ASSIGNMENT AND SCHEDULING PROBLEM IN IDENTICAL PARALLEL MACHINES

Damla KIZILAY

Supervisor: Mehmet Fatih TAŞGETİREN

Co-Advisor: Önder BULUT

Bornova, IZMIR

2014

YASAR UNIVERSITY
YAŞAR UNIVERSITY NATURAL AND APPLIED SCIENCES

ASSIGNMENT AND SCHEDULING PROBLEM IN IDENTICAL PARALLEL MACHINES

Damla KIZILAY

Supervisor: Assoc. Prof. Dr. M. Fatih TAŞGETİREN
Co-Advisor: Assist. Prof. Dr. Önder BULUT

Industrial Engineering

Bornova, IZMIR
2014

This study titled “Assignment and Scheduling Problem in Identical Parallel Machines ” and presented as Master’s Thesis by Damla KIZILAY has been evaluated in compliance with the relevant provisions of Y.U Graduate Education and Training Regulation and Y.U Institute of Science Education and Training Direction and jury members written below have decided for the defense of this thesis and it has been declared by consensus / majority of votes that the candidate has succeeded in thesis defense examination dated 27.01.2014.

Jury Members: **Signature:**

Head: Assoc. Prof. Fatih TAŞGETİREN

Rapporteur Member: Assist. Prof. Ahmet CAMCI

Member: Assist. Prof. Yücel ÖZTÜRKOĞLU

TEXT OF OATH

I declare and honestly confirm that my study titled "Assignment and Scheduling Problem in Identical Parallel Machines", and presented as Master's Thesis has been written without applying to any assistance inconsistent with scientific ethics and traditions and all sources I have benefited from are listed in bibliography and I have benefited from these sources by means of making references.

17 / 01 / 2014

Damla KIZILAY

ÖZET

ÖZDEŞ PARALEL MAKİNELERDE ATAMA VE ÇİZELGELEME PROBLEMİ

KIZILAY, Damla

Yüksek Lisans Tezi, Endüstri Mühendisliği Bölümü
Tez Danışmanı: Doç. Dr. Mehmet Fatih TAŞGETİREN
İkinci Tez Danışmanı: Yard. Doç. Dr. Önder BULUT
Ocak, 2014

Bu çalışmada ele alınan problem, üretim süreçleri tamamlanmış çeşitli boya tiplerinin, istenilen ambalajlarla eşlenerek, dolum makinalarında çizelgelenmesidir. DYO dolum ünitesinde işlem süreleri birbirinden farklı üç makina grubu bulunmaktadır; otomatik, yarı otomatik ve manuel. Her bir makina grubu ise farklı sayılarda özdeş makinalardan oluşmaktadır. Bu nedenle, problem iki aşamalı olarak ele alınmıştır; işlerin, dolum makinesi gruplarına atanması ve akabinde atama yapılan grup içerisindeki paralel makinalarda çizelgelenmesi.

Problemi çözmek için, genelleştirilmiş atama problemine gömülen genel değişken komşuluk arama (gDKA) algoritması geliştirilmiştir. Algoritma iki ana kısımdan oluşmaktadır. İlk kısımda, makina gruplarına işlerin atanması DKA algoritması ile ikinci kısımda (iç döngüde) ise, iş kümelerinin paralel makinalarda çizelgelenmesi gene DKA algoritmasına dayanan liste çizelgeleme yöntemi ile yapılmıştır. Ayrıca aynı problemi çözmek için ayrık yapay arı kolonisi algoritması ve genetik algoritma geliştirilmiştir.

Anahtar Sözcükler: Genelleştirilmiş atama problemi, özdeş paralel makina çizelgeleme, değişken komşuluk arama yöntemi, sezgisel optimizasyon.

ABSTRACT

ASSIGNMENT AND SCHEDULING PROBLEM IN IDENTICAL PARALLEL MACHINES

KIZILAY, Damla

MSc in Industrial Engineering
Supervisor: Assoc. Prof. Dr. M. Fatih TAŞGETİREN
Co-Advisor: Assist. Prof. Dr. Önder BULUT
January, 2014

This paper presents a discrete artificial bee colony algorithm to solve the assignment and scheduling problem in DYO painting company. In the DYO Paint Company, there are three types of filling machines groups: automatic, semiautomatic and manual. In each group, there are several numbers of identical machines. The problem is to first assign these filling production orders to machine groups. Then, these filling production orders on each machine groups should be scheduled on identical parallel machines to minimize the sum of makespan and total tardiness. We also develop a traditional genetic algorithm and variable neighborhood search algorithm to solve the same problem. The computational results show that the VNS algorithm slightly outperforms the GA and DABC on set of benchmark problems we generated.

Keywords: Generalized assignment problem, identical parallel machine scheduling, variable neighborhood search algorithm, and heuristic optimization.

ACKNOWLEDGEMENTS

First of all, I would like to thank to my advisor, Assoc. Prof. Dr. M. Fatih Taşgetiren. Without his quick understanding and practical advices on the issues, I could not have proceeded in this study.

Secondly, I would like to thank to my co-advisor Assist. Prof. Dr. Önder Bulut. Working under his guidance made me understand all issues clearly and learn much.

Finally, I would like to thank to employees of Dyo Paint Factory, Bilgehan Bostan and Mehmet Akif Yılmaz who supported and helped me for the part of data collection.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INDEX OF SYMBOLS AND ABBREVIATION

Symbols	**Explanations**
C_{max}	Makespan
C_{max}^{max}	Maximum of the maximum completion time
π	Set of partial jobs
$\emptyset_{ij}$	Uniform random number
S_i	Strategy
NP	Population size
$R//C_{max}$	Unrelated Parallel Machine Scheduling Problems with a goal of finding the shortest completion time

Abbreviations

GAP	Generalized Assignment Problem
ABC	Artificial Bee Colony
$DABC$	Discrete Artificial Bee Colony
GA	Genetic Algorithm
VNS	Variable Neighborhood Search
TWT	Total Weighted Tardiness
$MILP$	Mixed Integer Linear Programming

CHAPTER 1

INTRODUCTION

This work is based on SANTEZ project and supported by DYO Paint Factory and ministry of science, industry, and technology. This assignment and scheduling problem is applied for DYO Paint Factory in filling machine units. DYO Paint Factory is the first domestic brand of Turkish paint industry. DYO Paint Factories Industry and Trade SA were founded in 1954. There are three factories which are located in Çiğli, Gebze and Manisa. Yasaş, Bayraklı and Akril companies under the management of Yaşar Paint Group were merged under the name of DYO Paint Factories Industry and Trade SA in 2002. After this merging, the existing activities of these three companies are performed in three different business units in a more efficient way. Construction paints and substructure materials under the trade name of DYO and marine coatings under the trade name of DYO are manufactured and marketed in the business unit for construction paints. The business unit for construction paints has a modern factory, which can compete internationally with its automation level, in Gebze. Product range of the factory exceeds 2,500 in color and package and its annual manufacturing capacity is 120.000 tons. Beside these activities, high quality application tools such as brush and roller are manufactured for housepainters. Also, DYO-Wagner paint application machines have been introduced to the market in cooperation with Wagner.

In the DYO Paint Company there are different types of paints and different types of packages. There are four types of packages and these are:

- 1/1 1 kg
- Gallon 2-3 kg
- Can 15-30 kg
- Barrel 150-250 kg

Also, in DYO Company there are three types of filling machines groups: automatic, semiautomatic and manual. In each group, there is several numbers of identical machines. The production orders for the filling machines are generated by SAP software and they are sent to filling machine units. In the thesis it is assumed that all kinds of packages can be filled by all types of filling machine groups. Currently, the scheduling and assignment processes of production orders are based on experience. Production orders are scheduled without using any heuristic or meta-heuristic algorithm. They are just based the urgency of production orders. The production orders are accumulated in front of the filling machines and this causes bottlenecks.

Problem is handled by considering two stages. The first stage is the assignment stage which assigns each production orders to the machine groups. The following figure explains the assignment stage.

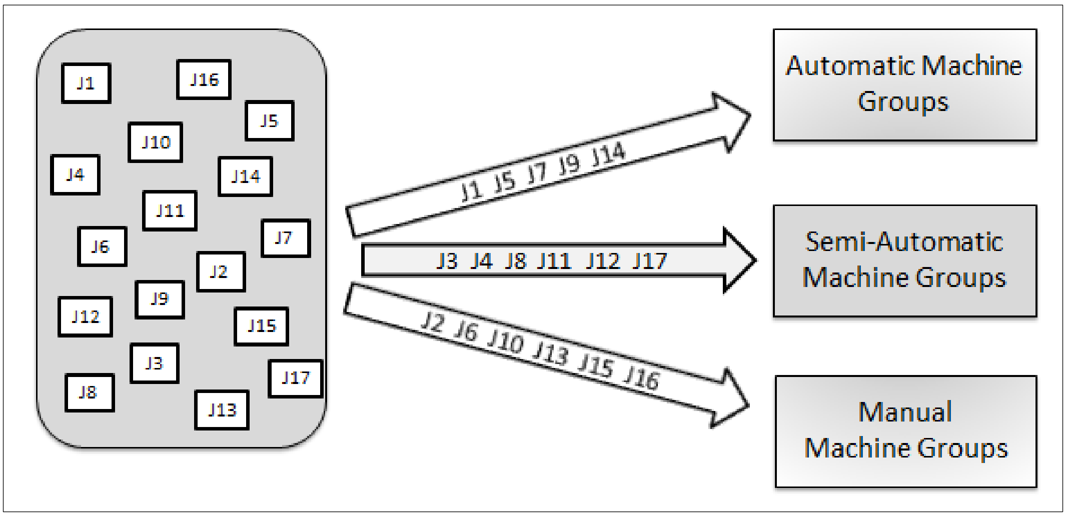

Figure 1. Assignment Stage of the Problem

In the left side of the figure 1, all production orders, which are generated by SAP software, are obtained. These production orders are formed by matched paint and package types. The figure is created as an example and there are 17 production orders and 3 machine groups. In DYO Company there are also 3 machine groups but more than 17 production orders in a week. As shown in the figure 1, all production orders are assigned to each machine group. One production order can be assigned to one machine group. In each machine group the filling times are different, but filling process is same. For this reason, we cannot assign a production order to more than one machine group. In the thesis work, production orders are called as jobs. The assigned production orders in each machine group are called as partial job sets, so there are three partial job sets.

