**DOKUZ EYLÜL UNIVERSITY**

**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

# A SOLUTION APPROACH FOR ALTERNATIVE SUBGRAPHS ASSEMBLY LINE BALANCING PROBLEM

**by**

**Ümmühan PALAMUT**

**October, 2021**

**İZMİR**

# A SOLUTION APPROACH FOR ALTERNATIVE SUBGRAPHS ASSEMBLY LINE BALANCING PROBLEM

**A Thesis Submitted to the**
**Graduate School of Natural and Applied Sciences of Dokuz Eylül University**
**In Partial Fulfillment of the Requirements for the Degree of Master of Science in Industrial Engineering, Industrial Engineering Program**

**by**

**Ümmühan PALAMUT**

**October, 2021**

**İZMİR**

**M.Sc THESIS EXAMINATION RESULT FORM**

We have read the thesis entitled **"A SOLUTION APPROACH FOR ALTERNATIVE SUBGRAPHS ASSEMBLY LINE BALANCING PROBLEM"** completed by **ÜMMÜHAN PALAMUT** under supervision of **ASSOC. PROF. DR. ŞENER AKPINAR** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Şener AKPINAR

Supervisor

Assist. Prof. Dr. Özgür YALÇINKAYA

(Jury Member)

Assist. Prof. Dr. Mustafa AVCI

(Jury Member)

Prof. Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

**ACKNOWLEDGMENTS**

Firstly, I would like to thank my advisor, Assoc. Prof. Dr. Şener AKPINAR thesis, for his support and advice, for patiently answering all my questions and for his guidance in the formation of my M. Sc. thesis.

Also, I would like to thank my family for their confidence, encouragement and support throughout in my whole life.

Ümmühan PALAMUT

# A SOLUTION APPROACH FOR ALTERNATIVE SUBGRAPHS ASSEMBLY LINE BALANCING PROBLEM

## ABSTRACT

Assembly line balancing problem (ALBP) is the process of assigning a set of tasks to a group of workstations, considering the precedence relations between the assembly tasks. Precedence relationships between tasks are shown with help of a graph diagram. In more complex versions of this problem, tasks may have alternative precedence relationships. This situation has led to the emergence of the Alternative Subgraph Assembly Line Balancing Problems (ASALBP). In this thesis, we worked on the Alternative Subgraph Assembly Line Balancing Problem of type 1 (ASALBP-1). ASALBP-1 aims to assign the tasks to a minimum number of workstations under precedence relations and cycle time constraints after the alternative subgraph is selected for a line balancing problem with different mounting alternatives. In this study, ASALBP-1 is solved with firefly, bat and proposed hybrid firefly-bat metaheuristic algorithms. At the beginning of proposed hybrid firefly-bat algorithm, the alternative subgraph selections of the problem are made, and then the initial solutions are generated by using the ranked positional weight method and randomly. After the initial solutions are formed, the firefly algorithm is used to generate new solutions initially. Later, the hybrid algorithm is terminated by adding the operators of the bat algorithm to the solution. Benchmark problems are solved with firefly and bat algorithms in addition to the hybrid algorithm. These three algorithms are used to solve twelve problem sets with different sizes, cycle times and subgraphs. Performance evaluations are made by comparing the hybrid algorithm against some heuristics taken from the literature for firefly, bat, and proposed hybrid algorithms. Comparative conclusions of algorithms give that the proposed hybrid firefly - bat algorithm is capable of producing promising results for ASALBP-1.

**Keywords:** Alternative subgraphs assembly line balancing problem, hybrid metaheuristic, firefly algorithm, bat algorithm

# ALTERNATİF ALTGRAFİK MONTAJ HATTI DENGELEME PROBLEMİ İÇİN BİR ÇÖZÜM YAKLAŞIMI

## ÖZ

Montaj hattı dengeleme problemi, bir dizi görevin montaj görevleri arasındaki öncelik ilişkilerini göz önünde bulundurarak bir grup iş istasyonuna atanması işlemidir. Görevler arasındaki öncelik durumları bir grafik diyagramı yardımıyla gösterilir. Bu problemin daha karmaşık versiyonlarında, görevler alternatif öncelik ilişkilerine sahip olabilirler. Bu durum, alternatif alt grafik montaj hattı dengeleme problemlerinin ortaya çıkmasına neden olmuştur. Biz bu tezde birinci tip alternatif alt grafik montaj hattı dengeleme problemi üzerinde çalıştık. ASALBP-1, farklı montaj alternatiflerine sahip bir hat dengeleme problemi için alternatif alt grafik seçildikten sonra, öncelik ilişkileri ve çevrim süresi kısıtlamaları altında minimum sayıda iş istasyonuna atamayı amaçlamaktadır. Bu çalışmada, ASALBP-1 ateşböceği, yarasa ve önerilen hibrit ateşböceği-yarasa metasezgisel algoritmaları ile çözülmüştür. Önerilen hibrit ateşböceği - yarasa algoritmasının başlangıcında, problemin alternatif alt grafik seçimleri yapılmış ve daha sonra sıralı konumsal ağırlık yöntemi kullanılarak ve rassal olarak ilk çözümler üretilmiştir. İlk çözümler oluşturulduktan sonra, yeni çözümler üretmek için ilk olarak ateşböceği algoritması kullanılır. Ardından yarasa algoritmasının operatörleri çözüme eklenerek hibrit algoritma sonlandırılır. Kıyaslama problemleri hibrit algoritmaya ek olarak ateşböceği ve yarasa algoritmalarıyla çözülmüştür. Bu üç algoritma farklı büyüklüğe, çevrim zamanına ve alt grafiğe sahip on iki problem seti kullanılarak çözülmüştür. Ateşböceği, yarasa ve önerilen hibrit algoritmalar için literatürden alınan bazı sezgisel yöntemler ile performans karşılaştırması yapılmıştır. Algoritmaların karşılaştırmalı sonuçları, önerilen hibrit ateşböceği yarasa algoritmasının ASALBP-1 için umut verici sonuçlar üretebileceğini göstermektedir.

**Anahtar kelimeler:** Alternatif montaj hattı dengeleme problemi, hibrit metasezgisel, ateşböceği algoritması, yarasa algoritması

**CONTENTS**

**Page**

## LIST OF FIGURES

**Page**

## LIST OF TABLES

**Page**

# CHAPTER ONE
# INTRODUCTION

## 1.1 The Importance of the Problem

In the developing industrial world, assembly lines are one of the most important parts of the production systems. Assembly lines play an important role in the production of larger quantities of products and efficient use of scarce resources. Assembly lines are flow-line production systems which are of a combination of workstations with a material handling system. The assembly line balancing problem is one of the main topics in the literature on optimization of the assembly lines. Assembly line balancing problem (ALBP) is known as the decision problem of optimally balancing the assembly work among the workstations with respect to some objectives (Scholl, 1999). In this problem type, the goal function is to either reduce the number of workstations or cycle time.

