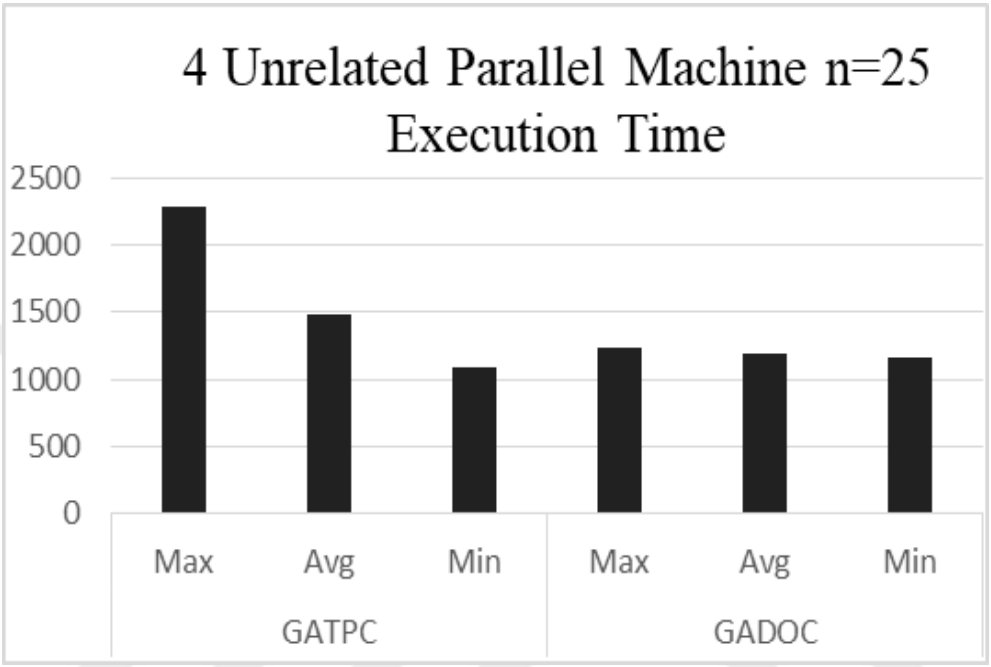
Appendix D-13 Execution Time Graph of GATPC and GADOC n=25 m=2



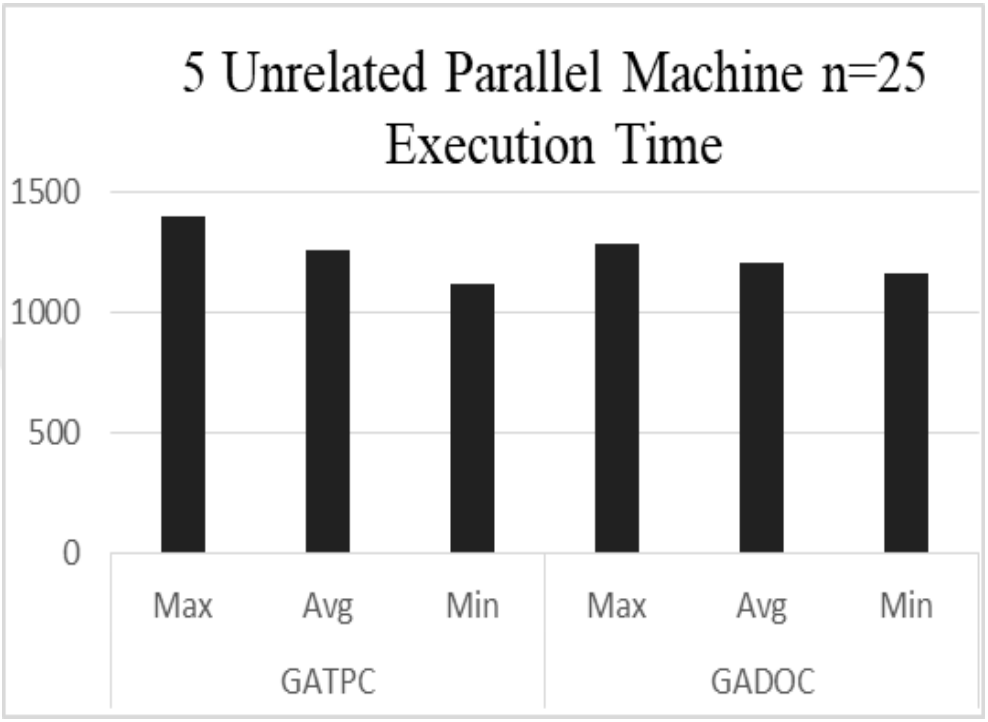
Appendix D-14 Execution Time Graph of GATPC and GADOC n=25 m=3



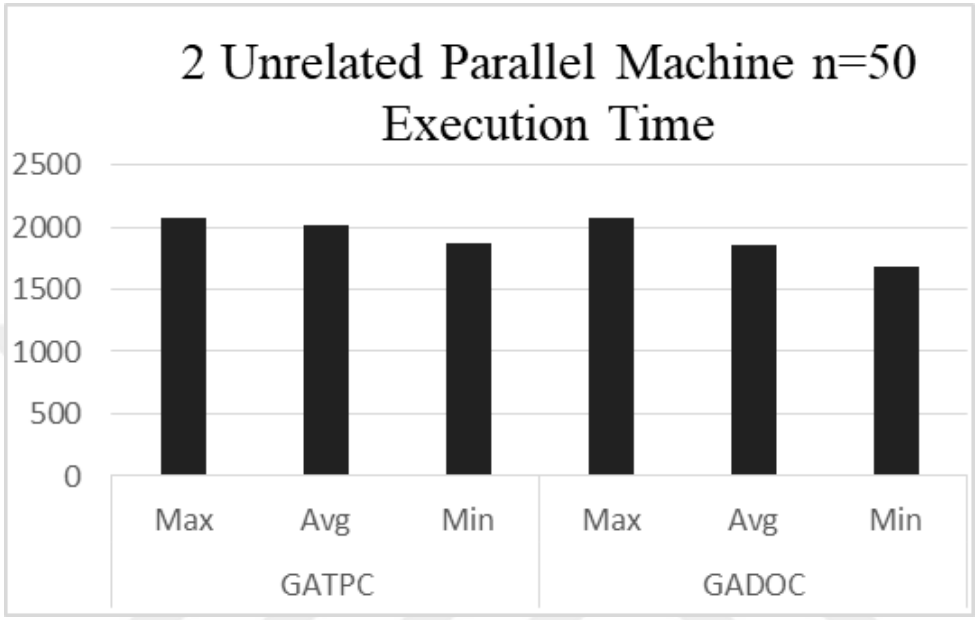
Appendix D-15 Execution Time Graph of GATPC and GADOC n=25 m=4



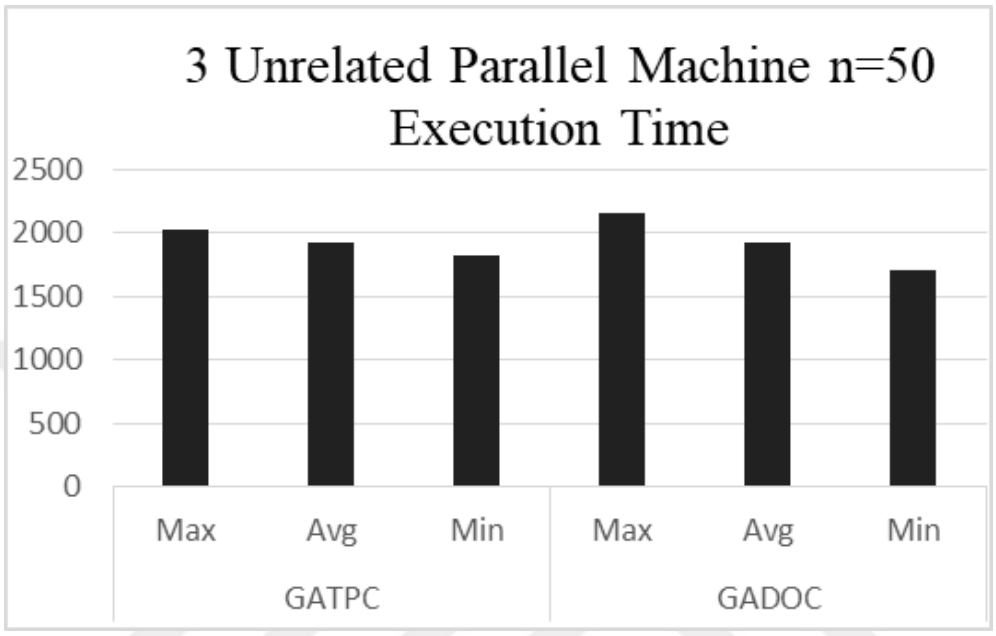
Appendix D-16 Execution Time Graph of GATPC and GADOC n=25 m=5