The second stage is the scheduling stage which is about scheduling of the partial jobs in each machine group having identical parallel machines. In each machine group there are several identical machines. According to data from DYO Company, for the automatic and semi-automatic machine groups, there are 9 identical parallel machines. For the manual machine group there are 4 identical parallel machines. Same scheduling process is done for each machine group. In the figure 2, scheduling stage is explained for the manual machine group.

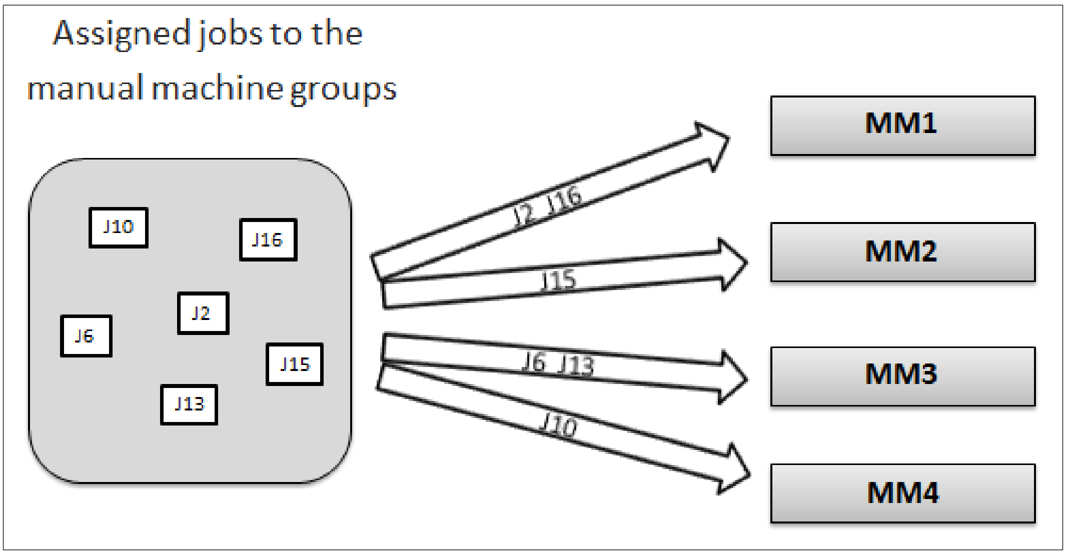

Figure 2. Scheduling Stage of the Problem

In the left side of the figure 2, there are 6 jobs which are assigned to manual machine group in the first stage (assignment stage). J2, J6, J10, J13, J15, and J16 are called as partial job set. MM1, MM2, MM3 and MM4 are the identical parallel machines in manual machine group. MM stands for manual machine in the figure. This job set is scheduled to these parallel machines using heuristic methods.

CHAPTER 2

LITERATURE REVIEW

Parallel machine scheduling has a wide range of literature. In DYO Company, the parallel machines are identical so a part of literature which includes parallel machine scheduling for identical machines is explained.

First study about parallel machine was at the end of the 50s by (McNaughton, 1959). After that, it is concentrated on the rules of assigning the jobs, which does not have any priorities, to the identical machines by (Graham L. , 1969). At scheduling the parallel machines procedure, while n jobs are scheduling m identical machines, total weighted tardiness and total flow time or maximum completion time functions are minimized. For the single machine systems, (Du & Leung, 1990) stated that minimizing the total tardiness with scheduling is NP-hard problem. Under the same conditions, scheduling problem in identical parallel machines is also strongly NP-hard problem. For this reason, the algorithms which are deterministic have some constraints for some special issues like common due date and equal processing times (Root, 1965), (Lawler, 1977), (Elmagraby & Park, 1974), (Dessouky, 1998). Therefore, many researchers are concentrated on heuristic methods. Many heuristic methods are based on List scheduling method in which, the jobs are sorted using a rule and based on this rule, they are assigned to the machines according to their earliest time to finish. These kind of heuristic methods are studied in by (Wilkerson & Irwin, 1971), (Dogramaci & Surkis, 1979), (Ho & Chang, 1991), (Koulamas C. P., 1994). In addition, a decomposition heuristic and hybrid simulated annealing heuristic are proposed by (Koulamas C. , 1997). Also, for the scheduling problem in parallel machines which has objective to minimize the total tardiness, Genetic Algorithm was used by (Bean, 1994). For minimizing the completion time of the parallel machine flowshop scheduling problem, tabu search was used by (Nowicki & Smutnicki, 1998). Tabu search and simulated annealing algorithms are compared by (Park & Kim, 1997). Recently, hybrid heuristic algorithm was proposed by (Anghinolfi & Paolucci, 2007). In order to minimize total tardiness of parallel machine problems, that include non-cumulative setup times, tabu search was used by (Bilge, Kyrac, Kurtulan, & Pekgun, 2004).

Moreover, in the literature there are wide range of topics about the usage of the algorithms such as insertion (taking off a job from its position and replacing it in a new position) and swap (switching the positions of two or more jobs) for the single machine system that minimizes the total weighted tardiness. The most known and exact solution algorithm of this problem in the parallel machine systems was proposed by (Pessoa, Uchoa, Aragao, & Rodrigues, 2008). This algorithm can find solution to the problems that have up to 50 jobs.

3.1 GENERALIZED ASSIGNMENT PROBLEM (GAP)

The main purpose in GAP is to assign a set of tasks to a set of agents with a minimum total cost. In each agent, there is a single resource and the resources in the agents have limited capacity. Each tasks that are assigned to an agent, needs a certain number of resource. GAP can be applied to several problems such as location problems, vehicle routing, group technology, and scheduling. Extended review of GAP and its applications was presented in (Martello & Toth, 1981) and (Cattrysse, Salomon, & Wassenhove, 1994). Several exact algorithms for GAP were proposed by (Ross & Soland, 1975), (Fisher, M. L.; Jaikumar, R.; Wassenhove, L.N. Van;, 1986), (Savelsbergh, 1997), and (Nauss, 2003). Several heuristic algorithms for GAP were proposed. Simulated annealing and tabu search algorithms were developed to solve GAP by (Osman, 1995). A genetic algorithm which tries to improve feasibility and optimality simultaneously for GAP was presented by (Chu & Beasley, 1997). Several meta-heuristic approaches were proposed for GAP, such as; Different variable depth search algorithms (Yagiura, M.; Yamaguchi, T.; Ibaraki, T., 1998), (Yagiura, M.; Yamaguchi, T.; Ibaraki, T., 1999) and another variable depth search algorithm (Racer & Amini, 1994), ejection chain based tabu search algortihms (Laguna, Kelly, Gonzalez-Valerde, & Glover, 1995), (Diaz & Fernandez, 2001), and (Yaguira, Ibaraki, & Glover, 2004), ant colony optimization (Randall, 2004), max-min ant system based on greedy randomized adaptive heuristic (Lourenço & Serra, 2002), genetic algorithm with constraint ratio heuristic (Feltl & Raidl, 2004), path relinking approaches (Alfadari, Plateau, & Tolla, 2004), (Yagiura, M.; Ibaraki, T.; Glover, F., 2006), Lagrangian heuristic algorithm (Haddadi & Ouzia, 2001).

Generalized assignment problem is known to be NP-hard by (Fisher, M. L.; Jaikumar, R.; Wassenhove, L.N. Van;, 1986), and (Sahni & Gonzalez, 1976). The GAP can be formulated as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} , \qquad (1)$$

$$subject\ to \sum_{i=1}^{n} a_{ij} x_{ij} \leq b_j \quad \forall j, \qquad 1 \leq j \leq m, \qquad (2)$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i, \qquad 1 \leq i \leq n, \qquad (3)$$

$$x_{ij} \in \{0,1\} \quad 1 \leq i \leq n \quad \forall i, \quad 1 \leq j \leq m \quad \forall j, \qquad (4)$$

In the formulation I is the set of tasks $I = \{1,2,\dots,n\}$; J is the set of agents $J = \{1,2,\dots,m\}$; b_j denotes the resource capacity of each agent j $(b_j \geq 0)$; a_{ij} denotes the needed amount of resource if task i is assigned to agent j $(a_{ij} \geq 0)$; c_{ij} denotes the cost of assigning task i to agent j $(c_{ij} \geq 0)$; x_{ij} is the decision variable:

$$x_{ij} = \begin{cases} 1, & if\ task\ i\ is\ assigned\ to\ agent\ j \\ 0, & otherwise \end{cases} \qquad (5)$$

In the formulation, objective tries to minimize total assignment cost, first constraint provides not to excess the resource capacity of each agent and second constraint provides that each task can be assigned to only one agent.

In our problem generalized assignment problem (GAP) is used, but with a small differences. The tasks can be thought as a production orders and agents can be thought as a machine groups. In our problem capacities of the machine groups are unlimited, so we do not need to use b_j and a_{ij} in our formulation.

3.2 UNRELATED PARALLEL MACHINES

In the parallel machine scheduling problem there are n independent jobs and m parallel machines. The jobs are processed on the parallel machines. Each job can be processed by only one machine while, a machine can process only one job at a time. If a job starts to be processed on a machine it has to be continued until completion. The set N of n jobs can be shown as $j = 1,2, \dots, n$ and the set M of m jobs can be shown as $i = 1,2, \dots, m$. The processing time of each job is known, finite and denoted as p_j. For the uniform parallel machine case, in each machine, the processing speed is different for the same job, and it is denoted as s_i. Therefore, the processing time of a job j on machine i can be derived as $p_{ij} = p_j/s_i$. Most generally, processing time of each job depends on the machine where it is processed and this referred to as the unrelated parallel machine scheduling problem. The data used for this problem is n, m and the matrix of the processing times p_{ij}. In parallel machine scheduling problems the commonly studied objective is to minimize the maximum completion time C_{max}. For the $\alpha/\beta/\gamma$ secheduling problems classification scheme is proposed initially by (Graham, Lawler, Lenstra, & Rinnooy, 1979).