An early one of the best-known classifications was prepared by Baybars (1986) on assembly line balancing problems. According to Baybars (1986), Assembly Line Balancing Problems consist of two basic types of problems, the first is Simple assembly line balancing problems (SALBP) and the second is General Assembly Line Balancing Problems (GALBP).The simple case (SALBP) is an assembly line where only one standard product is produced on a serial assembly line. If the problems have more complexity and limitations, they are considered as GALBP. Scholl & Becker (2006) presented a survey on the problems and methods in generalized assembly line balancing problems (GALBP) literature. Battaia & Dolgui (2013) reviewed the assembly line balancing problem and their solution approaches.

While assembly line balancing problems are considered simpler in general, real-life problems can have a more complicated structure. Therefore, it is thought that the ALBP needs to be discussed and researched in more detail. One of the studied complicated points is the alternative precedence graphs in assembly line balancing

problems. Normally, in an assembly line balancing problem, the precedence graph of all tasks is one, while in real-life problems some tasks may have optional precedence relations. Limited some studies in the literature like Pinto, Dannenbring &Khumawala (1983) and Bukchın & Tzur (2000), conducted research on processing alternatives for limited equipment selection. Also, Das & Nagendra (1997) for the production of toy and Senin, Groppetti & Wallace (2000) for the production of commercial hand-held drill are worked with alternative assembly process (Capacho & Pastor, 2005).

Firstly, about this topic, a new kind of GALBP named Alternative Subgraph Assembly Line Balancing Problem (ASALBP) has been introduced by Capacho & Pastor (2005). In ASALBP, each subassembly option has a specific task sequence with different precedence relations. This problem is mostly related to the assembly line balancing problem faced by suppliers with a large product range and product number, which enables them to offer more options to their customers in direct proportion to increasing customer needs. The distinctive aspect of the alternative sub-graph assembly line balancing problem is that it has alternative sub-graph precedence relations, not definite precedence relations. In the alternative subgraph assembly line balancing problem, mounting alternatives for different parts of an assembly or manufacturing process are considered. Each process is represented by a sub-graph that specifies the tasks, task times and task precedence relations required to produce a product.

The ASALBP is divided into two groups for its purposes. If the target is to minimize the number of stations for a given cycle time, problem is defined as ASALBP-1. If it is aimed to minimize the cycle time for a given number of stations, the problem is defined as ASALBP-2 (Capacho & Pastor, 2005).

In this study, the ASALBP-1 is studied. ASALBP-1 first chooses alternative precedence subgraph and later assigns the tasks selected in this subgraph to the most appropriate station number within the given cycle time.

As the number of precedence sub-graph alternatives and tasks numbers increases and therefore the solution space expands, ASALBP is evaluated in the NP-hard problem class (Capacho & Pastor, 2005). In previous studies, ASALBP-1 was solved with mathematical programming which is the exact solution method and heuristic and metaheuristic methods which are approximate solution methods. As the problem data grows, it takes a long time to solve the problem to optimality with mathematical programming model. Although ASALBP-1 has been solved by many heuristic models, metaheuristics have not been studied extensively on this subject. For these reasons, scientific studies suggested to solve the ASALBP with different metaheuristic methods for future research.

## 1.2 Research Methodology

In this paper, we propose metaheuristics of firefly algorithm (FA), bat algorithm (BA) and hybridize FA with BA algorithm for ASALBP. Firefly algorithm has been developed as a swarm-based metaheuristic algorithm by Yang (2008). Although firefly algorithm was first developed for continuous optimization problems, due to the good results obtained in later studies, some changes were made in the algorithm and started to be used in solving discrete optimization problems (Tilahun & Ngnotchouye, 2017). Bat algorithm is another swarm-based metaheuristic algorithm developed by Xin-She Yang (2010b). Bat algorithm was also designed for continuous optimization problems, and then it is applied by making changes in its structure according to the discrete optimization problems (Kongkaew, 2017).

In previous studies, both ALBP and many optimization problems have been solved by hybrid metaheuristic methods (Rahmani & MirHassani, 2014), (Zhou et.al, 2013). Hybrid metaheuristics can provide a more efficient behaviour and a higher flexibility when dealing with real-world and large-scale problems (Blum & Roli, 2008). Based on literature, a hybrid algorithm is proposed a combination of firefly and bat algorithms for solving ASALBP-1. Finally, we solved ASALBP-1 with these

three algorithms and compared the results against the previously developed solution procedures in the related literature.

### 1.3 Outline of the Thesis

The remainder of this study is organized as follows. The problem definition of the ASALBP-1 and a literature review on the ASALBP-1 are given in Chapter 2.

In Chapter 3, classical firefly algorithm, classical bat algorithm and their components are explained. Also, literature reviews are given about the firefly and bat algorithms.

Chapter 4 explains the proposed hybrid firefly-bat algorithm and executes a set of computational studies to evaluate the performances of firefly, bat and hybrid algorithms.

Lastly, all results are evaluated and resulting contributions of the thesis are presented in Chapter 5.

# CHAPTER TWO
# PROBLEM DEFINITION AND LITERATURE SURVEY

## 2.1 Introduction

This chapter consists of two subtitles. Firstly, the alternative subgraph assembly line balancing problem is defined and mentioned in its content. Also, an illustrative subgraph representation is exemplified, and the solution steps of ASALBP-1 are explained. In the second subtitle, the literature review on the alternative subgraph assembly line balancing problem is presented.

## 2.2 Definition of ASALBP

The alternative subgraph assembly line balancing problem (ASALBP) is a general assembly line balancing problem. ASALBP is more realistic for solution of real-life problems. The alternative subgraph assembly line balancing problem is that the product has different installation alternatives during the assembly process. The alternative subgraph assembly line balancing problem consists of two sub problems. The first one is the decision problem and one of the installation alternatives needs to be selected. The second one is the line balancing problem, and it is intended to assign tasks to the minimum number of workstations. In this study, only ASALBP-1 targeting workstation minimization was studied for a given cycle time.

The problem may come from multiple assembly sections and these assembly sections may have different sub-graphics. Sub-assemblies owned by ASALBP, and sub-graphs of these sub-assemblies are shown with S-graph. Task priorities and task numbers may change in sub graphs, and this change may result in different task times. All these changes cause a change in the objective function. The main goal of the problem is to find the minimum alternatives of precedence relations with different objective function results and to arrange the assembly tasks accordingly.