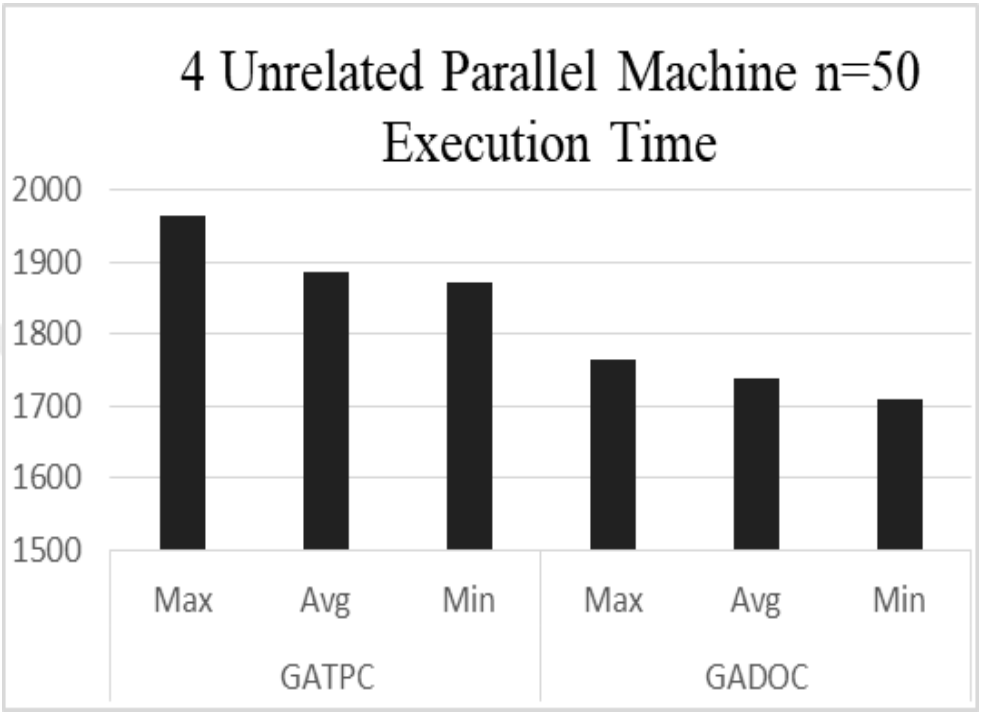
Appendix D-17 Execution Time Graph of GATPC and GADOC n=50 m=2



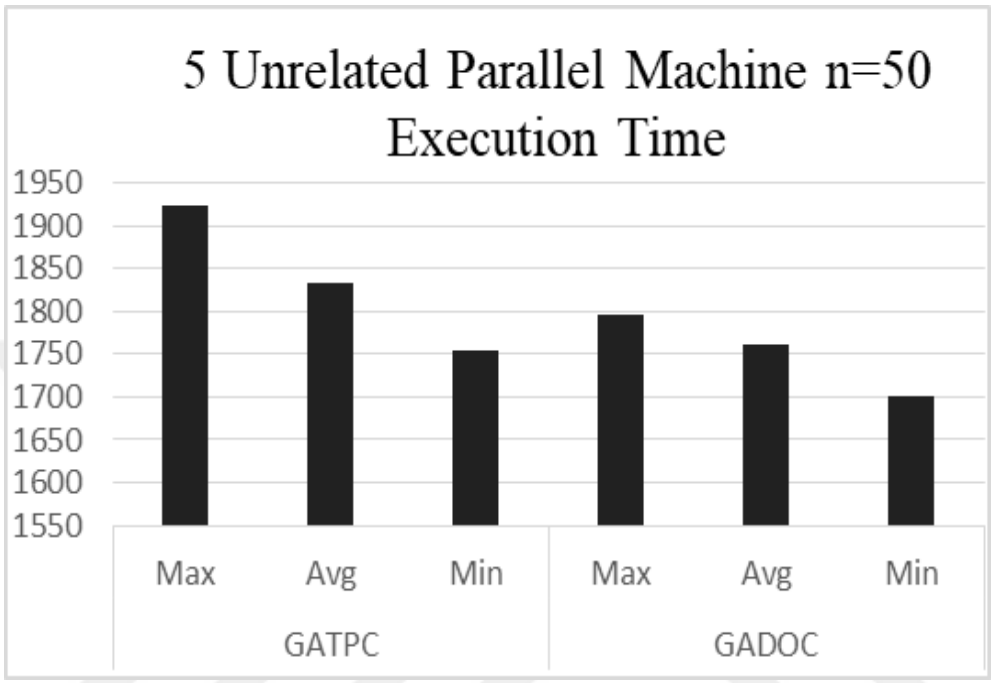
Appendix D-18 Execution Time Graph of GATPC and GADOC n=50 m=3



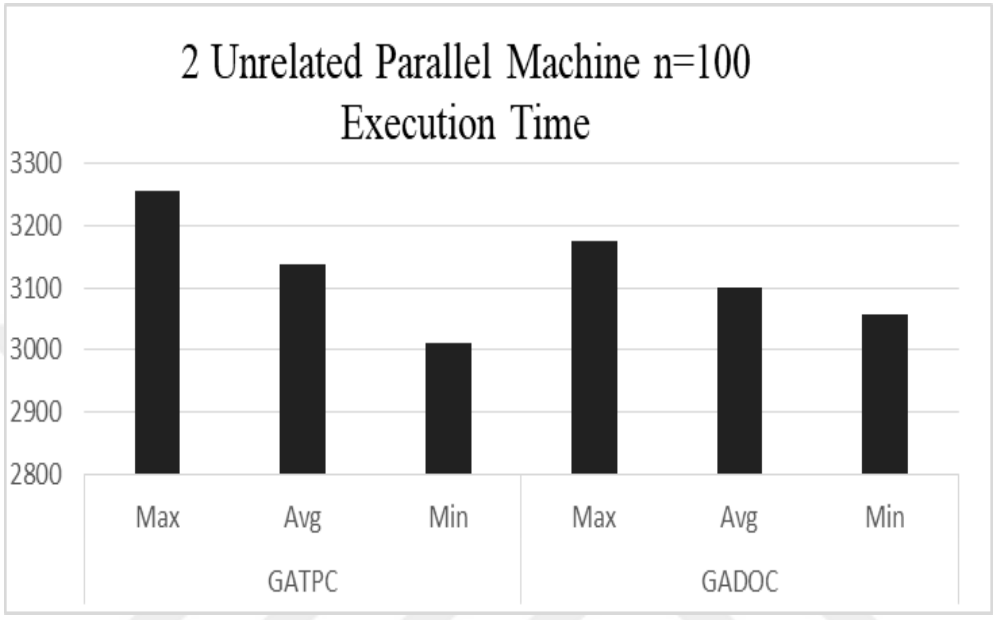
Appendix D-19 Execution Time Graph of GATPC and GADOC n=50 m=4



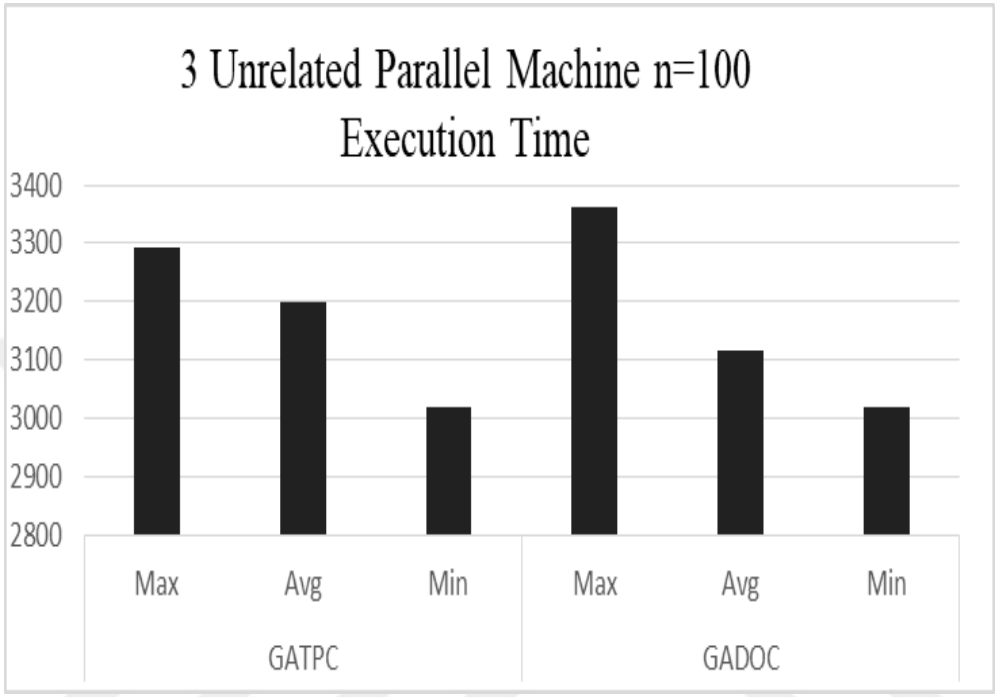
Appendix D-20 Execution Time Graph of GATPC and GADOC n=50 m=5