The $R//C_{max}$ problem is an assignment problem, because the processing orders of the jobs assigned to a machine do not alter the maximum completion time at that machine. There are m^n possible solutions to the problem after all possible assignments. Therefore, the $R//C_{max}$ problem is shown to be NP-hard by (Garey & Johnson, 1979). Also, the two machine version $P2//C_{max}$ is shown as NP-hard by (Lenstra, Rinnooy, & Brucker, 1977). In addition, no polynomial time algorithm exists for the general $R//C_{max}$ problem with a better worst case ratio approximation than $3/2$ unless $P = NP$, according to (Lenstra, J. K.; Shmoys, D. B.; Tardos, E., 1990). The Mixed Integer Linear Programming (MILP) formulation for the $R//C_{max}$ is shown below:

$$\min C_{max} \tag{6}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \forall j \in N, \tag{7}$$

$$\sum_{j=1}^{n} p_{ij} x_{ij} \leq C_{max} \quad \forall i \in M, \tag{8}$$

$$x_{ij} \in \{0,1\} \quad \forall j \in N, \qquad \forall i \in M. \tag{9}$$

In the formulation, the first constraint provides that one job can be processed by only one machine. Second constraint provides that the total processing times of assigned jobs on their machines must be smaller than the maximum completion time, for each machine. The decision variable x_{ij} is a binary variable;

$$x_{ij} = \begin{cases} 1, & if\ job\ j\ is\ assigned\ to\ machine\ i \\ 0, & otherwise \end{cases} \tag{10}$$

In our work, unrelated parallel machine scheduling can be applied for the machine groups. In each machine group we have identical parallel machines which have the same speed. However, Each machine group have different processing speeds. Their speed is decreasing respectively for automatic, semi-automatic, and manual machine groups.

In order to solve our problem bi-level optimization method is used and our formulation includes both assignment and scheduling parts of the problem. The formulation of the problem is explained detailed in the following sections.

In the project, for the solution of the problem under consideration, a novel algorithm with a combination of several heuristic methods was used. For the jobs, which are assigned in the first phase, variable neighborhood search algorithm (VNS), genetic algorithm (GA) and discrete artificial bee algorithm (DABC) which consists of shift and swap operations are applied and the current solution is tried to improve. In the second phase, for the scheduling part of the jobs variable neighborhood search algorithm (VNS) which consists of insert and swap operations is applied.

The remaining paper is organized as follows. Chapter 3 introduces the problem definition whereas Chapter 4 introduces the proposed algorithms; discrete artificial bee algorithm, genetic algorithm and variable neighborhood search algorithm. Chapter 5 discusses the computational results over benchmark problems in total of both objectives – makespan and total weighted tardiness. Finally, Chapter 6 summarizes the concluding remarks.

CHAPTER 3

PROBLEM DEFINITION

The problem will be handled by considering two stages: Assignment of the jobs to the machine groups and then Scheduling of the partial jobs in each machine group having identical parallel machines. In order to solve problem, a variable neighborhood search (VNS) algorithm embedded in generalized assignment problem (GAP) will be developed. The algorithm has two phases. First phase, the production orders (jobs) will assigned to machine groups. Second phase, the partial jobs in each machine group will be scheduled by using a variable neighborhood search (VNS) algorithm with list scheduling approach.

Both assignment and scheduling parts included in the model formulation as follows:

3.1 PROBLEM FORMULATION

The aim of the problem is to assign job to the machine groups and then schedule the jobs which are assigned. While designing the problem the objective is to minimize the maximum completion time and total weighted tardiness. First of all, all the parameters are defined:

$j, k =$ Production orders (jobs) $j, k = 0,1, \ldots, n$

$g =$ Machine groups $g = 1,2,3$

$i =$ Parallel machines in machine group g

$i = 1, \ldots, 9 \; for \; g = 1$

$i = 1, \ldots, 9 \; for \; g = 2$

$i = 1, \ldots, 4 \; for \; g = 3$

$m =$ Total number of parallel machines

G = Total number of machine groups

p_{jg} = Processing time of job j in machine group g

W_j = Weight of job j

D_j = Due date of job j

$C_j = Completion\ time\ of\ job\ j$

Tardiness can be computed as follows:

$$T_j = max\{0, C_j - D_j\} \tag{11}$$

Decision variables are defined as follows:

$$Y_{jgi} = \begin{cases} 1, & If\ job\ j\ is\ assigned\ to\ i^{th}\ machine\ in\ g^{th}\ machine\ group \\ 0, & otherwise \end{cases}$$

$$X_{j0gi} = \begin{cases} 1, & if\ job\ j\ is\ the\ first\ job\ to\ be\ processed\ at\ the\ i^{th} \\ & machine\ in\ g^{th}\ machine\ group \\ 0, & otherwise \end{cases}$$

$$X_{jkgi} = \begin{cases} 1, & if\ both\ jobs\ j\ and\ k\ are\ assigned\ to\ the\ i^{th} \\ & machine\ in\ g^{th}\ machine\ group\ and\ j\ is\ prcessed \\ & just\ before\ k \\ 0, & otherwise \end{cases}$$

Proposed model of our problem is given below:

$$Min\ (makespan\ and\ TWT) = \alpha(C_{max}) + (1-\alpha)\left(\sum_{j=1}^{n}(T_j W_j)\right)$$

$Subject\ to$:

$$\sum_{g=1}^{G}\sum_{i=1}^{m} Y_{jgi} = 1 \qquad \forall j \quad (her\ iş\ tek\ makinede, tek\ makine\ grubunda) \tag{12}$$

$$C_j \geq p_{jg} * x_{0jgi} \qquad \forall j, g, i \tag{13}$$

$$C_k \geq (C_j + p_{kg}) * x_{jkgi} \qquad \forall i, j, k, g \tag{14}$$

$$x_{0jgi} + \sum_{k=1}^{n} \sum_{g=1}^{G} \sum_{i=1}^{m} X_{jkgi} = Y_{jgi} \qquad \forall j \tag{15}$$

In the model, objective function tries to minimize maximum completion time and total weighted tardiness. The Equation (12) is first constraint provides that each job can be processed by only one machine in one machine group. Equation (13) is the second constraint means that if a job is the first job at a machine in machine groups then its completion time must be greater than or equal to it processing time of assigned machine group. Equation (14) is the third constraint means that if a job comes after the other jobs at a machine in each machine group, its completion time must be greater than or equal to total of its processing time in assigned machine group and the completion time of the previous job. Equation (15) is the last constraint states that a job at a machine i in a machine group g can only be a first job or an intermediate job.

The first part in the objective function tries to maximize the machine utilization and the second part tries to minimize the total weighted tardiness and this minimization provides customer satisfaction. To solve the problem described above, we propose single variable neighborhood search algorithm (VNS) and populated based algorithms such as a discrete artificial bee colony algorithm and genetic algorithm. Their details are given in subsequent sections.

CHAPTER 4

ALGORITHMS

4.1 Traditional Discrete Artificial Bee Colony (ABC) Algorithm

In the ABC model, the colony consists of three groups of bees: employed bees, onlookers and scouts (Karaboga, 2005). In the model, there is only one food source for each artificial employed bee so each solution in the population is assumed to be food source. The number of solutions in the population is equal to the number of food sources and represented by D-dimensional real-valued vector. ABC algorithm is stated to be an iterative process (Karaboga, 2005), (Karaboga, D.; Basturk, B., 2007), (Karaboga, D.; Basturk, B., 2008), (Karaboga, D., 2009), (Karaboga, D.; Akay, B., 2009), (Karabulut & Tasgetiren, 2009). The outline of the ABC algorithm is given below:

Initial food sources of the basic ABC algorithm are randomly created according to the range of the boundaries as follows:

$$x_{ij} = x_j^{min} + \left(x_j^{max} - x_j^{min}\right) \times r \quad (16)$$

In the equation, NP represents the number of food sources so $i = 1, \ldots, NP$; D represents the number of decision variables so $j = 1, .., D$; and r represents a uniform random number between 0 and 1. In the initial population a counter value 0 is used for each food source, i.e. $count_i = 0$. After generating the initial population, search process is applied for the solutions in the population. This process includes three groups such as the employed bees, the onlooker bees and the scout bees. For the each group, there is a cycle goes over again until a maximum cycle number (MCN) is achieved.

In the employed bee phase, we generate the neighboring food source according to the equation below:

$$v_{ij} = x_{ij} + \emptyset_{ij}\left(x_{ij} - x_{kj}\right) \quad (17)$$

In the equation j is an integer value between 1 and D, and is chosen randomly. $k \in \{1, .., NP\}$ is a food source which is different than the food source x_i, and is also chosen randomly from the population. $\emptyset_{ij}$ is a uniform random number which is generated in the range $[-1,1]$. By changing only a parameter value of x_i, a new food source v_i is generated. Provided that, this changed parameter value does not fit the boundaries, the random one is kept in the memory which is generated according to (8).

If the v_i fits the boundaries, then its fitness value is obtained according to the equation below:

$$fitness_i = \begin{cases} 1/(1+f_i) & if\ f_i \geq 0 \\ 1+abs(f_i) & if\ f_i < 0 \end{cases} \tag{18}$$

This equation is defined for the minimization problem. In the equation, objective function value of the new food source v_i is represented by f_i. After that, the fitness values of the previous food source x_i and the new food source v_i are compared in order to select the better one in a greedy manner. If an improvement occurs for the x_i, the counter $count_i$ is kept as 0, else it is increased to 1. For all the employed bees in the population, same process goes over again.

In the onlooker bee phase, the roulette wheel selection is used and for the each food source a probability is generated as follows:

$$p_i = \frac{fitness_i}{\sum_{i=1}^{NP} fitness_i} \tag{19}$$

A uniform random number r, which is generated in the range “0-1”, is assigned to each food source x_i. Provided that r is smaller than the probability p_i, a neighboring food source is generated according to (8). The same greedy selection is applied to the solutions. If an improvement occurs for the x_i, the counter $count_i$ is kept as 0, else it is increased to 1. For all the onlooker bees in the population, same process goes over again.