### *2.2.1 An Illustrative Example of ASALBP*

This part includes a small example problem solution and representation for ASALBP. This problem has 13 tasks and 2 sub-assembly lines. Figure 2.1 shows S-graph with 4 alternative precedence relations. This example is an assembly line balancing problem that consists of 2 sub-assembly lines, each with 2 alternative subgraphs.

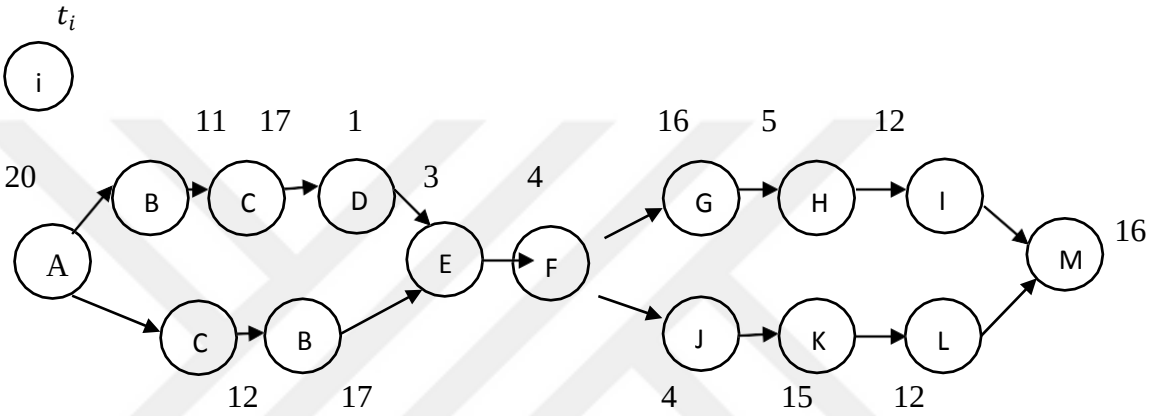


Figure 2.1 Precedence S-Graphs

Table 2.1 shows the precedence relations and task times of the assembly line with different sub graph options, and a complete assembly line that will be generated with different combinations of subgraphs. In the first subassembly, there are 2 alternative precedence sub-graphs with the same and different tasks: B, C, D, but their orders and different times. In the second subassembly, there are 2 alternative precedence sub-graphs with different tasks: S3 with tasks G, H, and I; S2 with tasks J, K and L. A, E, F and M fixed tasks. The goal of the problem is to assign tasks to the minimum number of stations and other purpose is to choose assembly subgraphs with the minimum number of stations for ASALBP-1. The cycle time for each machine in the problem was 28. In Figure 2.1, while $i$ index shows the tasks number in parameter,$t_i$, $t_i$ shows the proccesing time of the *ith* task.

Table 2.1 Display of the priorities and times of the assembly alternatives

| Tasks | S1 - S3 | | S1 - S4 | | S2 - S3 | | S2 - S4 | |
|---|---|---|---|---|---|---|---|---|
| | Process time | Predecessors | Process time | Predecessors | Process time | Predecessors | Process time | Predecessors |
| A | 20 | - | 20 | - | 20 | - | 20 | - |
| B | 11 | A | 11 | A | 17 | C | 17 | C |
| C | 17 | B | 17 | B | 12 | A | 12 | A |
| D | 1 | C | 1 | C | - | - | - | - |
| E | 3 | D | 3 | D | 3 | B | 3 | B |
| F | 4 | E | 4 | E | 4 | E | 4 | E |
| G | 16 | F | - | - | 16 | F | - | - |
| H | 5 | G | - | - | 5 | G | - | - |
| I | 12 | H | - | - | 12 | H | - | - |
| J | - | - | 4 | F | - | - | 4 | F |
| K | - | - | 15 | J | - | - | 15 | J |
| L | - | - | 12 | K | - | - | 12 | K |
| M | 16 | I | 16 | L | 16 | I | 16 | L |

In Table 2.2, it is shown that the tasks of the assembly subgraphs of the example problem are assigned to which workstation, paying attention to the precedence relations and cycle time constraints. Table 2.2 shows that the best solution for the solution of this example problem, namely the minimum number of workstations, is the optimum assembly process to be performed with the 1st and 4th sub-graphics. The assembly line can be balanced by assigning to a minimum of 4 workstations. In this example, line balancing was applied to all alternative assemblies by paying attention to the precedence constraints according to the number of task sequences

first, and as a result, the mounting alternative that can be operated on the least number of workstations was determined as the optimum result.

Table 2.2 Assignment results in the example problem

| Assembly alternatives | Station load (time) | | | | | Number of stations |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| S1 - S3 | A (20) | B, C (28) | D, E, F, G (24) | H, I (17) | M (16) | 5 |
| S1 - S4 | A (20) | B, C (28) | D, E, F, J, K (27) | L, M (28) | - | 4 |
| S2 - S3 | A (20) | C (12) | B, E, F (24) | G, H (21) | I, M (28) | 5 |
| S2 - S4 | A (20) | C (12) | B, E, F (24) | J, K (19) | L, M (28) | 5 |

### *2.2.2 Mathematical Formulization of ASALBP-1*

In this section, the mathematical programming model developed by Capacho & Pastor (2005) for ASALBP-1 is given. The problem formulized as a binary linear programming model. The notations with their definitions belonging to this mathematical formulation are given in Table 2.3.

Table 2.3 Indices, parameters and decision variables of ASALBP's mathematical model

| | | |
|---|---|---|
| Indices | i | Number of tasks |
| | j | Workstations |
| | r | Routes |
| Parameters | n | Number of tasks ($i$ = 1, …, $n$) |
| | $m_{max}$ | Upper bound on the number of workstations ($j$ = 1, …, $m_{max}$) |
| | $m_{min}$ | Lower bound on the number of workstations |
| | $nr$ | Number of alternative routes ($r$ = 1, …, $nr$) |
| | $t_{ir}$ | Duration of task $i$ when processed through route $r$ ($i$ = 1, …, $n$; $r$ = 1, …, $nr$); in some cases, this value is independent of route $r$ ($t_i$) |
| | $C_{max}$ | Upper bound on the cycle time |
| | $PD_{ir}$ | Set of the immediate predecessors of task $i$, if task $i$ is processed through route $r$ ($i$ = 1, …, $n$; $r$ = 1, …, $nr$) |
| | $E_{ir}$, $L_{ir}$ | Earliest and latest station respectively that task $i$ can be assigned to, if task $i$ is processed through route $r$ ($i$ = 1, …, $n$; $r$ = 1, …, $nr$) |
| | $T_{jr}$ | Set of tasks potentially assignable to workstation $j$, {i \|j ∈ [ $E_{ir}$, $L_{ir}$]}, if the tasks are processed through route $r$ ($j$ = 1, …, $m_{max}$; $r$ = 1, …, $nr$) |
| Decision variables | $x_{ijr}$ | 1 if task $i$ is assigned to workstation $j$ and processed through route $r$ ($\forall i, \forall r, \forall j \in [E_{ir}, L_{ir}]$) |
| | $y_j$ | 1 if there is any task assigned to workstation $j$ ($j$= $m_{min}$ +1, ..., $m_{max}$) |