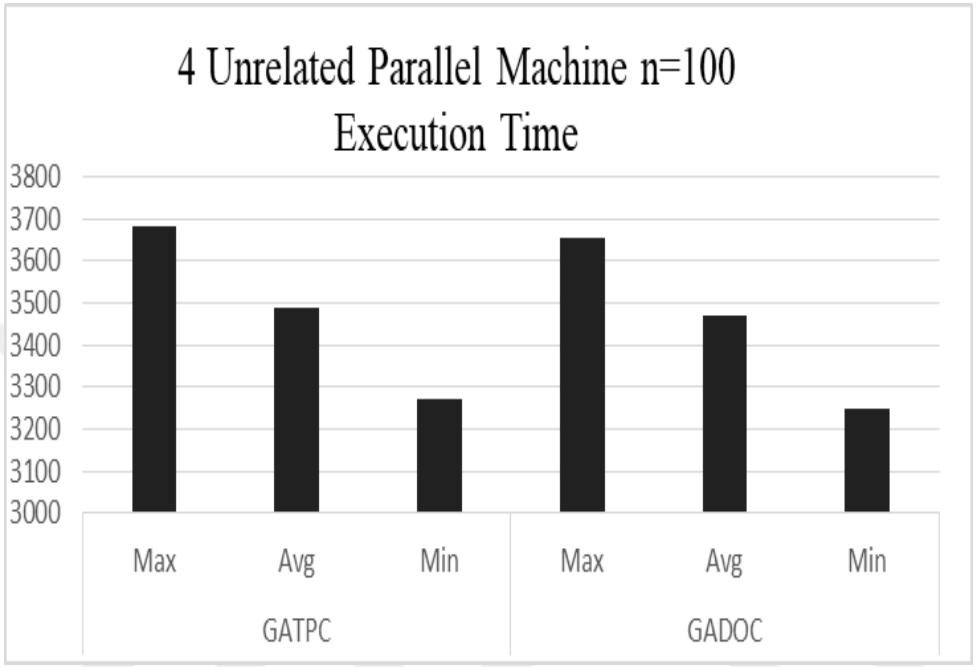
Appendix D-21 Execution Time Graph of GATPC and GADOC n=100 m=2



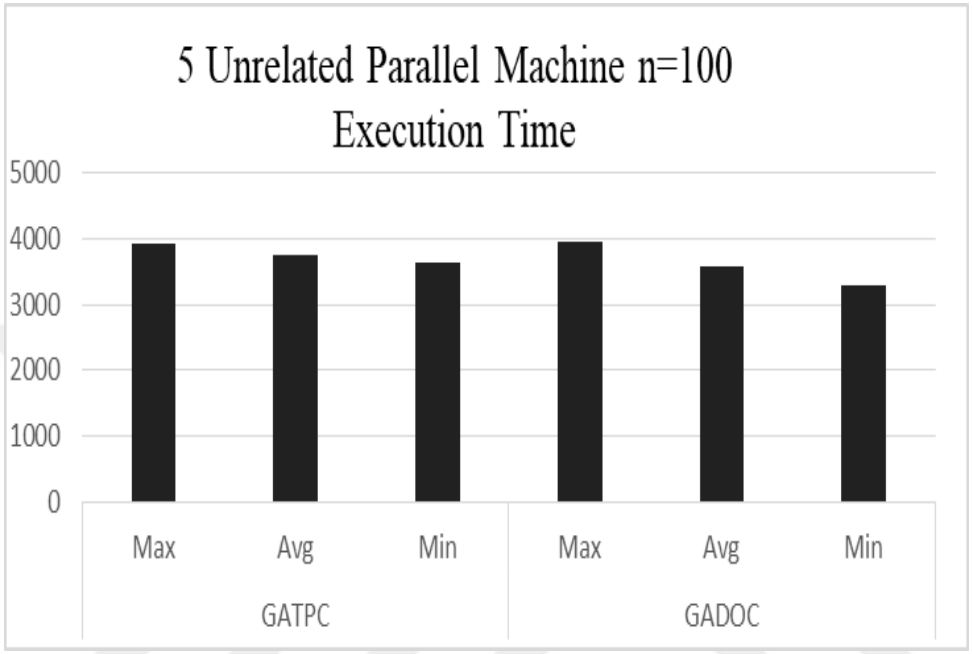
Appendix D-22 Execution Time Graph of GATPC and GADOC n=100 m=3