In the scout bee phase, the sources abandoned are determined according to the counter of each solution. Determination is done through the comparison between the value of the counter $count_i$ and the control parameter "$limit$". Having greater $count_i$ than the "$limit$" means that the food source x_i is abandoned. In order to provide diversification for the ABC algorithm, abandoned x_i is forgotten and the new one is generated instead by using (8).

In order to apply ABC algorithm for both discrete and continuous decision variables of the ELSP, some unique modifications are proposed. Because, the original structure of the ABC algorithm can be applied for the real-parameter optimization problems and it is impossible to apply it for discrete/combinatorial problems. These required unique modifications are explained below.

4.1.1 Discrete ABC

In the discrete version, we still follow the basic framework of the original one as follows:

1. Initialize the population.
2. Employed bee phase to exploit the food sources.
3. Onlooker bee phase to search for new food sources.
4. Scout bee phase to search for new food sources.
5. Keep the best food source found so far.
6. If a termination criterion has not been satisfied, go to step 2; otherwise stop the procedure and report the best food source found so far.

4.1.2 Solution Representation

We employ a unique solution representation inspired by the Generalized Assignment Problem (Fisher & Jaikumar, 1981), (Yagiura, Yamaguchi, & Ibaraki, 1998), (Yagiura, M.; Ibaraki, T.; Glover, F., 2004), (Yagiura, M.; Ibaraki, T.; Glover, F., 2006). For example, For 15 production orders and 3 machines groups, the solution representation is given in Figure 1:

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_i	3	2	2	1	1	3	1	3	3	1	2	1	3	2	2
p_j	2	5	4	9	7	1	10	2	3	8	5	9	1	4	6
d_j	3	8	6	15	11	2	15	3	5	12	8	15	2	6	9

Figure 3. Example of Solution Representation

In Figure 1, $x_1 = 1$ represents the manual machine group, $x_2 = 2$ represents the semi-automatic machine group and $x_3 = 3$ represents the fully automatic machine group. Due dates are computed as $d_j = p_j \times k$ where due date tightness factor is taken as $k = 2$.

4.1.3 Initial Population

In the DABC algorithm, the initial population is established randomly in the range "1-3". For each food source in population, one strategy amongst three is assigned to each food source randomly. These strategies generating new food sources will be explained later on.

In order to show the assignment part small example is given below. In the following example, we show how the solution representation works.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_i	3	2	2	1	1	3	1	3	3	1	2	1	3	2	2
p_j	2	5	4	9	7	1	10	2	3	8	5	9	1	4	6
d_j	3	8	6	15	11	2	15	3	5	12	8	15	2	6	9

Partial set of jobs, $\pi_1 = \{4,5,7,10,12\}$, $\pi_2 = \{2,3,11,14,15\}$, $\pi_3 = \{1,6,8,9,13\}$.

We assume that there are two parallel machines in each machine groups. So these partial job sets will be scheduled on parallel machine by using a list-scheduling approach as follows:

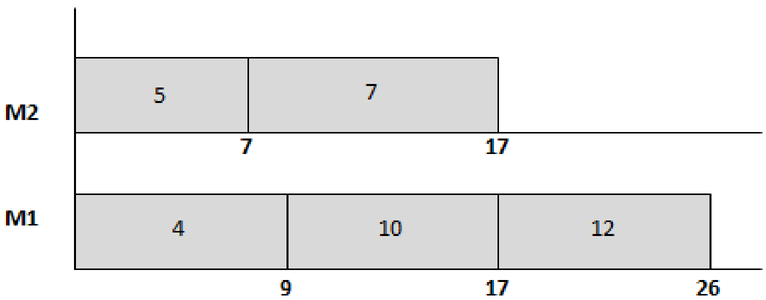

Figure 4. Manual Machines

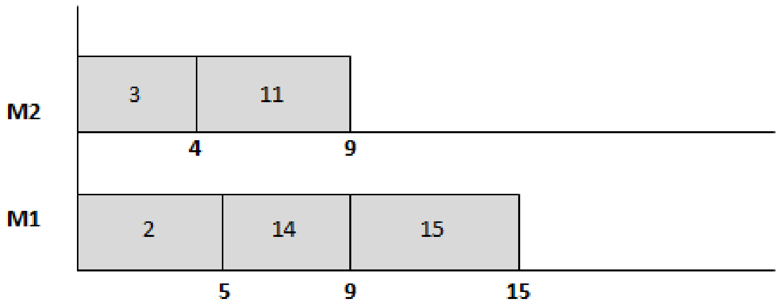

Figure 5. Semi-automatic Machines

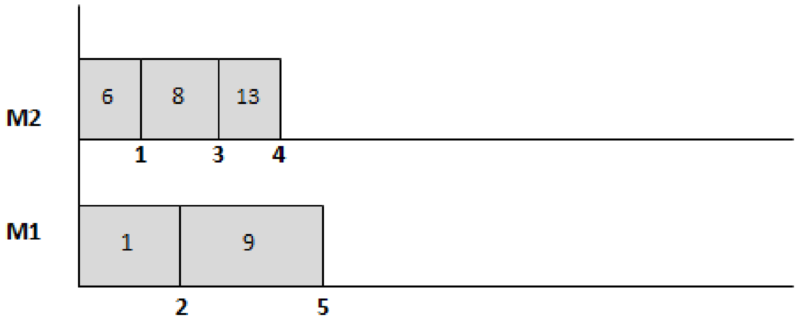

Figure 6. Automatic Machines

4.1.4 Neighborhood Structures

As the neighborhood structures, we employ shift and swap moves in the DABC algorithm as shown below:

$$N_1(x) = shift(x)$$
$$N_2(x) = swap(x)$$

A shift move means that a randomly selected machine group is changed to another machine group. A swap move means that two machine groups are exchanged in the solution. Following example illustrates shift and swap moves in the solution *s*:

$$N_1(x) = shift(x)$$

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_i	3	2	2	**1**	1	3	1	3	3	1	2	**1**	3	2	2
x_i	3	2	2	**3**	1	3	1	3	3	1	2	**3**	3	2	2

Figure 7. Shift Machine Groups

Partial set of jobs, $\pi_1 = \{5,7,10\}$, $\pi_2 = \{2,3,11,14,15$, $\pi_3 = \{1,4,6,8,9,12,13\}$.

$N_2(x) = swap(x)$

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_i	3	2	2	**1**	1	3	1	3	3	1	2	1	**3**	2	2
x_i	3	2	2	**3**	1	3	1	3	3	1	2	3	**1**	2	2

Figure 8. Swap Machine Groups

Partial set of jobs, $\pi_1 = \{5,7,10,13\}$, $\pi_2 = \{2,3,11,14,15\}$, $\pi_3 = \{1,4,6,8,9,12\}$

4.1.5 Employed Bee Phase

In the employed bee phase, new food sources are obtained through some strategies around the neighborhood of the current position. We employ three types of neighborhood structures. These structures are based on shift and swap operators. S_i Using these strategies denoted as S_i , new food sources in the neighborhood are obtained for the employed bees as follows::

S_1: Applying one shift, one swap move to the solution x_i.

S_2: Applying two shift, two swap moves to the solution x_i.

S_3: Applying three shift, three swap moves to the solution x_i.

In each strategy, it is possible to have different performances, so in the population, for each individual (food source), Neighboring food source is obtained by the strategy assigned to each individual.

After obtaining a neighboring food source, we apply a variable neighborhood search algorithm (VNS) (Mladenovic & Hansen, 1997) to the new food source to further enhance the solution quality. For the selection, a new source will always be accepted if it is better than the current food source.

4.1.6 Onlooker Bee Phase

In the onlooker bee phase, a food source x is determined by the tournament selection of size 2. In the tournament selection, two food sources are randomly chosen from the population, and the better one is chosen according to their fitness values. Then, similar to the employed bee phase, corresponding strategy is applied to the food source selected. After applying the corresponding strategy, the VNS algorithm is applied to the food sources.

4.1.7 Scout Bee Phase

In the scout bee phase, a tournament selection with the size of 2 is again used to discard the worse of two randomly selected food sources that have been picked out from the population. Then, the scout obtains a food source by the strategy assigned to it.

4.1.8 Variable Neighborhood Search

The following VNS local search described in (Tasgetiren, Liang, Sevkli, & Gencyilmaz, 2007) is employed in both the employed bee and onlooker bee phases. The aim is to further improve the objective function on the partial job sets. Sequentially, the VNS local search is applied to each partial job set. As the neighborhood structures, single insert or swap move is applied to the permutation in the each partial job set. The VNS local search is given in Figure 7.

$\boldsymbol{VNSListScheduling}(\boldsymbol{\pi_i}\ \boldsymbol{and}\ \boldsymbol{n_i}\ \in \boldsymbol{x})$

$d_{max} = 2$

$for\ i = 1\ to\ g$

$\pi_i\ ve\ n_i\ \in (x)$

$\pi = \pi_i^* = \pi_i$

$Endfor$

$for\ i = 1\ to\ g\{$

$d = 1$

$do\{$

$\pi_1 = N_d(\pi)$ $\quad$ % $N_1(\pi_i) = Insert(\pi_i)$

$if\ f(\pi_1) < f(\pi_i^*)\ then$ $\quad$ % $N_2(\pi_i) = Swap(\pi_i)$

$\pi_i^* = \pi_1$

$d = 1$

$else$

$d = d + 1$

$\}while\ (d \leq d_{max})$

$\}Endfor$

$Return\ f(x) \in (f(\pi_i^*)\ for\ i = 1, .., g)$

Figure 9. Referenced Local Search

Computational Procedure of DABC Algorithm

Procedure ABC

Step 1. Set parameters NP and S_{max}

Step 2. Establish initial population randomly

Step 3. Assign a strategy to each food source in the population randomly

Step 4. Evaluate population and find x_{best}

Step 5. *Repeat the following for each employed bee* x_i *(Employed Bee Phase)*

a. *Generate a new food source by startegy* $x_{new} = S_i(x_i)$
b. $x_{new} = Apply\ VNS\ ListScheduling(x_{new})$
c. $if\ f(x_{new}) < f(x_i),\quad x_i = x_{new}$
d. $if\ f(x_{new}) < f(x_{best}),\quad x_{best} = x_{new}$