$$\text{Min Z} = \sum_{j=m_{min}+1}^{m_{max}} j.\, y \tag{2.1}$$

$$\sum_{r=1}^{nr} \sum_{j=E_{ir}}^{L_{ir}} x_{ijr} = 1 \qquad \forall\, i \tag{2.2}$$

$$\sum_{r=1}^{nr} \sum_{\substack{\forall i \\ \in T_{jr}}} t_{ir}\, x_{ijr} \leq C_{max} \qquad (j = 1, \ldots, m_{min}) \tag{2.3}$$

$$\sum_{r=1}^{nr} \sum_{\substack{\forall i \\ \in T_{jr}}} t_{ir}\, x_{ijr} \leq C_{max}\, .y_j \qquad (j = m_{min} + 1, \ldots, m_{max}) \tag{2.4}$$

$$\sum_{j=E_{kr}}^{L_{kr}} j.\, x_{kjr} \leq \sum_{j=E_{ir}}^{L_{ir}} j.\, x_{ijr} \qquad \forall \text{r}, \forall \text{i}, \forall \text{k} \in PD_{ir} \tag{2.5}$$

$$\sum_{j=E_{1r}}^{L_{1r}} x_{1jr} \leq \sum_{j=E_{ir}}^{L_{ir}} x_{ijr} \qquad \forall \text{r; } i\text{=2, ..., n} \tag{2.6}$$

$$x_{ijr} \in \{0, 1\} \qquad \forall i, \forall \text{r}, \forall \text{j} \in [E_{ir}, L_{ir}] \tag{2.7}$$

$$y_j \in \{0,1\} \qquad (j = m_{min} + 1, \ldots, m_{max}) \tag{2.8}$$

The objective function (2.1) aims to assign tasks to minimum workstations. Constraint (2.2) guarantees that every task is assigned to one and only one workstation. Constraints (2.3) and (2.4) check whether the sum of the processing times of the tasks assigned to a workstation exceeds the given cycle time. Constraint (2.5) is to provide precedence relations for the tasks to be assigned to the workstation. Route uniqueness constraint (2.6) ensures that all the tasks are assigned to the same route. Lastly, constraint (2.7) checks whether a job on any route is assigned to a station, and constraint (2.8) checks whether a job is assigned to a workstation.

## 2.3 Literature Review for ASALBP

There are very few referenced studies on ASALBP according to Capacho and Pastor (2005). Therefore, in this section, papers on alternative assembly process, before the ASALBP was defined, and papers on the ASALBP were examined from 1986 to the present.

Pinto, Dannenbring & Khumawala (1983) discussed the problem of manufacturing design, which has more than one processing alternatives. Each alternative is represented on the same precedence graph and selected tasks are assigned. But precedence relations are not change with alternatives in this problem. In this problem, cost minimization is aimed with alternative manufacturing design. Processing alternative assembly line balancing problem formulized and solved through an integer programming model.

Mello & Sanderson (1990) worked with the AND/OR graph to show the assembly variants of a product in this study. AND/OR graph represented robotic assembly plans and feasible assembly sequences. It provided best assembly plan for the selection of disassembly or repair plan. Using this representation, it is compared to the fixed sequence and precedence graph representations. A better solution was obtained with the AND/OR graph. Top-down and bottom-up search algorithms were used to solve the problem. The scheduling efficiency using this representation was compared with fixed sequence and precedence graph representations. The AND/OR graph consistently reduced the average number of operations.

Das & Nagendra (1997) defined routing flexibility as the ability of manufacturing a product by several alternative routes in the same facility. In this study, feasible route solutions were created for the products. In order to reduce the economic cost, it was aimed to choose the best route for products with several production routes. Mixed integer programming method was used to solve the problem.

Park, Park & Kim (1997) tried to solve an ALB problem that arose in a company that manufactures power washers. In this problem, precedence relationships between tasks were difficult to show with a precedence graph. Firstly, they proposed to solve the problem by separating it into two sub-problems. The first sub-problem allowed floating tasks to be reassigned while keeping fixed tasks in place. The second sub problem aimed at optimizing the assignment of the fixed tasks while preserving the current relative sequence of the float tasks. Heuristic solution methods and network theory are proposed to solve the problem.

Bukchin & Tzur (2000) studied the process of optimally designing an assembly line with different assembly equipment options. It was aimed to select equipment for workstations, taking into account task assignments and cost minimization. But in this study, tasks did not have different times. By using the branch and bound method, large-scale problems are solved with heuristic solution methods.

Firstly, Capocho & Pastor (2005) presented a new kind of GALBP, the alternative subgraph assembly line balancing problem (ASALBP). They gave the character traits of ASALBP and described the problem with small examples and presented the integer linear mathematical model to solve the problem.

Capocho & Pastor (2006a) discussed in this paper the solution of a more complex version of ASALBP with assembly sub-processes with different tasks. This new version of ASALBP has been reformulated and solved with the mathematical solution method.In small problems descripted by Model-1(M1) for each assembly precedence graph (i.e., global route) formed by combining the alternative subgraphs of available subassembly containing the same tasks in the S-graph. In M2, namely the extended ASALBP definition, alternative assembly process may contain either the same or different tasks. This type of the ASALBP was thought to be more suitable for real life problems. The new extended version mathematical model was given for second model (M2). As a result, M2 gave better results than M1 in medium and big scale test problems.

Capacho & Pastor (2006b) solved the ASALBP-1 with some heuristic approaches. In previous works, they developed an integer linear mathematical programming model to solve this problem optimally. Until this study, it has been seen that only small-scale problems can be solved in reasonable time with ASALP with exact solution methods. Therefore, heuristic methods are proposed in this study to solve large and medium-sized problems and improve their solutions.In this reason, three single-pass and two multi-pass heuristic methods have been proposed to solve ASALBP-1.

Capacho, Guschinskaya, Dolgui & Pastor (2006a) solved ASALBP-1 using 14 different heuristic methods. Subgraph selections were made randomly, and heuristic methods were used in ordering the tasks. 10 medium and large-scale benchmark problems were solved, and their results were compared.