Appendix D-23 Execution Time Graph of GATPC and GADOC n=100 m=4

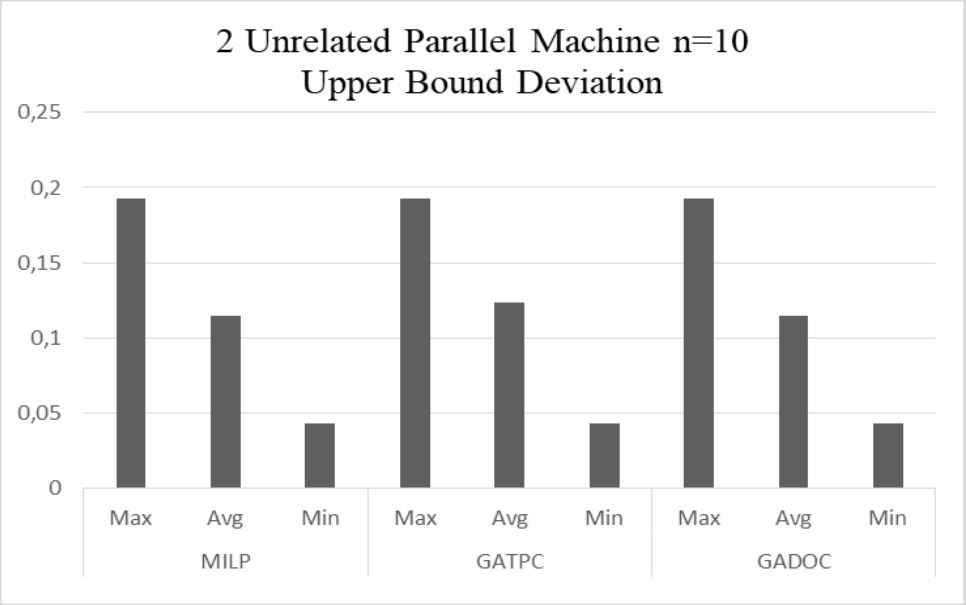


Appendix D-24 Execution Time Graph of GATPC and GADOC n=100 m=5

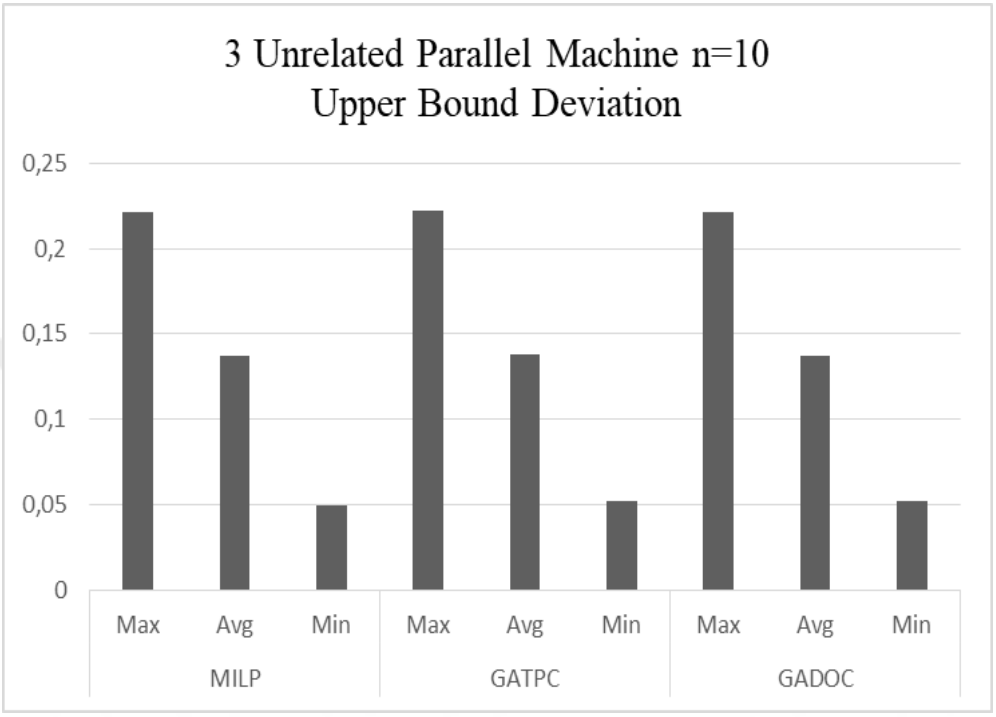


Appendix E- Upper Bound Deviation Graphs of MILP, GATPC, and GADOC Algorithm

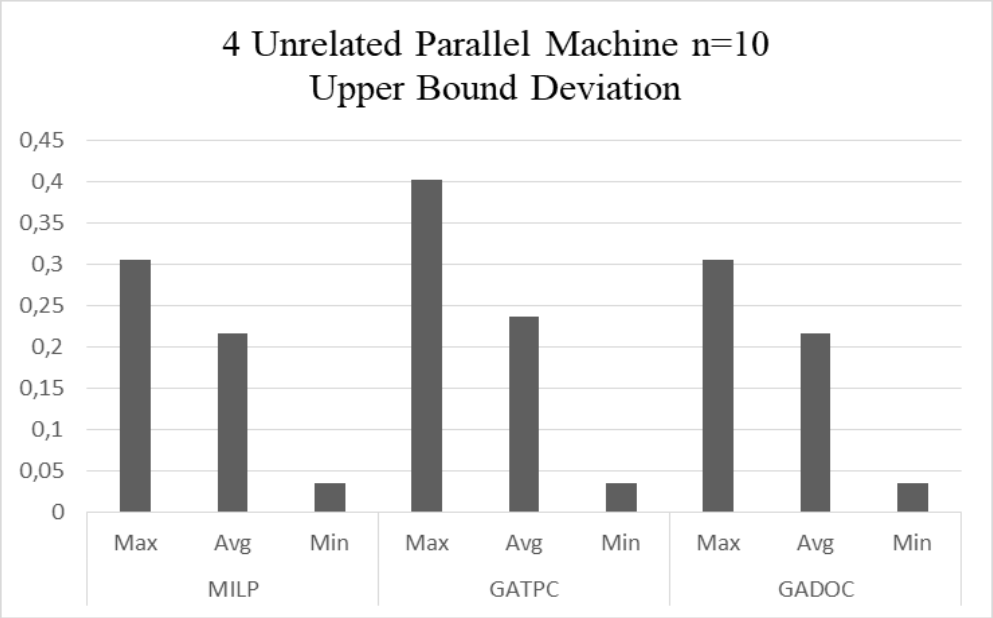
Appendix E-1 UBD Graph of MILP, GATPC, and GADOC n=10 m=2



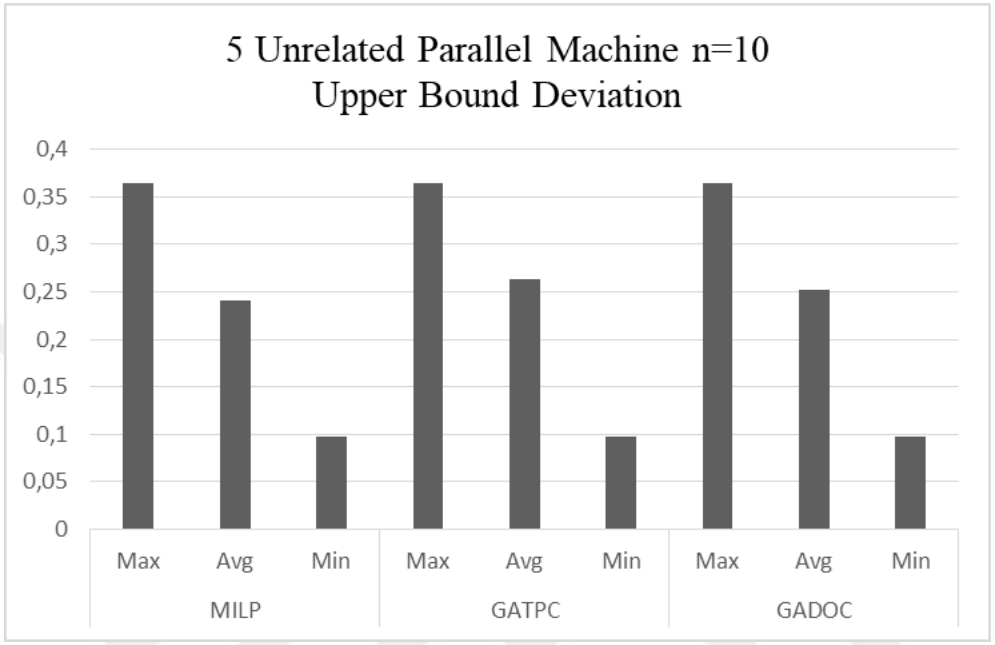
Appendix E-2 UBD Graph of MILP, GATPC and GADOC n=10 m=3



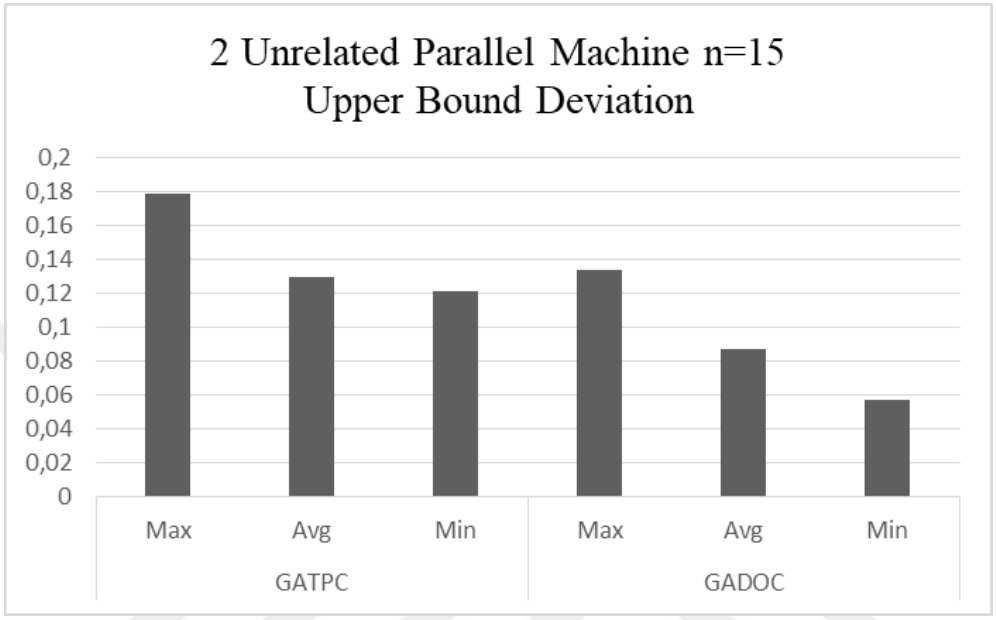
Appendix E-3 UBD Graph of MILP, GATPC and GADOC n=10 m=4



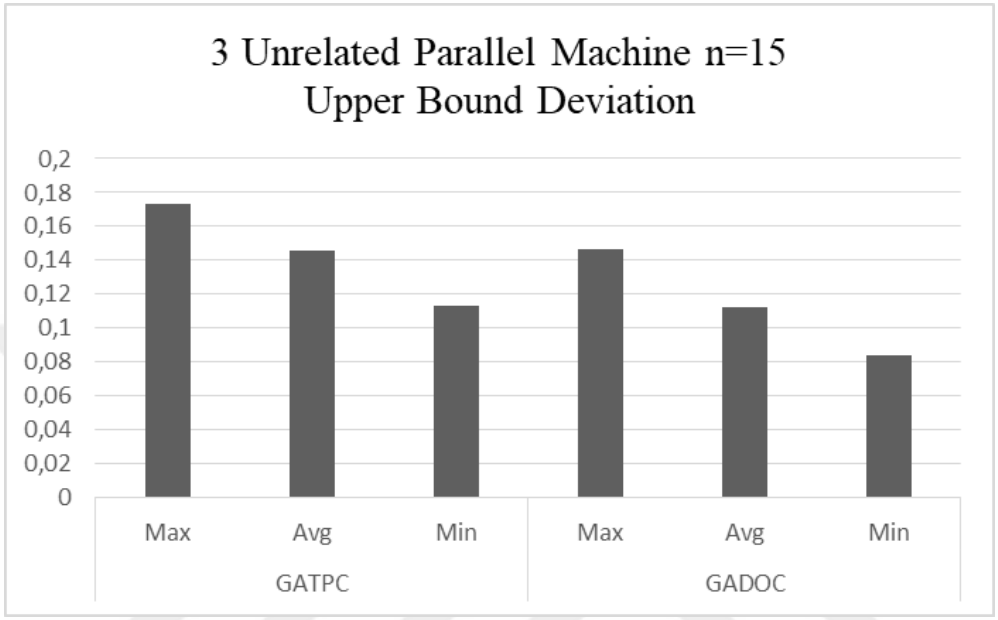
Appendix E-4 UBD Graph of MILP, GATPC and GADOC n=10 m=5