Step 6. Repeat the following for each onlooker bee x_k *(Onlooker Bee Phase)*

a. *Select a food source by tournament selection* $x_k = TS_{best\ of\ two}(x_k \in NP)$
b. *Generate a new food source by startegy k* $x_{new} = S_k(x_k)$
c. $x_{new} = Apply\ VNS\ ListScheduling(x_{new})$

d. $if\ f(x_{new}) < f(x_k),\ x_k = x_{new}$

e. $if\ f(x_{new}) < f(x_{best}),\quad x_{best} = x_{new}$

$Step\ 7.\ Repeat\ the\ following\ for\ each\ scout\ bee\ x_k$

$(Scout\ Bee\ Phase)$

a. $Select\ a\ \ food\ source\ by\ tournament\ selection$

$x_k = \quad TS_{worst\ of\ two}(x_k \in NP)$

b. $Generate\ a\ new\ food\ source\ by\ startegy\ k$

$x_{new} = \quad S_k(x_k)$

c. $x_{new} = Apply\ VNS\ ListScheduling(x_{new})$

d. $x_k = x_{new}$

e. $if\ f(x_{new}) < f(x_{best}),\quad x_{best} = x_{new}$

f. $If\ the\ stopping\ criterion\ is\ not\ met, got\ to$

$Step\ 5, else\ stop\ and\ return\ x_{best}$

Figure 10. Outline of the ABC algorithm

4.2 GENETIC ALGORITHM

Genetic algorithms (GA) are a part of parallel search heuristics originated by the biological process of natural selection and evolution (Ruiz & Maroto, 2005). In GA optimization, solutions are coded into chromosomes in order to construct a population being evolved through generations. At each generation, we use crossover operator, which is a process of taking more than one parent solutions and producing a child solution from them. Then, mutation and perturbation occurs for some of the individuals. After that, they are gathered to select new individuals for next generation. This procedure is repeated until the stopping criterion is satisfied.

However, in the proposed GA, we take each individual and another individual with the tournament selection of size 2 to mate them. By using them, we generate an offspring with PTL crossover operator (Pan, Tasgetiren, & Liang, 2008). To consistent with the DABC algorithm, we compare the offspring with the ith individual, we replace the ith individual if better. This procedure is repeated until the stopping criterion is achieved. The following computational procedure explains the components of the proposed GA:

Step 1. Set the population size NP,

Step 2. Initialize the population randomly:

Step 3. For $i = 1,2, \ldots, NP$, repeat the following sub-steps:

a. For the individualx_i, select a mate $x_k$from the population by the tournament selection with size 2.

b. Produce a new offspring y_i by recombining them with PTL crossover

c. Mutate y_i with a mutation probability.

d. Evaluate the new offspring$y_i$and apply VNS ListScheduling to y_i.

e. If y_i is better than x_i, let $x_i = y_i$ and update best so far solution x_B.

Step 4. If the termination criterion is reached, return the best solution found so far x_B; otherwise go to Step 3.

4.3 VARIABLE NEIGHBORHOOD SEACH

Variable neighborhood search (VNS) is a common approach to enhance the solution quality with systematic changes of neighborhood within a local search. It is proposed by (Mladenovic & Hansen, 1997). The algorithm involves iterative exploration of larger and larger neighborhoods for a given local optima until there is an improvement, after which time the search is repeated.

VNS local search is employed in DABC and GA algorithms as it is explained in the previous sections. Besides the populated algorithms, it is also used to improve the single solution. In order to apply VNS to the single solution, the replication length is adjusted according to the populated ones. The aim is to further improve the objective function on the partial job sets. Sequentially, the VNS local search is applied to each partial job set. As the neighborhood structures, single insert or swap move is applied to the permutation in the each partial job set. Since VNS used in DABC and GA algorithms, we called these algorithms as DABC_VNS and GA_VNS in order to distinguish them from VNS, in the following sections.

CHAPTER 5

COMPUTATIONAL RESULTS

The DABC_VNS, GA_VNS and VNS algorithms were coded in Visual C++ and run on an Intel® Core™ i5-3360M CPU 2.80 GHz PC with 8 GB memory, 64 bit operating system. We generated our own benchmarks as follows: For automatic machines group, the processing times are generated between 5 and 11, for semi-automatic machines groups, processing times are generated between 11 and 16, for manual machines groups, processing times are generated between 16 and 21. In each machine group we have different number of parallel machines. For automatic machine group, the number of parallel machines is 9, for semi-automatic machine group, the number of parallel machines is also 9 and for manual machine group, the number of parallel machines is 4. These parallel machines numbers are taken according to the data gathered from DYO Paint Company. We devised 10 instances for 100 jobs, 200 jobs, 300 jobs, 400 jobs and 500 jobs. Since the objective function is bi-objective, we give a weight α to the first part of the objective function whereas $1 - \alpha$ is given to the second part. Results are generated for different α values, $\alpha = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$.

For each instance, we carried out 5 replications and we provide the average (Avg), minimum (Min), maximum (Max), standard deviation (Std) and central processing unit time (Cpu) of five runs. Also, the average of each instances are taken for each number of job. We fixed the population size at 10 for both algorithms. The computational results for three different algorithms (DABC_VNS, GA_VNS, and VNS) are given in tables below. Nonparametric Mann-Whitney Test is applied and the results are given for the different α values.

Table 1. Run Results for Each Algorithms with α=0

	$\alpha = 0$														
	DABC_VNS					**GA_VNS**					**VNS**				
Jobs	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu
100	1000,20	1225,00	1066,10	94,11	58,60	1017,10	1346,60	1125,50	142,98	96,45	1008,00	1018,90	1010,40	4,92	30,08
200	6717,90	8513,00	7349,90	746,90	203,70	7377,80	9562,90	8298,00	900,02	324,52	6402,60	6630,20	6464,70	100,78	114,97
300	18140,80	23709,80	20191,00	2291,88	424,72	20989,20	25387,60	22945,40	1889,15	690,07	16839,50	17668,00	17058,80	364,04	250,99
400	36440,10	46052,50	40506,00	3895,14	727,15	42973,10	48198,00	45425,30	2205,12	1160,36	32278,30	35270,60	33140,60	1284,09	407,93
500	60698,10	75276,50	66787,60	5846,81	1102,01	71631,90	78261,60	74589,50	2777,84	1806,59	53361,20	58462,20	54782,30	2149,50	635,43

Mann-Whitney Test and CI ($\alpha = 0$):

DABC_VNS; GA_VNS

	N	Median
DABC_VNS	5	18141
GA_VNS	5	20989

Point estimate for ETA1-ETA2 is -2848
96,3 Percent CI for ETA1-ETA2 is (-53491;39709)
W = 25,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

DABC_VNS; VNS

	N	Median
DABC_VNS	5	18141
VNS	5	16840

Point estimate for ETA1-ETA2 is 1301
96,3 Percent CI for ETA1-ETA2 is (-35221;43859)
W = 29,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,8345

GA_VNS; VNS

	N	Median
GA_VNS	5	20989
VNS	5	16840

Point estimate for ETA1-ETA2 is 4150
96,3 Percent CI for ETA1-ETA2 is (-32372;54792)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

Figure 11. Mann-Whitey Test Results for α = 0

Table 2. Run Results for Each Algorithms with α=0,1

	$\alpha = 0{,}1$														
	DABC_VNS					**GA_VNS**					**VNS**				
Jobs	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu
100	916,10	1060,10	955,70	62,45	59,59	939,90	1312,90	1059,90	159,65	96,27	904,30	916,50	906,90	5,43	33,95
200	6042,80	7904,50	6718,40	760,95	207,83	6743,70	8527,50	7500,90	765,14	330,33	5864,40	5970,90	5894,00	46,51	115,01
300	16519,10	21395,90	18386,20	1959,25	433,94	18899,40	23154,30	20691,80	1725,54	683,64	15301,80	15896,00	15453,50	254,38	239,02
400	32441,90	40375,50	35886,90	3166,65	730,14	37961,10	43288,60	40149,90	2158,85	1160,89	29145,10	31216,80	29704,50	877,87	413,80
500	53967,30	67117,70	59532,40	5190,48	1125,24	64563,10	70654,00	67293,30	2538,63	1779,56	48159,10	52159,20	49291,40	1692,03	644,52

Mann-Whitney Test and CI($\boldsymbol{\alpha = 0,1}$):

DABC_VNS; GA_VNS

	N	Median
DABC_VNS	5	16519
GA_VNS	5	18899

Point estimate for ETA1-ETA2 is -2380
96,3 Percent CI for ETA1-ETA2 is (-48044;35068)
W = 25,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

DABC_VNS; VNS

	N	Median
DABC_VNS	5	16519
VNS	5	15302

Point estimate for ETA1-ETA2 is 1217
96,3 Percent CI for ETA1-ETA2 is (-31640;38666)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

GA_VNS; VNS

	N	Median
GA_VNS	5	18899
VNS	5	15302

Point estimate for ETA1-ETA2 is 3598
96,3 Percent CI for ETA1-ETA2 is (-29260;49261)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

Figure 12. Mann-Whitey Test Results for α = 0,1

Table 3. Run Results for Each Algorithms with α=0,2

	$\alpha = 0,2$														
	DABC_VNS					**GA_VNS**					**VNS**				
Jobs	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu
100	831,10	1015,50	887,60	76,71	60,58	847,40	1172,00	952,70	137,10	97,10	822,10	834,10	824,80	5,46	30,93
200	5452,90	7093,30	6029,30	680,83	235,95	6170,30	7696,90	6711,80	646,07	325,40	5229,90	5338,30	5252,80	48,52	111,11
300	14496,20	19202,80	16365,00	1904,92	476,95	17182,90	20523,40	18524,70	1380,19	687,28	13711,90	14266,40	13845,80	240,30	241,61
400	28718,10	36117,90	31681,80	3007,98	751,75	34468,30	38410,70	36280,30	1629,25	1158,40	26043,60	27800,30	26527,00	734,97	418,72
500	48697,20	60246,40	53509,40	4605,25	1121,25	57153,50	63302,60	59776,40	2506,41	1833,37	42661,40	46311,60	43784,10	1520,29	653,53

Mann-Whitney Test and CI($\boldsymbol{\alpha = 0,2}$):