Capacho, Guschinskaya, Dolgui & Pastor (2006b) used the approximation methods to solve ASALBP-1. They evaluated a set of five heuristic methods and four decision rules for tasks used to solve the novel Alternative Subgraphs Assembly Line Balancing Problem (ASALBP-1). The solution methods were applied to nine medium and large benchmark problems. In this research report, the results of all the solved data sets were given in detail.

Capacho et al. (2009) evaluated constructive heuristics methods for solution ASALBP-1. For selection assembly subgraphs three priority rules and random choice were used, and for selection of tasks thirteen heuristic methods and random choice were used. A total of 39 single pass heuristic and 17 multi pass heuristic methods were proposed in the study.

Boysen, Fliedner & Scholl (2007) classified assembly line balancing problems in their paper; firstly, ASALBP notation was characterized as [$pa^{subgraph}$||m]. This demonstration shows us that the priority graph has process options and that the number of workstations changes according to these alternatives.

Salum & Supciller (2008) proposed rule-based representation for alternative assembly plans. Researchers said that the classical precedence diagram is insufficient for this type of problem. They wanted to show that the assembly constraints in the problem are better expressed with if then rule based representation. The paper also showed that the rule-based model can easily be employed by heuristics used for this type of line balancing problems.

Schooll, Boysen & Fliedner (2009) gave an alternative representation of the ASLBP in their work. OR- node representation was used to show alternative subgraphs more specifically in the precedence diagram and to improve the mathematical model. They suggested a new mixed integer programming model and used a solution approach based on a new type called SALOME branch and bound procedure. SALOME has different branching like local lower bound method.

Koc, Sabuncuoglu & Erel (2009) studied on disassembly line balancing problem which has different dismantling alternatives. These problems were shown with AND/OR graph. The AND/OR graph was used to determine independent tasks in subassemblies. The objective function was minimization of workstation for a given cycle time with different disassembly processes. They used two exact formulations as integer programing (IP) and dynamic programming (DP).

Capacho & Pastor (2011) solved the Alternative Subgraphs Assembly Line Balancing Problem (ASALBP) with a Greedy Randomized Adaptive Search Procedure (GRASP). The proposed GRASP metaheuristic algorithm consisted of the first phase of selecting subgraph, initial assignment of tasks and the second phase of local search. Swap and insertion search algorithms were used in local search.

Topaloglu, Salum & Supciller (2012) proposed a rule-based (if-then rules) model for solving alternative precedence graphs constraint about assembly line balancing problem. Rule-based model representation, integer programming and constraint programming were used to solve the problem. Rule-based and graph-based representations were compared to see which priority representation contributed better

to the problem. In rule-based model, the performance of constraint programming and integer programming approaches were compared for small and medium scale problems.

The literature review is summarized in Table 2.4. The purpose, the precedence relations diagram representation and the solution methods they used of the examined papers were explained.

Table 2.4 Solution approaches of the studies in the literature review for ASALBP

| **Authors(s)** | **Purpose of the Study** | **Graph representation scheme** | **Solution approach** |
|---|---|---|---|
| Pinto et al.(1983) | Manufacturing design Cost minimization | Classic precedence graph | 0-1integer programming Brach and bound procedure |
| Sanderson and Mello et al.(1990) | Disassembly planning | AND/OR graph representation | Top-down search and bottom-up search |
| Das & Nagendra (1997) | Route selection problem | Classic precedence graph | Mixed integer programming |
| Park et al.(1997) | Minimization workstation numbers for ALBP | Classic precedence graph | Heuristic algorithms |
| Bukchın & Tzur (2000) | Equipment selection Cost minimization | Classic precedence graph | Integer programming, Branch and Bound algorithm |
| Capacho and Pastor (2005) | Workstation minimization for ASALBP | S-graph representation for ASALBP-1 | Mathematical programming |
| Capacho & Pastor (2006) | Minimization workstation numbers for ASALBP | S-graph precedence representation | Integer linear mathematical model |
| Capacho et al. (2006) | Minimization workstation numbers for ASALBP | S-graph precedence representation | Single-pass heuristic methods |
| Capacho & Pastor (2006) | Minimization workstation numbers for ASALBP | S-graph precedence representation | Mıxed integer linear programming |
| Capacho et al. (2006) | Minimization workstation numbers for ASALBP | S-graph precedence representation | Single pass and multi pass heuristics |

Table 2.4 Continues

| | | | |
|---|---|---|---|
| Capacho et al. (2006) | Minimization workstation numbers for ASALBP | S-graph precedence representation | Heuristics |
| Capacho et al.(2009) | Minimization workstation numbers for ASALBP | S-graph precedence representation | Single pass and multi pass heuristics |
| Boysen et. al. (2007) | Classification and characterization | - | - |
| Salum and Supciller (2008) | Rule based representation for ALBP | If then rule based Graph representation | - |
| Scholl et. al.(2009) | Minimization workstation numbers for ASALBP | OR-node precedence representation | Mixed integer linear programming |
| Koc (2009) | Disassembly planning | AND/OR graph precedence representation | Integer programming Dynamic programming |
| Capacho and pastor (2011) | Minimization workstation numbers for ASALBP | S -graph precedence representation | Metaheuristics GRASP |
| Topaloglu et al. (2012) | Minimization workstation numbers for ALBP | If then rule based Precedence representation | Integer programming Constraint programming |

The relevant literature shows us that there are some missing parts of the studies for the alternative subgraph assembly line balancing problems. Among the assembly line balancing problems, alternative assembly line balancing problems are better suited to real life problems. It is more difficult to solve large and medium-sized problems in the relevant industries and the solution of these problems is sought more in this context. Mathematical solution methods have been insufficient in solving medium and large-scale problems. For heuristic methods, results can be improved since they are operated in a short time. Studies in the literature have suggested metaheuristic solution methods for future studies. In line with these gaps in the studies, we proposed a hybrid solution approach with metaheuristic solution methods of firefly and bat algorithms, and we solved the alternative subgraph assembly line balancing problem type 1 with pure firefly, pure bat and proposed hybrid firefly-bat algorithms.

# CHAPTER THREE

# AN OVERVIEW ON BAT AND FIREFLY ALGORITHMS

## 3.1 Introduction

This chapter basically describes two metaheuristic algorithms that constitute the proposed hybrid metaheuristic algorithm for the solution of the type-1 alternative sub-graph assembly line balancing problem. These metaheuristics are firefly and bat algorithms. Firstly, classical firefly algorithm (FA) is explained and a literature review about the firefly algorithm is given. Later, bat algorithm (BA) is explained in detail and a literature review on the bat algorithm is also given.