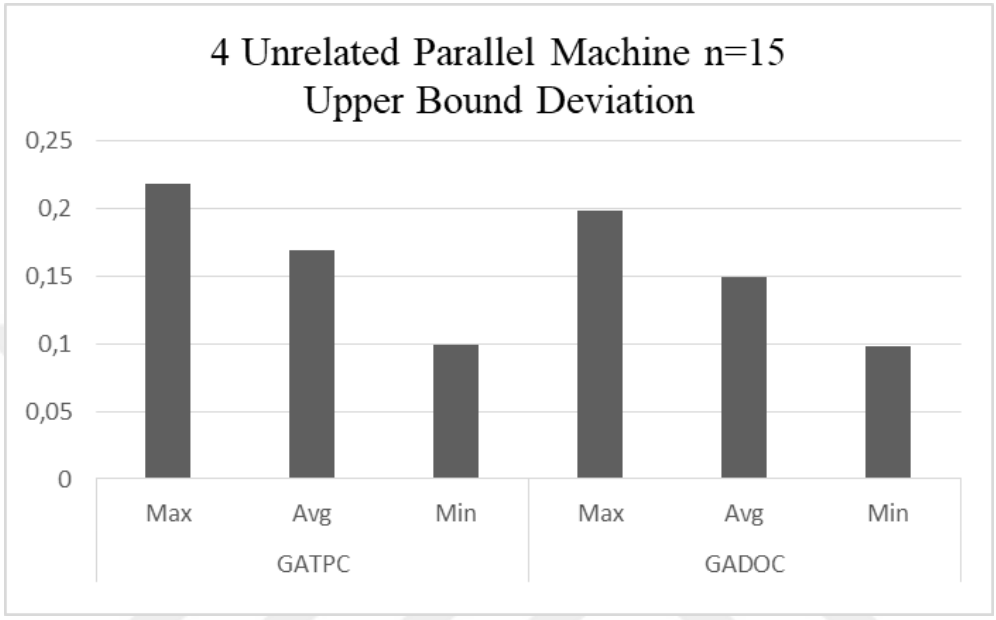
Appendix E-5 UBD Graph of GATPC and GADOC n=15 m=2



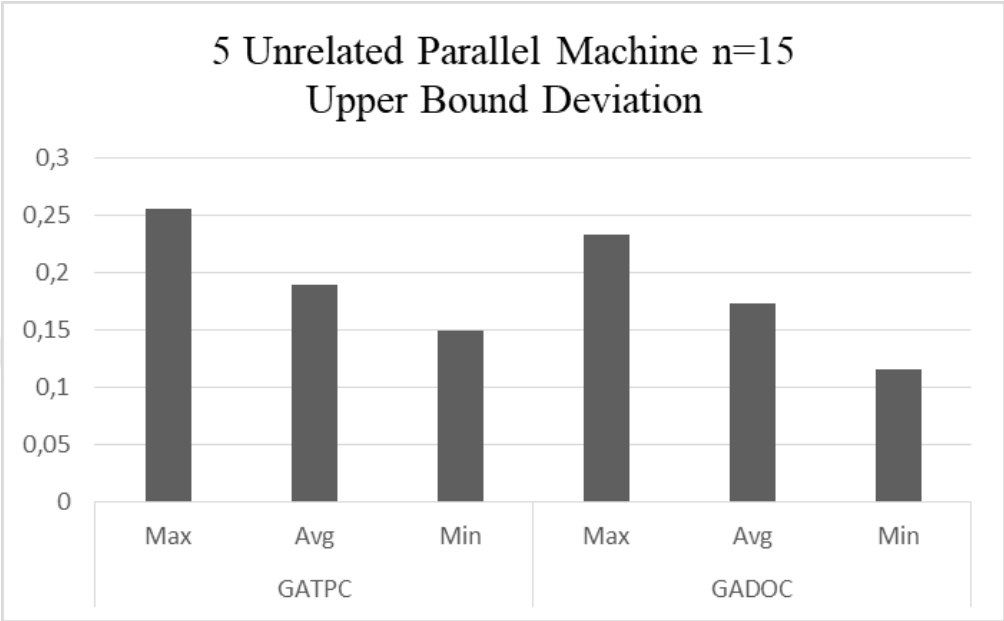
Appendix E-6 UBD Graph of GATPC and GADOC n=15 m=3



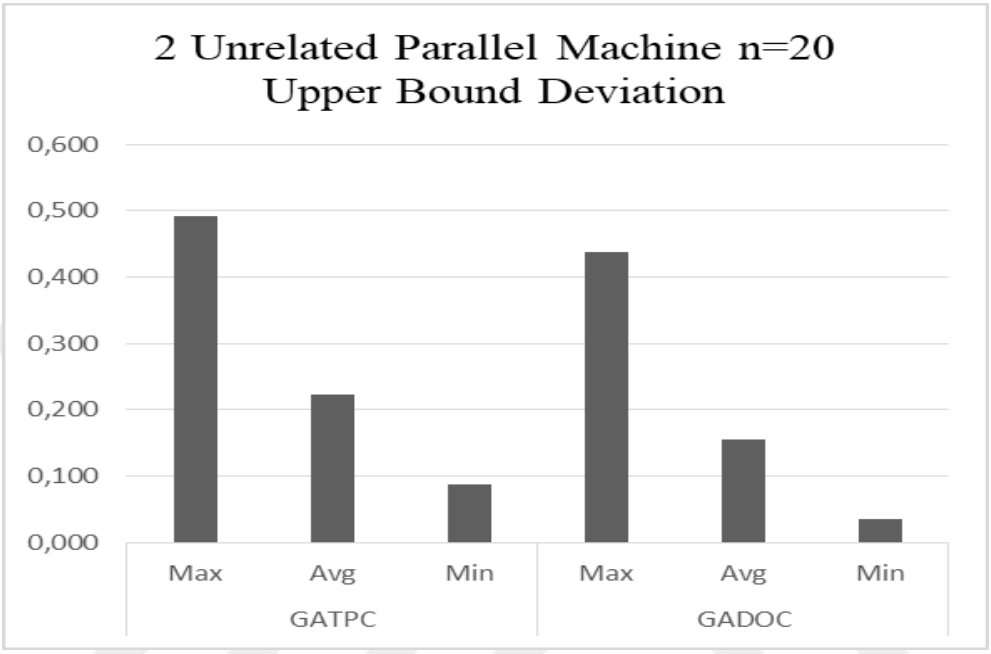
Appendix E-7 UBD Graph of GATPC and GADOC n=15 m=4



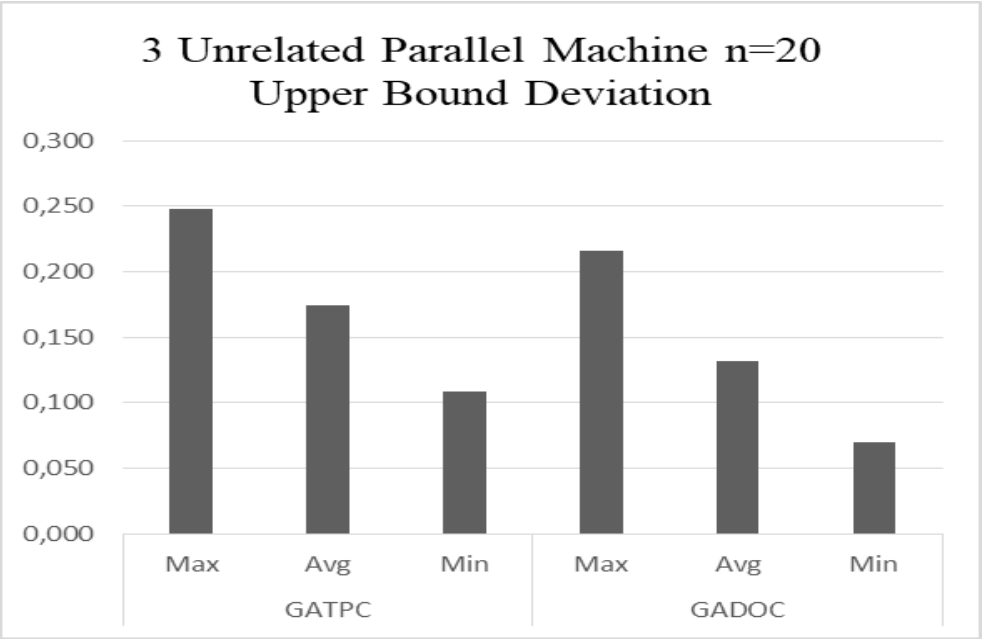
Appendix E-8 UBD Graph of GATPC and GADOC n=15 m=5