DABC_VNS; GA_VNS

	N	Median
DABC_VNS	5	14496
GA_VNS	5	17183

Point estimate for ETA1-ETA2 is -2687
96,3 Percent CI for ETA1-ETA2 is (-42657;31514)
W = 25,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

DABC_VNS; VNS

	N	Median
DABC_VNS	5	14496
VNS	5	13712

Point estimate for ETA1-ETA2 is 784
96,3 Percent CI for ETA1-ETA2 is (-28165;34985)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

GA_VNS; VNS

	N	Median
GA_VNS	5	17183
VNS	5	13712

Point estimate for ETA1-ETA2 is 3471
96,3 Percent CI for ETA1-ETA2 is (-25479;43441)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

Figure 13. Mann-Whitey Test Results for α = 0,2

Table 4. Run Results for Each Algorithms with α=0,3

	$\alpha = 0{,}3$														
	DABC_VNS					**GA_VNS**					**VNS**				
Jobs	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu
100	746,50	908,70	794,40	66,72	68,60	762,70	1014,50	844,40	107,17	98,94	739,80	754,30	743,00	6,47	32,06
200	4797,30	6330,30	5319,90	628,37	228,74	5477,80	6951,50	6120,00	606,31	345,72	4606,00	4736,60	4637,70	57,74	114,55
300	12837,80	16580,50	14354,90	1486,34	432,50	14969,30	17859,10	16268,40	1174,93	720,17	12005,20	12582,80	12154,70	248,89	248,12
400	25566,40	32250,20	28428,60	2687,41	740,31	30081,40	33760,00	31762,80	1524,95	1177,26	22762,20	24381,30	23179,40	690,86	431,44
500	42562,00	51923,90	46493,50	3739,78	1132,64	48675,20	54943,50	51632,10	2540,89	1806,78	37471,90	41073,70	38494,40	1508,77	655,09

Mann-Whitney Test and CI($\boldsymbol{\alpha = 0,3}$):

DABC_VNS; GA_VNS

	N	Median
DABC_VNS	5	12838
GA_VNS	5	14969

Point estimate for ETA1-ETA2 is -2131
96,3 Percent CI for ETA1-ETA2 is (-35837;27593)
W = 25,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

DABC_VNS; VNS

	N	Median
DABC_VNS	5	12838
VNS	5	12005

Point estimate for ETA1-ETA2 is 833
96,3 Percent CI for ETA1-ETA2 is (-24634;30557)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

GA_VNS; VNS

	N	Median
GA_VNS	5	14969
VNS	5	12005

Point estimate for ETA1-ETA2 is 2964
96,3 Percent CI for ETA1-ETA2 is (-22503;36670)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

Figure 14. Mann-Whitey Test Results for α = 0,3

Table 5. Run Results for Each Algorithms with α=0,4

	$\alpha = 0{,}4$														
	DABC_VNS					**GA_VNS**					**VNS**				
Jobs	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu
100	661,30	804,90	705,50	59,87	62,17	677,30	926,30	762,70	102,51	96,25	657,10	673,40	661,60	7,14	31,88
200	4146,20	5478,90	4609,90	540,45	214,71	4608,90	6022,90	5155,70	595,41	330,74	4020,40	4097,80	4039,30	34,09	116,47
300	11171,10	14441,30	12449,30	1310,32	452,52	12907,10	15619,40	14081,20	1131,72	717,76	10404,10	10683,60	10466,80	123,56	244,36
400	21731,30	27522,80	24157,10	2325,44	759,71	26001,90	29204,00	27536,10	1320,54	1269,81	19618,70	21075,40	20022,80	619,25	417,49
500	37060,10	45178,00	40483,80	3273,11	1194,34	43910,50	47179,50	45431,60	1336,93	1837,01	32033,90	35193,50	32877,40	1345,37	646,06

Mann-Whitney Test and CI($\alpha = 0,4$):

DABC_VNS; GA_VNS

	N	Median
DABC_VNS	5	11171
GA_VNS	5	12907

Point estimate for ETA1-ETA2 is -1736
96,3 Percent CI for ETA1-ETA2 is (-32739;24153)
W = 25,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

DABC_VNS; VNS

	N	Median
DABC_VNS	5	11171
VNS	5	10404

Point estimate for ETA1-ETA2 is 767
96,3 Percent CI for ETA1-ETA2 is (-20863;26656)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

GA_VNS; VNS

	N	Median
GA_VNS	5	12907
VNS	5	10404

Point estimate for ETA1-ETA2 is 2503
96,3 Percent CI for ETA1-ETA2 is (-19127;33507)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

Figure 15. Mann-Whitey Test Results for α = 0,4

Table 6. Run Results for Each Algorithms with α=0,5

	$\alpha = 0{,}5$														
	DABC_VNS					**GA_VNS**					**VNS**				
Jobs	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu
100	583,10	709,70	618,60	53,58	58,58	594,60	763,10	639,80	72,58	95,81	586,00	592,10	587,20	2,68	30,97
200	3536,40	4618,20	3910,30	439,53	200,50	3951,00	5081,90	4397,60	462,92	328,09	3415,90	3508,70	3439,80	40,97	108,69
300	9380,70	12188,10	10519,50	1139,34	420,61	10771,60	13014,80	11724,30	940,51	694,73	8669,60	9233,20	8813,70	241,97	235,68
400	18302,30	23086,50	20306,70	1906,69	719,32	21503,40	24252,10	22840,40	1136,55	1186,21	16414,40	17754,80	16820,20	588,41	418,17
500	30994,70	38144,20	34060,50	2871,37	1185,57	35755,00	39614,00	37387,30	1538,86	1812,71	26758,40	29625,00	27655,80	1194,20	637,38

Mann-Whitney Test and CI($\boldsymbol{\alpha = 0,5}$):

DABC_VNS; GA_VNS

	N	Median
DABC_VNS	5	9381
GA_VNS	5	10772

Point estimate for ETA1-ETA2 is -1391
96,3 Percent CI for ETA1-ETA2 is (-26374;20223)
W = 25,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

DABC_VNS; VNS

	N	Median
DABC_VNS	5	9380,7
VNS	5	8669,6

Point estimate for ETA1-ETA2 is 711,1
96,3 Percent CI for ETA1-ETA2 is (-17377,7;22324,9)
W = 29,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,8345

GA_VNS; VNS

	N	Median
GA_VNS	5	10772
VNS	5	8670

Point estimate for ETA1-ETA2 is 2102
96,3 Percent CI for ETA1-ETA2 is (-15987;27086)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

Figure 16. Mann-Whitey Test Results for α = 0,5

Table 7. Run Results for Each Algorithms with α=0,6

	$\alpha = 0{,}6$														
	DABC_VNS					**GA_VNS**					**VNS**				
Jobs	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu
100	499,70	602,50	530,80	42,53	64,44	510,60	682,60	573,50	73,78	104,30	497,90	504,40	499,30	2,87	30,57
200	2870,70	3737,20	3171,90	356,04	220,96	3153,80	4165,80	3489,40	422,15	357,55	2792,30	2863,20	2811,40	31,41	111,33
300	7609,80	9775,80	8451,60	879,95	477,16	8948,00	10539,80	9641,10	676,98	747,45	6976,30	7407,90	7097,40	182,26	239,51
400	14884,40	18923,00	16588,60	1641,86	813,70	17406,70	19854,40	18439,30	1004,39	1277,79	13164,30	14246,10	13481,80	471,70	412,85
500	24635,20	30569,10	27078,70	2384,81	1125,00	29643,00	31522,90	30290,60	762,50	1893,14	21959,70	23875,60	22426,20	830,27	637,18

Mann-Whitney Test and CI($\alpha = 0,6$):

DABC_VNS; GA_VNS

	N	Median
DABC_VNS	5	7609,8
GA_VNS	5	8948,0

Point estimate for ETA1-ETA2 is -1338,2
96,3 Percent CI for ETA1-ETA2 is (-22033,2;15687,1)
W = 25,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

DABC_VNS; VNS

	N	Median
DABC_VNS	5	7609,8
VNS	5	6976,3

Point estimate for ETA1-ETA2 is 633,5
96,3 Percent CI for ETA1-ETA2 is (-14349,8;17658,9)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

GA_VNS; VNS

	N	Median
GA_VNS	5	8948,0
VNS	5	6976,3

Point estimate for ETA1-ETA2 is 1971,7
96,3 Percent CI for ETA1-ETA2 is (-13011,8;22666,7)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

Figure 17. Mann-Whitey Test Results for α = 0,6

Table 8. Run Results for Each Algorithms with α=0,7

	$\alpha = 0{,}7$														
	DABC_VNS					**GA_VNS**					**VNS**				
Jobs	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu
100	421,30	523,30	450,80	42,51	67,54	426,60	561,20	473,30	56,51	102,58	412,80	423,80	415,10	4,95	30,99
200	2245,30	2887,10	2483,10	264,60	222,30	2482,00	3188,90	2769,50	291,11	340,53	2168,20	2229,10	2184,40	27,00	110,00
300	5835,50	7670,90	6541,30	735,04	466,08	6892,70	8147,30	7390,30	521,17	741,58	5376,40	5617,90	5439,80	107,24	236,19
400	11255,70	14195,30	12460,80	1186,71	723,92	13330,90	14884,40	13999,50	642,58	1152,57	10175,10	10991,00	10382,20	348,80	410,10
500	18745,40	23133,90	20559,50	1751,52	1125,24	21982,20	24100,50	22875,40	888,76	1767,12	16388,80	17992,90	16866,60	682,32	629,50

Mann-Whitney Test and CI($\alpha = 0,7$):

DABC_VNS; GA_VNS

	N	Median
DABC_VNS	5	5835,5
GA_VNS	5	6892,7

Point estimate for ETA1-ETA2 is -1057,2
96,3 Percent CI for ETA1-ETA2 is (-16146,8;11852,7)
W = 25,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

DABC_VNS; VNS

	N	Median
DABC_VNS	5	5835,5
VNS	5	5376,4

Point estimate for ETA1-ETA2 is 459,1
96,3 Percent CI for ETA1-ETA2 is (-10553,2;13369,1)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