### *3.2.1 Firefly Algorithm*

FA is a population and swarm-based metaheuristic algorithm which was developed by Yang (2008). The firefly algorithm is a nature inspired algorithm, based on the principle that fireflies produce light with special structures in their bodies in order to catch their prey or draw their pairs. Firstly, it was suggested for continuous problems. Firefly algorithm was determined that it gave better results for optimization problems and especially for NP-hard problems than particle swarm optimization and genetic algorithms by first testing it on test problems (Yang, 2010a). Also, it has been tried to be adapted to discrete problems due to its success in continuous problem solving (Fister, Fister Jr, Yang & Brest, 2013). Firefly algorithm was proposed for the solution of many kind of problems such as assembly sequence planning problem (Li, Zhang, Zeng, Zhou & Liu, 2015), scheduling problem, (Marichelvam, Prahabaran & Yang, 2014) vehicle routing problem (Osaba et al., 2017; Osaba, Carbelloda, Yang & Diaz, 2016), traveling salesman problem (Zhou, Ding & Qiang, 2014), capacitated facility location problem (Rahmani & MirHassani, 2014), and knapsack problem (Baykasoglu & Özsoydan, 2014).

Fister et al. (2013) and Yang (2014) mentioned some advantages of firefly algorithm and the following points where it is superior to other algorithms.

- The fact that the algorithm is population-based enables better search space and the inclusion of more candidate solution alternatives, especially in large data problems.

- Using parameters that give better results with parameter adjustment to speed up the achievement of feasible solutions.

- It allows the firefly algorithm to focus on the best local options in the population at each iteration, increasing the probability that the global solution will eventually approach to the best solution.

As with every different algorithm, the firefly algorithm has some unique features. First, the firefly algorithm is a swarm-based algorithm with a certain number of individuals. The light intensity and location of the individuals in the swarm effect to find their prey, food, and interactions with each other. The movement of fireflies is inversely proportional to the square of the distance to their prey, to their food and to each other. As a result, the higher the light intensity and attractiveness of fireflies, the lower the light intensity of the individual moves towards the higher light intensity individual.

Also, Yang (2008) said that the metaheuristic firefly algorithm is based on the following three basic properties.

- Firstly, all fireflies are of the same sex, in which case all individuals are affected in the same way.

- The second is the intensity of the light they emit. Attractiveness is proportional with firefly light intensity. Attractiveness criterion that allows them to be influenced by each other. Whichever firefly is brighter and is closer to the

distance in range, the other moves towards the brighter one. Fireflies move randomly if they have equal brightness.

- Thirdly, the brightness of the firefly correlates with the structure of its objective function.

```
1 Set parameters of FA (α,γ)
2 Set maximum number of iterations (MaxItr)
3 Formation the firefly population X_i (i=1,2, 3, …, n)
4 Define the objective function f(x) of X(x_1, x_2, … , x_d) ;
5 Compute light intensity I_i for each x_i with f(x)
6  for iteration= 1: MaxItr
7     for i= 1: n
8        for j=1: n
9           if I_j >I_i
10             with all d dimensions the ith firefly moves towards the jth firefly
11          end if
12          attractiveness changes with exp(−γr²)
13          evaluate solution and compute light intensity
14       end for
15    end for
16    rank the fireflies and find the current best, update the global best if necessary
17 end for
18 Results
```

Figure 3.1 Pseudocode of the classic firefly algorithm.

In Figure 3.1, the basic structure of FA is given as a pseudocode. In lines 1 and 2, the parameters α (randomization parameter), $\gamma$ (constant light absorption coefficient) and the number of iterations is set. Between lines 3-5, the starting population is determined and the objective functions of the individuals in the population are calculated. In lines 6-17, throughout the iteration number, the firefly with less brightness moves towards the firefly with higher brightness. Thus, the position is updated in each iteration, the current best and the global best are recorded at the end of the iteration, and the best results are ranked. The best solutions are transferred to the next iteration. In line 18, the best solution obtained at the end of all iterations is the result of the algorithm.

*3.2.1.1 Classic Firefly Algorithm Search Strategy*

The search strategy of the firefly algorithm consists of three main components. These are attractiveness, distance and movement. Position update is done with the mathematical equivalents of these components. In the firefly algorithm, each firefly represents a solution. The swarm represents all fireflies in the population. The objective function is represented by the light intensity or brightness of the fireflies.

In the algorithm for position updating, the attractiveness between two fireflies is calculated first. Attractiveness depends on light intensity or brightness that fireflies have. The light intensity or brightness is inversely proportional to the square of the distance as given in Equation 3.1.

$$I=1/r^2 \tag{3.1}$$

As the distance between fireflies decreases, the light intensity that they use to impress other fireflies increases. Therefore, the attractiveness between the two fireflies is increased. The attractiveness formula between two fireflies is given in Equation 3.2. If the light intensity or attractiveness of the *jth* firefly is greater than *i,* the *ith* firefly will move towards *jth* firefly. In this way, the search direction of the firefly or the algorithm is determined.

$$\beta\,(ij) = \beta_0\, e^{-\gamma r^2} \tag{3.2}$$

$\beta_0$ is the attraction when r=0. $\gamma$ is light absorption coefficient. The distance parameter, *r,* between two fireflies *i* and *j* is calculated with the Cartesian distance formula given in Equation 3.3.

$$r_{ij}=\|X_i - X_j\|=\sqrt{\sum_{k=1}^{d}(X_{ik} - X_{jk})} \tag{3.3}$$

In the formula, d indicates the total dimension number of fireflies, $X_{ik}$ while indicates kth dimension of *i.* After the attractiveness between the two fireflies is calculated, if the firefly *j* has a better light intensity or attractiveness than the firefly *i* , the movement step takes place. Movement formulation is given Equation 3.4. Thus, the new position of the *ith* firefly is determined.

$$X_i = X_j + \beta_0 e^{-\gamma r_{ij}^2}(X_j - X_i) + \alpha\ (\text{rand} - \frac{1}{2}) \tag{3.4}$$

Firefly *i* moves towards firefly *j*. α is randomization parameter. It takes a value between [0, 1]. rand is a random number that takes a value between 0 and 1. These values determine the step size of the firefly *i*. If the objective function value of the new firefly i solution is better than the previous solution, the position is updated.

*3.2.1.2 Literature Review for Firefly Algorithm*

The firefly algorithm has been proposed for many types of engineering and optimization problems so far. This sub-section reviews the studies in which the firefly algorithm was used to solve both the continuous and discrete optimization problems.