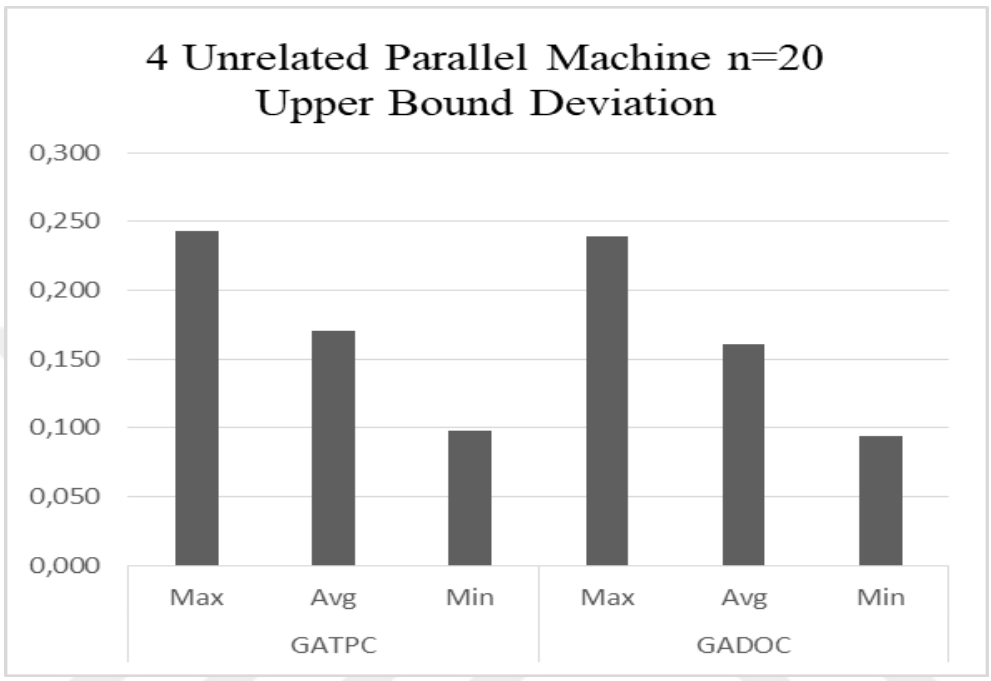
Appendix E-9 UBD Graph of GATPC and GADOC n=20 m=2



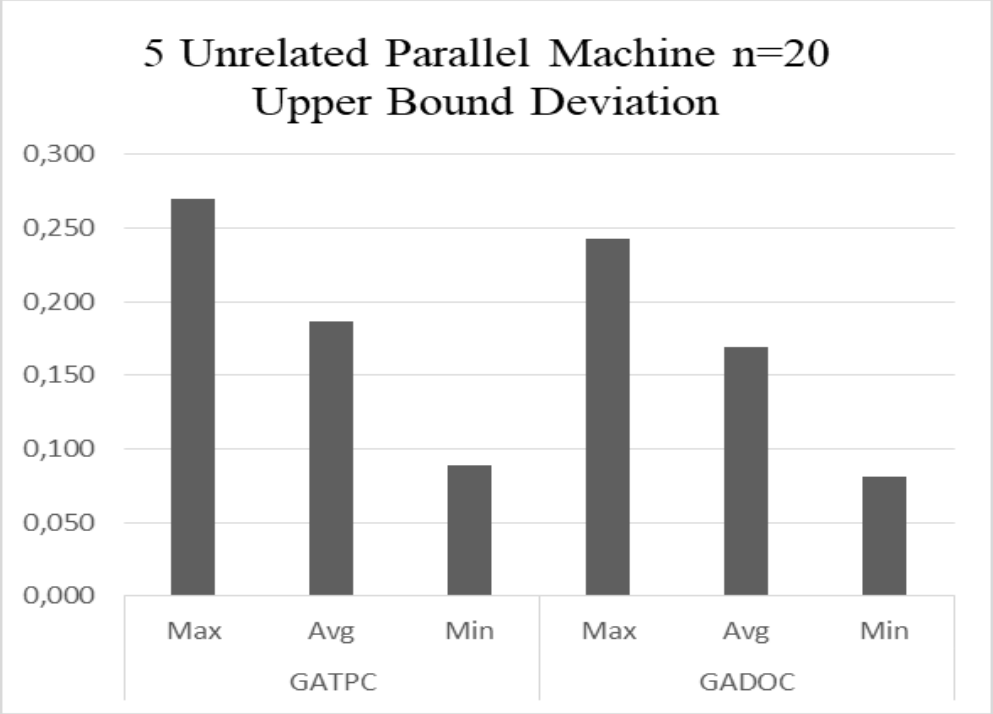
Appendix E-10 UBD Graph of GATPC and GADOC n=20 m=3



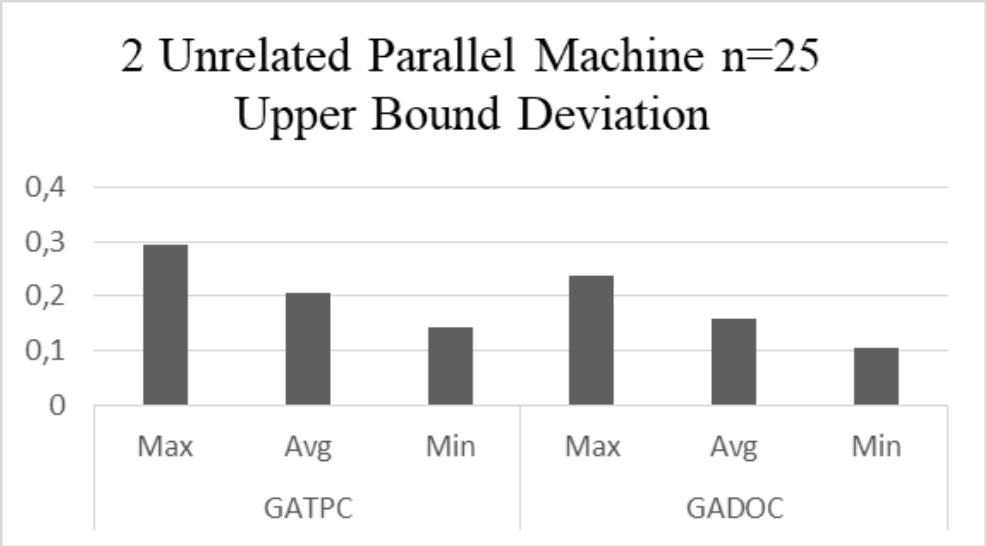
Appendix E-11 UBD Graph of GATPC and GADOC n=20 m=4



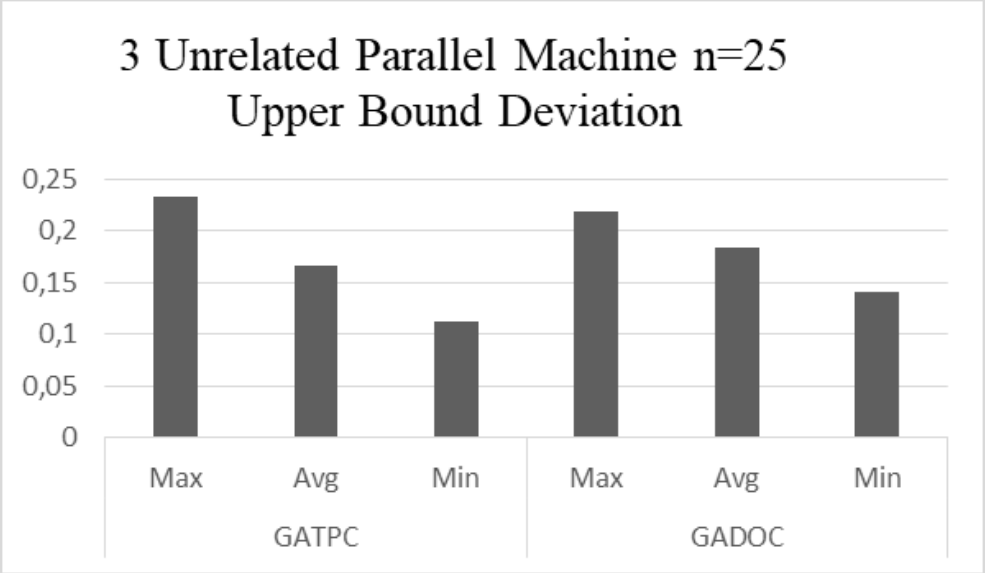
Appendix E-12 UBD Graph of GATPC and GADOC n=20 m=5