GA_VNS; VNS

	N	Median
GA_VNS	5	6892,7
VNS	5	5376,4

Point estimate for ETA1-ETA2 is 1516,3
96,3 Percent CI for ETA1-ETA2 is (-9748,5;16605,8)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

Figure 18. Mann-Whitey Test Results for α = 0,7

Table 9. Run Results for Each Algorithms with α=0,8

	$\alpha = 0{,}8$														
	DABC_VNS					**GA_VNS**					**VNS**				
Jobs	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu
100	334,70	398,80	353,20	26,78	60,55	347,40	455,10	385,30	44,92	100,32	326,20	336,80	329,20	4,62	30,38
200	1604,70	2082,60	1773,00	193,09	212,96	1767,40	2251,10	1986,20	203,76	328,78	1543,50	1563,70	1547,60	8,98	109,55
300	4078,80	5248,50	4523,70	472,79	448,97	4756,00	5644,10	5138,90	362,95	667,93	3743,50	3878,80	3779,70	58,88	231,48
400	7833,30	9712,90	8626,60	758,74	705,58	9279,20	10205,00	9660,10	371,90	1271,20	6952,30	7543,50	7132,60	244,89	423,74
500	12769,90	15725,70	14005,60	14005,60	1096,15	15127,50	16572,30	15697,40	580,92	1738,41	11267,30	12442,80	11633,60	498,26	625,71

Mann-Whitney Test and CI($\alpha = 0,8$):

DABC_VNS; GA_VNS

```
               N  Median
DABC_VNS       5  4078,8
GA_VNS         5  4756,0
```

Point estimate for ETA1-ETA2 is -677,2
96,3 Percent CI for ETA1-ETA2 is (-11048,7;8013,9)
W = 25,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

DABC_VNS; VNS

```
               N  Median
DABC_VNS       5  4078,8
VNS            5  3743,5
```

Point estimate for ETA1-ETA2 is 335,3
96,3 Percent CI for ETA1-ETA2 is (-7188,4;9026,5)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

GA_VNS; VNS

```
               N  Median
GA_VNS         5  4756,0
VNS            5  3743,5
```

Point estimate for ETA1-ETA2 is 1012,5
96,3 Percent CI for ETA1-ETA2 is (-6604,9;11384,0)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

Figure 19. Mann-Whitey Test Results for α = 0,8

Table 10. Run Results for Each Algorithms with α=0,9

	$\alpha = 0{,}9$														
	DABC_VNS					**GA_VNS**					**VNS**				
Jobs	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu
100	250,30	304,60	268,80	22,23	59,66	255,30	334,80	280,70	33,68	95,07	242,20	246,50	243,20	1,97	29,81
200	965,50	1251,70	1070,10	117,25	202,20	1088,50	1354,20	1199,50	110,01	308,85	920,70	946,20	926,20	11,33	105,08
300	2273,40	2938,20	2540,60	267,37	421,99	2701,90	3116,60	2879,70	168,48	594,28	2099,90	2207,90	2127,50	46,20	222,24
400	4321,30	5377,00	4766,50	422,73	723,43	5019,20	5612,40	5322,10	233,62	1054,16	3798,60	4084,90	3880,50	121,20	386,46
500	6915,20	8466,90	7584,70	620,10	1142,90	8186,80	8831,20	8452,90	265,56	1654,96	6076,20	6585,20	6221,80	217,27	600,92

Mann-Whitney Test and CI($\alpha = 0,9$):

DABC_VNS; GA_VNS

	N	Median
DABC_VNS	5	2273,4
GA_VNS	5	2701,9

Point estimate for ETA1-ETA2 is -428,5
96,3 Percent CI for ETA1-ETA2 is (-5913,6;4213,4)
W = 25,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

DABC_VNS; VNS

	N	Median
DABC_VNS	5	2273,4
VNS	5	2099,9

Point estimate for ETA1-ETA2 is 173,5
96,3 Percent CI for ETA1-ETA2 is (-3802,8;4815,2)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

GA_VNS; VNS

	N	Median
GA_VNS	5	2701,9
VNS	5	2099,9

Point estimate for ETA1-ETA2 is 602,0
96,3 Percent CI for ETA1-ETA2 is (-3543,4;6087,0)
W = 30,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

Figure 20. Mann-Whitey Test Results for α = 0,9

Table 11. Run Results for Each Algorithms with α=1

	$\alpha = 1$														
	DABC_VNS					**GA_VNS**					**VNS**				
Jobs	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu	Min	Max	Avg	Std	Cpu
100	165,90	202,60	179,90	15,42	40,75	181,00	213,00	193,70	13,30	67,63	151,20	163,00	153,60	5,23	22,36
200	344,30	422,50	377,60	32,44	120,28	395,70	436,60	413,30	17,67	192,59	284,30	328,40	299,00	18,66	67,77
300	537,50	644,00	583,10	42,84	263,48	622,60	663,60	639,10	17,25	370,26	420,90	517,10	451,70	40,26	135,28
400	740,60	864,70	798,00	49,05	436,75	857,90	885,90	869,30	11,92	621,52	568,40	721,20	619,70	63,22	224,12
500	973,00	1085,60	1027,50	45,03	579,37	1083,90	1105,60	1091,90	9,13	903,04	717,20	913,20	792,50	77,59	343,77

Mann-Whitney Test and CI($\alpha = 1$):

DABC_VNS; GA_VNS

	N	Median
DABC_VNS	5	537,5
GA_VNS	5	622,6

Point estimate for ETA1-ETA2 is -85,1
96,3 Percent CI for ETA1-ETA2 is (-691,9;559,7)
W = 25,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,6761

DABC_VNS; VNS

	N	Median
DABC_VNS	5	537,5
VNS	5	420,9

Point estimate for ETA1-ETA2 is 116,6
96,3 Percent CI for ETA1-ETA2 is (-372,9;589,3)
W = 31,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,5309

GA_VNS; VNS

	N	Median
GA_VNS	5	622,6
VNS	5	420,9

Point estimate for ETA1-ETA2 is 201,7
96,3 Percent CI for ETA1-ETA2 is (-321,5;706,8)
W = 32,0
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at p = 0,4034

Figure 21. Mann-Whitey Test Results for α = 1

Non-parametric statistical test of null hypothesis named Mann-Whitney test is applied to compare the results of the three different algorithms. Mann-Whitney test can be performed to 2 sample rank test of the equality of two population medians in order to calculate the corresponding point estimate and confidence interval. There are some assumptions for the Mann-Whitney test such as:

Data are independent random samples from two populations.
Random samples have the same shape and the responses are ordinal.
Data is supposed to have a random distribution.

The results are obtained from Mann-Whitney test with confidence level 0.95. Hypothesis test is established according to the medians of the results to see if the algorithms are equal or not. Our data is assumed to be normally distributed with confidence interval 0.95, so in order to make comment about the results, we should consider the p value. If it is smaller than 0.05, we can state that the algorithms are not equal, otherwise they are equal.

In Mann-Whitney test, the parameters can be calculated as follows:

N: The number of observations

$Point\ Estimate$: The median of all possible pairwise differences between the two samples

W: (number of positive differences) + 0.5*(number of differences that equal 0) + 0.5*($n_1(n_1 + 1)$)

n_1 = number of observations in the first sample

$p\ value$: is based on the test statistic for W.

The test statistic Z, is a normal approximation using the mean and variance of W.

Mean of W = 0.5*($n_1(n_1$+n_2+ 1))

Variance of W = n_1*n_2*(n_1+n_2+1)/12

The p-value for Ha: Eta1 not= Eta2 is 2*(1 - CDF(Z)).

Adjusted ties: not to reject the null hypothesis at the 5% significance level.

As seen in the tables 1 to 11, all the p values are greater than 0.05, so all three algorithms are assumed to be equal. However, in the tables, average value of 10 instances is given for each job groups (100 jobs, 200 jobs, 300 jobs, 400 jobs and 500 jobs). If a single group is analyzed i.e. 100 jobs with 10 instances, the test results will be meaningful, as shown below:

Table 12. Run Results for Each Algorithms with 100 jobs, 10 instances, and α=0,9

		$\alpha = 0,9$											
		VNS				**DABC_VNS**				**GA_VNS**			
Jobs	**Ins**	**Min**	**Max**	**Avg**	**Std**	**Min**	**Max**	**Avg**	**Std**	**Min**	**Max**	**Avg**	**Std**
100	1	245,00	258,00	250,00	6,08	248,00	371,00	289,00	52,21	255,00	501,00	330,00	103,85
	2	241,00	243,00	241,00	1,00	260,00	280,00	267,00	7,68	244,00	310,00	265,00	27,27
	3	240,00	245,00	241,00	2,24	248,00	302,00	265,00	22,69	251,00	322,00	269,00	30,07
	4	236,00	240,00	236,00	2,00	250,00	277,00	258,00	10,98	240,00	275,00	253,00	14,47
	5	240,00	240,00	240,00	0,00	237,00	263,00	244,00	11,54	262,00	273,00	265,00	5,00
	6	241,00	241,00	241,00	0,00	257,00	331,00	283,00	29,72	253,00	334,00	285,00	30,46
	7	250,00	255,00	251,00	2,24	253,00	296,00	266,00	17,74	264,00	315,00	289,00	21,32
	8	244,00	244,00	244,00	0,00	243,00	305,00	270,00	23,69	266,00	358,00	286,00	40,14
	9	243,00	251,00	245,00	3,46	257,00	310,00	271,00	22,33	252,00	304,00	273,00	28,12
	10	242,00	248,00	243,00	2,69	250,00	311,00	275,00	23,76	266,00	356,00	292,00	36,10

Mann-Whitney Test and CI($\boldsymbol{\alpha = 0,9}$):

DABC_VNS; GA_VNS

	N	Median
DABC_VNS	10	250,00
GA_VNS	10	254,00

Point estimate for ETA1-ETA2 is -5,00
95,5 Percent CI for ETA1-ETA2 is (-14,00;4,00)
W = 86,5
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,1736

Figure 22. Mann-Whitney Test Result for DABC_VNS and GA_VNS

DABC_VNS; VNS

	N	Median
DABC_VNS	10	250,00
VNS	10	241,50

Point estimate for ETA1-ETA2 is 8,00
95,5 Percent CI for ETA1-ETA2 is (3,00;14,00)
W = 139,5
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,0102
The test is significant at 0,0099 (adjusted for ties)

Figure 23. Mann-Whitney Test Result for DABC_VNS and VNS

VNS; GA_VNS

	N	Median
VNS	10	241,50
GA_VNS	10	254,00

Point estimate for ETA1-ETA2 is -12,50
95,5 Percent CI for ETA1-ETA2 is (-22,00;-6,00)
W = 65,5
Test of ETA1 = ETA2 vs ETA1 not = ETA2 is significant at 0,0032
The test is significant at 0,0031 (adjusted for ties)

Figure 24. Mann-Whitney Test Result for VNS and GA_VNS

As seen in the table and the results, VNS algorithm outperforms the GA_VNS and DABC_VNS, their p values are 0.0031 and 0.0099 respectively. However, still we can assume that GA_VNS and DABC_VNS algorithms are equal, because of having greater p value (0.1736) than 0.05.