Yang (2008) developed and formalized the firefly algorithm, which is a metaheuristic solution method, with this study. In this article, the firefly algorithm was formalized in accordance with the continuous and maximization problems.

Yang (2010a) solved nonlinear design problems by firefly algorithm. Firefly algorithm performance was evaluated on some new and old test functions.

Jati & Suyonto (2011) proposed the solution of the traveling salesman problem with the discrete firefly algorithm by integrating the evolutionary algorithm. In the proposed algorithm, the distance step was made compatible with discrete variables and inversion mutation movement was used. Results showed that the firefly algorithm has a better performance than the memetic algorithm.

Sayadi, Ramezanian & Ghaffari - Nasab (2010) presented a firefly algorithm metaheuristic method to solve permutation flow shop scheduling problem and to shorten processing time. In this discrete version of the firefly algorithm sigmoid function was used. In addition, a local search approach was used to improve the solutions.

Osaba, Yang, Diaz, Onieva, Masegesa & Peralles (2017) tried to solve the vehicle routing problem with Firefly algorithm. In this study, fireflies moved with insertion function. Best firefly was selected to form new fireflies.

Gandomi, Yang & Alavi (2011) worked with some general structural problem types to evaluate the performance of the firefly algorithm. 6 structural optimization problems known in the literature were solved with the firefly algorithm. The same problems were also solved for performance measurement with simulated annealing (SA), Genetic algorithm (GA), particle swarm optimization (PSO) and harmony search (HS) metaheuristic methods and the results were compared. For the solved problems, it was seen that the firefly algorithm produced better results than these algorithms.

Kota (2012) proposed a discrete firefly algorithm to tackle the multiple travelling salesman problems. Also, the swap method was applied as a neighbourhood search to generate new solutions. The results were compared with harmony search and particle swarm optimization algorithms.

Rahmani & MirHassani (2014) proposed a hybrid firefly genetic algorithm for the capacitated facility location problem. They used the firefly algorithm to produce new results, and the new results are improved by using genetic algorithm operators. In the proposed hybrid algorithm, the results of small and medium-sized problems are compared with exact solution methods, while the results of large-scale problems are compared with PSO. While the proposed algorithm gave near optimal results for small and medium sized problems, it was concluded that it is applicable for large scale problems.

Marichelvam et al. (2014) solved the multi objective hybrid flow shop scheduling problem with firefly algorithm. For discretization, smallest position value was proposed. Obtained results were compared with genetic, simulated annealing and ant colony optimization algorithms.

Baykasoglu & Özsoydan (2014) developed a Firefly algorithm to solve the dynamic multidimensional knapsack problem. The related problem was also solved with GA, Differential Evolution (DE) and pure FA for performance comparison. It was observed that the results of the developed firefly algorithm gave much better results compared to these algorithms.

Li, Zhang, Zeng, Zhou & Liu (2016) modified the firefly algorithm for assembly sequence planning problem. To improve the global search and prohibit the algorithm from getting trapped in local search, new solutions were formed by moving the fireflies towards only the fireflies that were within visual range of the best firefly.

Tilahun & Ngnotchouye (2017) reviewed the discrete implementations of the firefly algorithm. The solutions of the problems with discrete variables with the firefly algorithm were examined and suggestions were made for future studies.

The studies given above show that the firefly algorithm produces good results for the problems studied. In some cases, the classical algorithm has been modified to improve the solutions and hybridized with other metaheuristic algorithms to make the classic firefly algorithm more efficient for the selected problem. Specially, these papers recommended working on discrete problems for future problems.

### *3.2.2 Bat Algorithm*

The bat algorithm (BA) is a bio-inspired metaheuristic algorithm originally introduced by Yang (2010b) for global optimization problems. The bat algorithm is formed on the basis of bats' use of echolocation features with the help of the frequency they emit. They use this feature to determine the position of any object to them and to establish communication between them. They create sound signals at a certain frequency to communicate with their prey, objects they see as obstacles, and with each other. They perform their actions by detecting the situation according to the return time or distance of the sound they produce. Inspired by these behavioural systems of bats, it was formulated and the bat algorithm was formed. This algorithm was first developed and applied for continuous problems. In the future, the bat algorithm was used in discrete problems.

Kongkaew (2017) listed the advantages of the bat algorithm and its features that distinguish it from other algorithms as given below.

- Owing to its automatic zooming feature, the local search density is high.

- With its frequency tuning feature, it can have the advantages of algorithms such as particle swarm optimization and harmony search.

- Since there is no crossover mechanism, it allows individuals in the population to maintain their characteristics.

- Having controllable parameter values during iterations.

Yang (2010b) listed the basic features of the bat algorithm as follows.

- Using the echolocation feature, bats can calculate the distance between them and their prey, as well as distinguish other objects from their prey.

- In order to catch their prey, bats move from $x_i$ position with velocity $v_i$, frequency $f_{min}$ , and loudness $A_0$.

- Being able to adjust their frequency and pulse emission rates according to their proximity to their prey;

In Figure 3.2, the basic structure of the bat algorithm is given through a pseudocode. First, the objective function is defined and the initial bat population is formed. Then the parameters of the bat algorithm are determined for each individual in the population. These parameters are the velocity$n$ $v_i$, frequency$f_i$, pulse rate $r_i$ and loudness $A_i$. In the algorithm, solutions are represented by $x_i$. Later initial steps, the position update steps start and the bats in the population change their positions depending on the frequency and velocity parameters and then new candidate solutions are evaluated. If the random number is greater than the pulse emission rate, local updates are made to good solutions. A new random number is generated for every individual. If this random number is less than the loudness rate and the updated individual objective function is less than current best objective function, new results are accepted.

*3.2.2.1 Classic Bat Algorithm Search Strategy*

The bat algorithm position update mechanism was formed by interpreting and mathematically expressing the echolocation behaviour of bats (Yang, 2010). In the

algorithm, first, the objective function, the parameters to be used in the algorithm pulse rate $r_i$, velocity $v_i$ and loudness $A_i$ must be defined.

```
1 Definition the objective function f(x);
2 Formation the bat population X = x1, x2…, xn.
3 for each bat x_i in the population do
4 the pulse rate r_i, velocity v_i and loudness A_i parameters are started
5 Define the pulse frequency f_i at x_i;
6 end for
7 repeat
8 for each bat x_i in the population do
9     Generate new solutions through Equations 3.5, 3.6 and 3.7;
10    if rand>r_i then
11      Select one solution among the best ones;
12     Generate a local solution around the best one;
13    end if
14    if rand<A_i and f (x_i) <f (x_*) then
15      Accept the new solution;
16      Increase r_i and reduce A_i;
17    end if
18 end for
19 until criterion achieved;
20 Select the global best result reached;
```