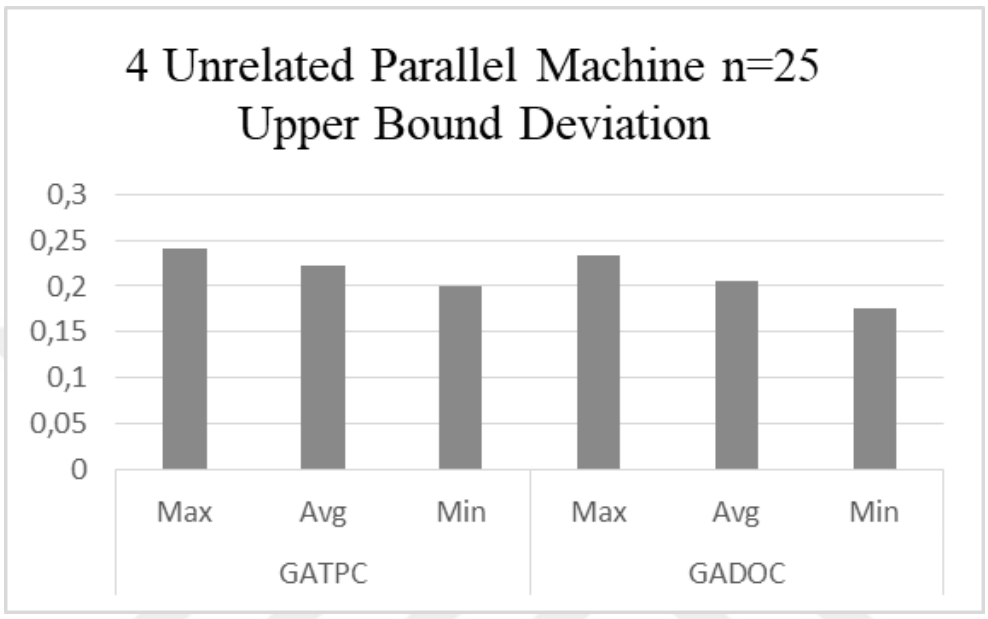
Appendix E-13 UBD Graph of GATPC and GADOC n=25 m=2



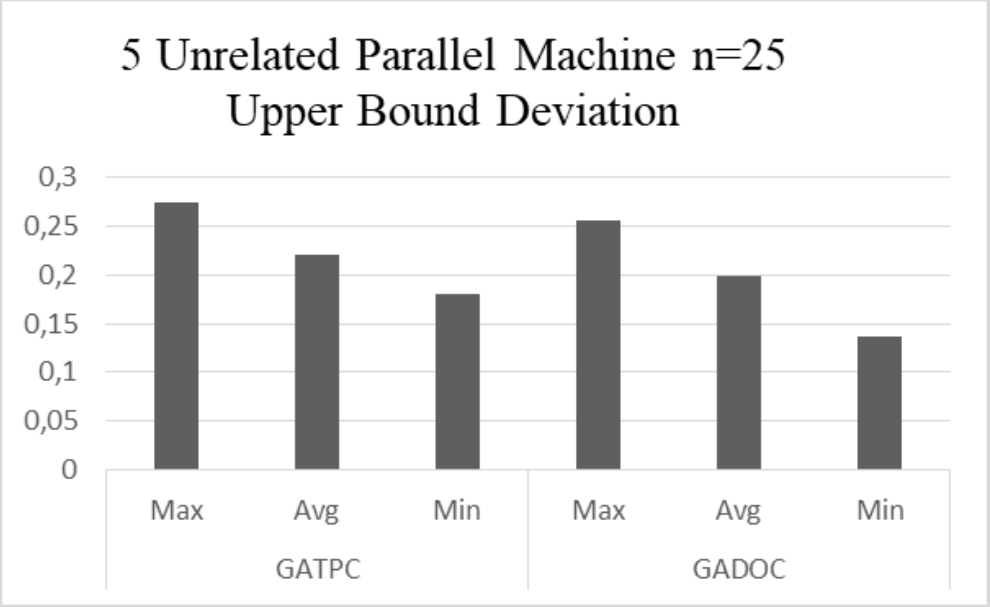
Appendix E-14 UBD Graph of GATPC and GADOC n=25 m=3



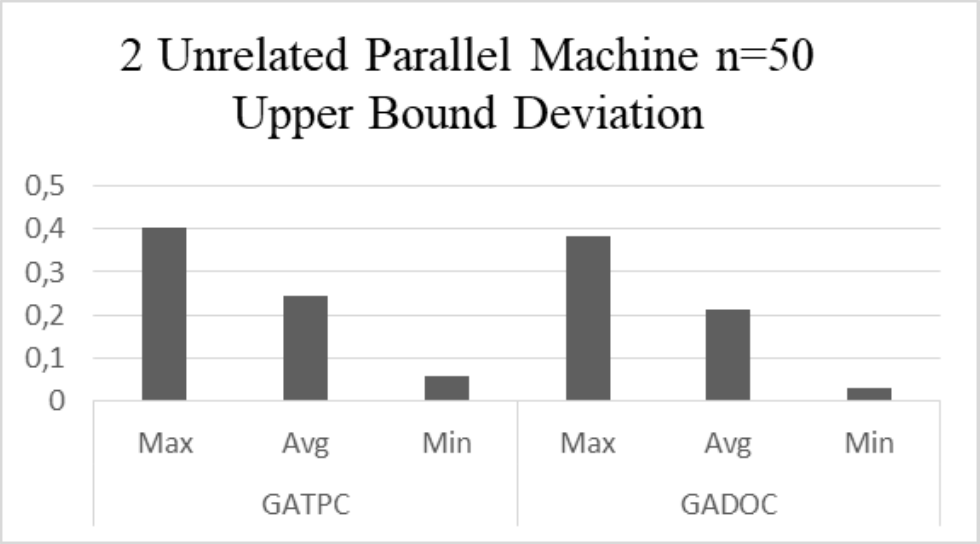
Appendix E-15 UBD Graph of GATPC and GADOC n=25 m=4



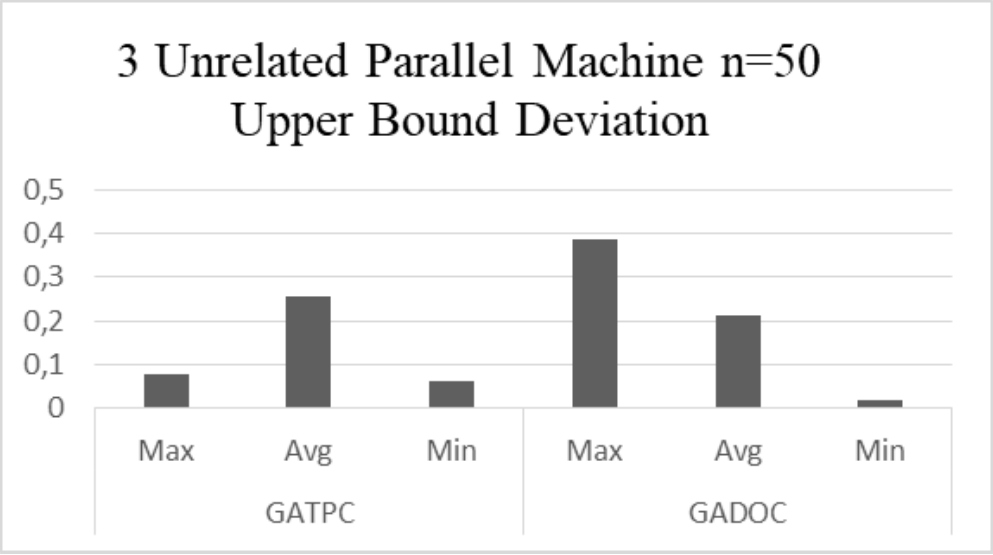
Appendix E-16 UBD Graph of GATPC and GADOC n=25 m=5



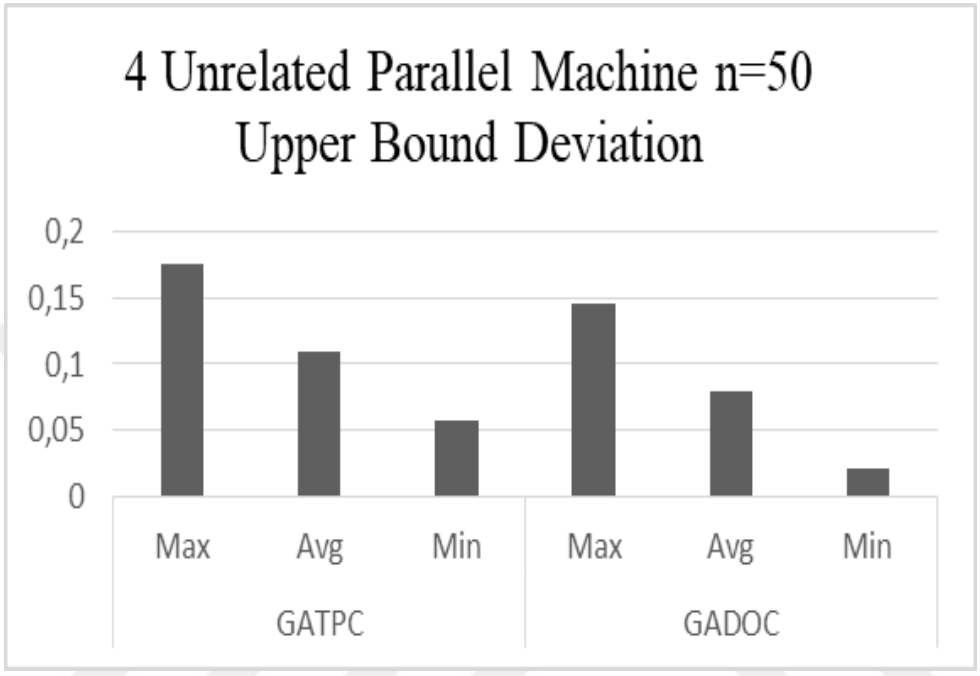
Appendix E-17 UBD Graph of GATPC and GADOC n=50 m=2