CHAPTER 6

CONCLUSION

In this paper, we presented a DABC_VNS and GA_VNS to solve a problem from the real-life. We developed DBAC_VNS and GA_VNS algorithms to assign the filling production orders to machine groups, and then schedule them on each identical parallel machine groups. We also presented a unique solution representation inspired from general assignment problem. In addition, we developed a novel VNS local search to further improve the solution quality. We also devised benchmark instances to test the performance of the algorithms proposed. The computational results show that the VNS algorithm slightly outperforms the GA_VNS and DABC_VNS on set of benchmark problems we generated.

As a future work, we will apply these algorithms to real-life data from DYO painting company in order to develop a decision support system for them.

BIBLIOGRAPHY

Alfadari, L., Plateau, A., & Tolla, P. (2004). *A Path Relinking Algortihm for the Generalized Assignment Problem, Resende M.G.C., Sousa J.D. (Eds.), Metaheuristics: Computer Decision Making.* Boston: Kluwer Academic Publishers.

Anghinolfi, D., & Paolucci, M. (2007). Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. *Computers and Operations Research*, 34: 3471-3490.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6: 154-160.

Bilge, U., Kyrac, F., Kurtulan, F., & Pekgun, M. (2004). A tabu search algorithm for parallel machine total tardiness problem. *Computers and Operations Research*, 31: 397-414.

Cattrysse, D., Salomon, M., & Wassenhove, L. V. (1994). *A Set Partitioning heuristic for the Generalized Assignment Problem.* European Journal of Operational Research.

Chu, P. C., & Beasley, J. E. (1997). *A genetic algortihm for the generialized assignment problem.* Operations Research.

Dessouky, M. (1998). Scheduling identical jobs with unequal ready times on uniform parallel machines to minimize the maximum lateness. *Computers and Industrial Engineering*, 34(4): 793-806.

Diaz, J. A., & Fernandez, E. (2001). *A Tabu Search Heuristic for the Generalized Assignment Problem.* European Joural of Operational Research.

Dogramaci, A., & Surkis, J. (1979). Evaluation of a heuristic for scheduling independent jobs on parallel identical processors. *Management science*, 25: 1025-1041.

Du, J., & Leung, J. T. (1990). Minimizing total tardiness on one machine is NP-Hard. *Mathematics of Operations Research*, 17(2):416-429.

Elmagraby, S. E., & Park, S. H. (1974). Scheduling jobs pn a number of identical machines. *AIIE Transactions*, 6: 1-13.

Feltl, H., & Raidl, G. R. (2004). *An Improved Hybrid Genetic Algorithm for the Generalized Assignment Problem.* Nicosia, Cyprus: Proceedings of the 2004 ACM Symposium on Applied Computing.

Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, (11) 109-124.

Fisher, M. L.; Jaikumar, R.; Wassenhove, L.N. Van;. (1986). *A Multiplier Adjustment Method for the Generalized Assignment Problem.* Management Science.

Garey, M. R., & Johnson, D. S. (1979). *A Guide to the Theory of NP-Completeness.* Freeman, San Francisco: Computers and Intractability.

Graham, L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy, A. K. (1979). *Optimization and Approximation in Deterministic Sequencing and Scheduling.* Annals of Discrete Mathematics.

Haddadi, S., & Ouzia, H. (2001). *An Effective Lgrangian Heuristic for the Generalized Assignment Problem.* INFOR.

Ho, J. C., & Chang, Y. L. (1991). Heuristics for minimizing mean tardiness for m parallel machines. *Naval research logistics*, 38: 367-381.

Karaboga, D. (2005). An idea based on honey bee swarm for numarical optimization. *Technical Report*, TR06.

Karaboga, D. (2009). A new design method based on artificial bee colony algorithm for digital IIR filters. *Journal of The Franklin Institute*, 346: 328-348.

Karaboga, D.; Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Matheatics and Computation*.

Karaboga, D.; Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39: 459-471.

Karaboga, D.; Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, 8: 687-697.

Karabulut, K., & Tasgetiren, M. F. (2009). A discrete artificial bee colony algorithm for the travelling salesman problem with time windows. *IEEE Congress on Evolutionary Computation*, 1-7.

Koulamas, C. (1997). Decomposition and hybrid simulated annealing heuristics for the parallel machine total tardiness problem. *Naval Research Logistics*, 44: 109-125.

Koulamas, C. P. (1994). The total tardiness problem: Review and Extensions. *Operations Research*, 42: 1025-1041.

Laguna, M., Kelly, J. P., Gonzalez-Valerde, J. L., & Glover, F. (1995). *Tabu Search for the Multilevel Generealized Assignment Problem.* European Journal of Operational Research.

Lawler, E. (1977). A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1: 331-42.

Lenstra, J. K., Rinnooy, A. K., & Brucker, P. (1977). *Complexity of Machine Scheduling Problems.* Annals of Discrete Mathematics.

Lenstra, J. K.; Shmoys, D. B.; Tardos, E. (1990). *Approximation Algorithms for Schedulig Unrelated Parallel Machines.* Mathematical Programming.

Lourenço, H. R., & Serra, D. (2002). *Adaptive Saerch Heuristics for the Generalized Assignment Problem.* Mathware and Soft Computing.

Martello, S., & Toth, P. (1981). *An Algorithm for Generalized Assignment Problems.* North-Holland: J.P. Brans (Ed.), Operational Research.

McNaughton, R. (1959). Scheduling with deadlines and loss functions. *Management Science*, 1-12.

Mladenovic, N., & Hansen, P. (1997). Variable Neighborhood Search. *Computers and Operations Research*, 24: 1097-1100.

Nauss, R. M. (2003). *Solving the Generalized Assignment Problem: An Optimizing and Heuristic Approach.* Informs Journal of Computing.

Nowicki, E., & Smutnicki, C. (1998). The flow shop with parallel machines: A tabu search approach. *EJOR*, 106: 226-253.

Osman, I. H. (1995). *Heuristics for the Generalized Assigment Problem: simulated annealing and tabu search approaches.* OR Spektrum.

Pan, Q. K., Tasgetiren, M. F., & Liang, Y. C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem with makespan and total flowtime criteria. *Computers and Operations Research*, 35(9), 2807-2839.

Park, M. W., & Kim, Y. D. (1997). Search heuristics for a parallel machine scheduling problem with ready times and due dates. *Computers and Industrial Engineering*, 33(3-4): 793-796.

Pessoa, A., Uchoa, E., Aragao, M. P., & Rodrigues, R. (2008). Algorithms over arc-time indexed formulations for single and parallel machine scheduling problems. *Technical Support RPEP*, vol.8 no.8.

Racer, M., & Amini, M. M. (1994). *A Robust Heuristic for the Generalized Assignment Problem.* Annals of Operations Research.

Randall, M. (2004). *Heuristics for Ant Colony Optimization Using the Generalized Assignment Problem.* Portland, Oregon, USA: Proceedings of IEEE Congress on Evolutionary Computation.

Root, J. (1965). Scheduling with deadlines and loss function on k paralel machines. *Management Science*, 11: 460-75.

Ross, G. T., & Soland, P. M. (1975). *A Branch and Bound Based Algorithm for the Generalized Assignment Problem.* Mathematical Programming.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of flowshop heuristics. *Eur. J. Oper. Res.*, 165(2): 479–494.

Sahni, S., & Gonzalez, T. (1976). *P-complete Approximation Problems.* Journal of the ACM.

Savelsbergh, M. (1997). *A Branch and Price Algorithm for the Generalized Assignment Problem.* Operations Research.

Tasgetiren, M. F., Liang, Y. C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research*, 177, 1930-1947.

Wilkerson, L. J., & Irwin, J. D. (1971). An improved algorithm for scheduling independent tasks. *AIIE Transactions*, 3: 239-245.

Yagiura, M., Yamaguchi, T., & Ibaraki, T. (1998). A variable depth search algorithm with branching search for the generalized assignment problem. *Optimization Methods and Software*, 10, 419–441.

Yagiura, M.; Ibaraki, T.; Glover, F. (2004). An ejection chain approach for the generalized assignment problem. *INFORMS Journal of Computing*, 16 (2), 133–151.

Yagiura, M.; Ibaraki, T.; Glover, F. (2006). 2006. A path relinking approach with ejection chains for the generalized assignment problem. *European Journal of Operational Research*, 169, 548–569.

Yagiura, M.; Ibaraki, T.; Glover, F. (2006). *A Path Relinking Approach with Ejection Chains for the Generalized Assignment Problem.* European Journal of Operational Research.

Yagiura, M.; Yamaguchi, T.; Ibaraki, T. (1998). *A Variable Depth Search Algorithm with Branching Search for the Generalized Assignment Problem.* Optimization Methods and Software.

Yagiura, M.; Yamaguchi, T.; Ibaraki, T. (1999). *A Variable Depth Search Algorithm for the Generalized Assigment Problem in : Vob S., Martello S., Osman I.H., Roucairol C.i (Eds.), Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization.* Boston: Kluwer Academic Publishers.

Yaguira, M., Ibaraki, T., & Glover, F. (2004). *An Ejection Chain Approach for the Generalized Assignment Problem.* Informs Journal of Computing.