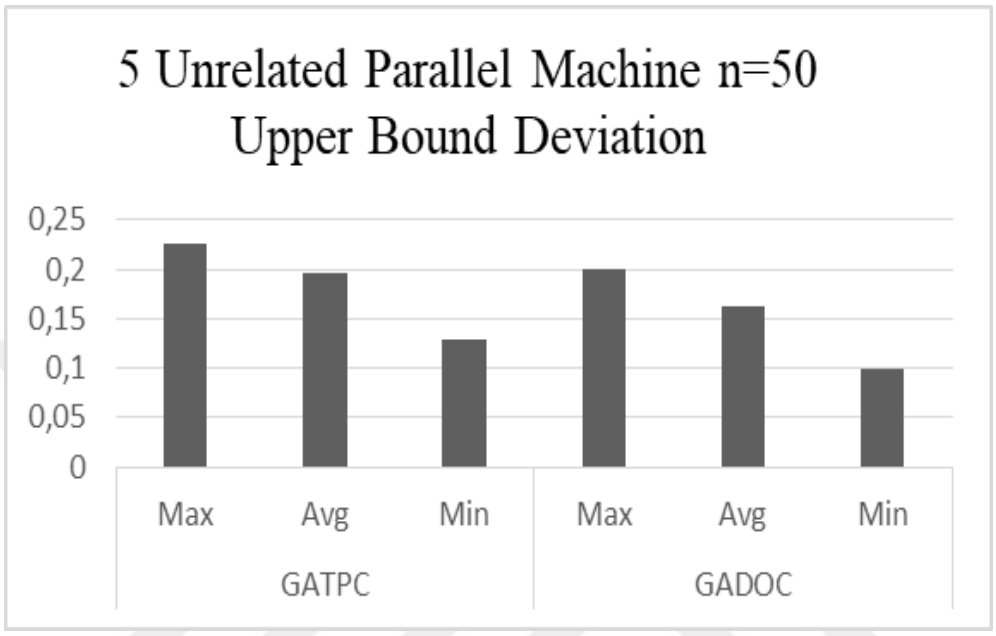
Appendix E-18 UBD Graph of GATPC and GADOC n=50 m=3



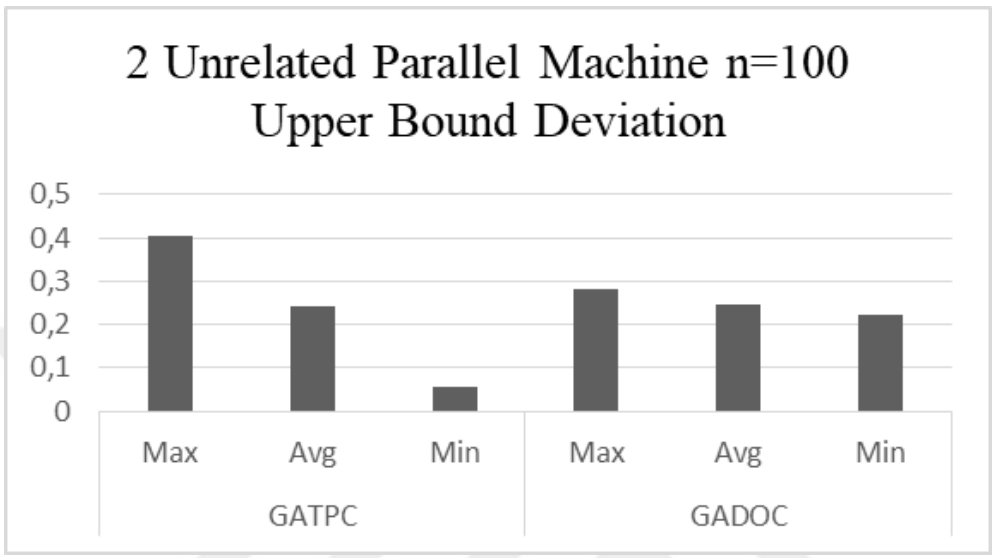
Appendix E-19 UBD Graph of GATPC and GADOC n=50 m=4



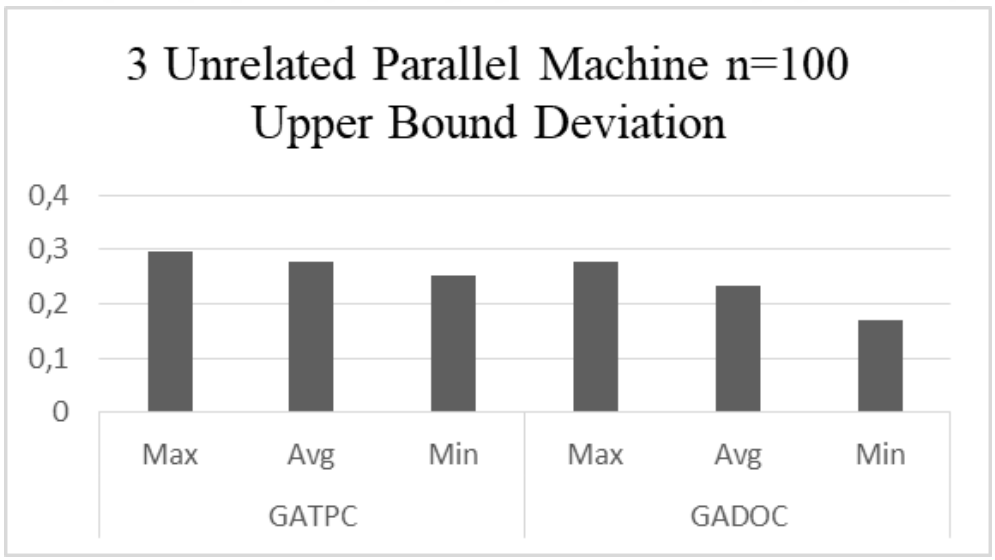
Appendix E-20 UBD Graph of GATPC and GADOC n=50 m=5



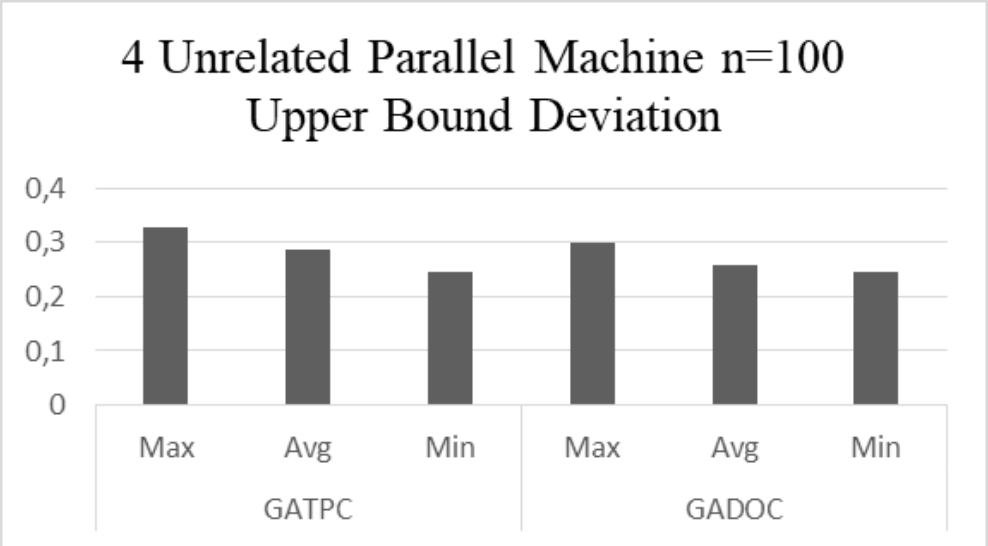
Appendix E-21 UBD Graph of GATPC and GADOC n=100 m=2



Appendix E-22 UBD Graph of GATPC and GADOC n=100 m=3



Appendix E-23 UBD Graph of GATPC and GADOC n=100 m=4



Appendix E-24 UBD Graph of GATPC and GADOC n=100 m=5