Figure 3.2 Pseudocode of the bat algorithm

Then the initial solutions are calculated. In the population, each *bat* denoted by the index *i* in the formulation. *Population* refers to the total number of individuals in the algorithm, denoted by *n*. After the initial population is formed in the algorithm, the strategy of searching for new solution begins in order to reach better solutions occurs by updating the bats' positions. Using the formula given in Equation 3.5 the frequencies at which each bat emitted echolocation pulses are determined. *Frequency* value *f* is a random number distributed uniformly in the range [$f_{min}$ ,$f_{max}$] for each bat in Equation 3.5. The parameter β is a randomly generated number in the [0, 1] interval.

$$f = f_{min} + (f_{min} - f_{max})\,\beta \tag{3.5}$$

In Equation 3.6 and Equation 3.7, velocity and position updates are made to produce new solutions. Solution and velocity are denoted by the indices *x* and *v*

respectively. Position of bat *i* at iteration *t* is represented by $x_i^t$. Velocity of bat *i* at iteration *t* is represented by $v_i^t$. The current best solution is denoted by $x_*$ in the swarm.

$$v_i^t = v_i^{(t-1)} + (x_i^t - x_*) f_i \tag{3.6}$$

$$x_i^t = x_i^{(t-1)} + v_i^t \tag{3.7}$$

In equation 3.8, the local search phase of the bat algorithm is formulated. If the random number *rand* is greater than $r_i$ a new solution is generated using a random movement for the local search phase, ε is a randomly generated number within the interval [-1, 1], and $A^t$ is the average loudness of the swarm at time step t.

$$x_{new} = x_{old} + \varepsilon A^t \tag{3.8}$$

Finally, two conditions are sought for each individual during the acceptance of new solutions. For the individual *i*, in the first condition, it is requested that the random number *rand* be less than the loudness parameter $A_i$. In the second condition, the new solution must be better than the old solution. If these two conditions occur, the new solution is accepted and the loudness $A_i$ and the rate $r_i$ of each bat have to be updated and the best results are updated after the movement for every bat. These update formulations are given 3.9 and 3.10 equations. In these equations, α and γ are predetermined fixed values. In order to record the best solutions, at the end of the iteration, the best solutions in the population are recorded and this loop continues until all iterations are completed.

$$A_i^{(t+1)} = \alpha A_i^t \tag{3.9}$$

$$r_i^{(t+1)} = r_i^0 [1 - \exp(-\gamma t] \tag{3.10}$$

*3.2.2.2 Literature Review for Bat Algorithm*

Bat algorithm is a more recently developed metaheuristic algorithm. In literature, bat algorithm has been proposed for the improvement of many optimization problems and some of the works composed using the bat algorithm given below.

Yang (2010b) presented the bat algorithm, which is a metaheuristic solution method. While forming the bat algorithm, it was developed based on the echo locality characteristics of bats. In this study, bat algorithm was applied to some test functions and the results are compared with the particle swarm optimization and simulated annealing algorithms.

Luo and Zhou, Xie, Ma & Li (2014) proposed a bat algorithm for the optimal permutation flow shop scheduling problem. Sub scheduling problems solved using the NEH heuristic. Also, some neighbourhood search algorithms were used to improve the solutions.

Zhou, Xie & Zheng (2013) proposed a hybrid bat algorithm in combination with the path relinking to solve the capacitated vehicle routing problem. GRASP method was also integrated into this proposed hybrid algorithm. In addition, single point local search mechanisms and subsequence used with neighbourhood search methods to produce new solutions.

Saji & Riffi (2016) studied on discrete version of bat algorithm to solve the symmetric travelling salesman problem. For this problem, neighbourhood search algorithms, 2 exchange crossover and 2 opt search, were used to improve solutions. Lastly, performance of the bat algorithm compared against the particle swarm optimization.

Gandomi, Yang, Alavi & Talatahari (2013) proposed a bat algorithm to solve the classical constrained benchmark problems. Bat algorithm solutions compared against other mostly used metaheuristics on the engineering design problems.

Osaba, Yang Diaz, Lopez – Garcia & Carbelledo (2016) implemented the bat algorithm for symmetric and asymmetric travelling salesman problem. 2 opt and 3 opt mechanisms were suggested as local search and movement steps of individuals.

Osaba, et al. (2019) improved a discrete bat algorithm for medical good distribution problem. They proposed swap and insertion methods for BA. The proposed algorithm was compared against the evolutionary simulated annealing algorithm, evolutionary algorithm, and FA. Obtained results demonstrated that the proposed algorithm produces better results than the compared algorithms.

Zhou, Xie & Tang (2013) proposed a modified bat algorithm to solve the permutation flow shop problem. Largest order value and NEH heuristics rules applied to adapt classic continuous bat algorithm steps to discrete version.

Mirjalili, Yang & Seyed Mohammed Mirjalili (2014) proposed the binary bat algorithm to solve 22 benchmark problems. The continuous working space has been converted to discrete working space with the v-shaped transfer function. Binary bat algorithm was compared against genetic algorithm and the binary PSO.

# CHAPTER FOUR

# PROPOSED HYBRID ALGORITHM FOR SOLVING ALTERNATIVE SUBGRAPH ASSEMBLY LINE BALANCING PROBLEM

## 4.1 Introduction

In this section, proposed hybrid firefly-bat algorithm for ASALBP-1 is explained. The application and computational study stages of the proposed hybrid algorithm are given in detail in the subsections.

## 4.2 A Hybrid Firefly Algorithm with Bat Algorithm for ASALBP

In this study, a hybrid firefly-bat algorithm is proposed to solve the ASALBP-1. By using the advantageous aspects of the firefly and bat algorithms, the proposed hybrid algorithm is aimed to discover the search space satisfactorily and find results closer to the optimum. Hybrid firefly algorithm updates the poor solutions and to accelerate its convergence speed (Guo, Wang, Wang & Wang, 2013). One of the advantages is automatic zooming aspect of the bat algorithm. This aspect provides more rate of convergence with other algorithms (Yang & He, 2013). Therefore, we used these two metaheuristic algorithms as hybrid to improve the solutions of ASALBP-1.

The flow diagram of the hybrid algorithm is shown in Figure 4.1. The proposed hybrid algorithm starts with setting the values of the needed parameters. After that the algorithm forms the swarm population. Later the hybrid algorithm continues with the firefly algorithm. The new individuals formed because of the firefly algorithm are caught with the bat algorithm local search and loudness operators. The proposed algorithm was terminated by recording the best results. In addition, all parts of the hybrid algorithm are explained in the following subsection.

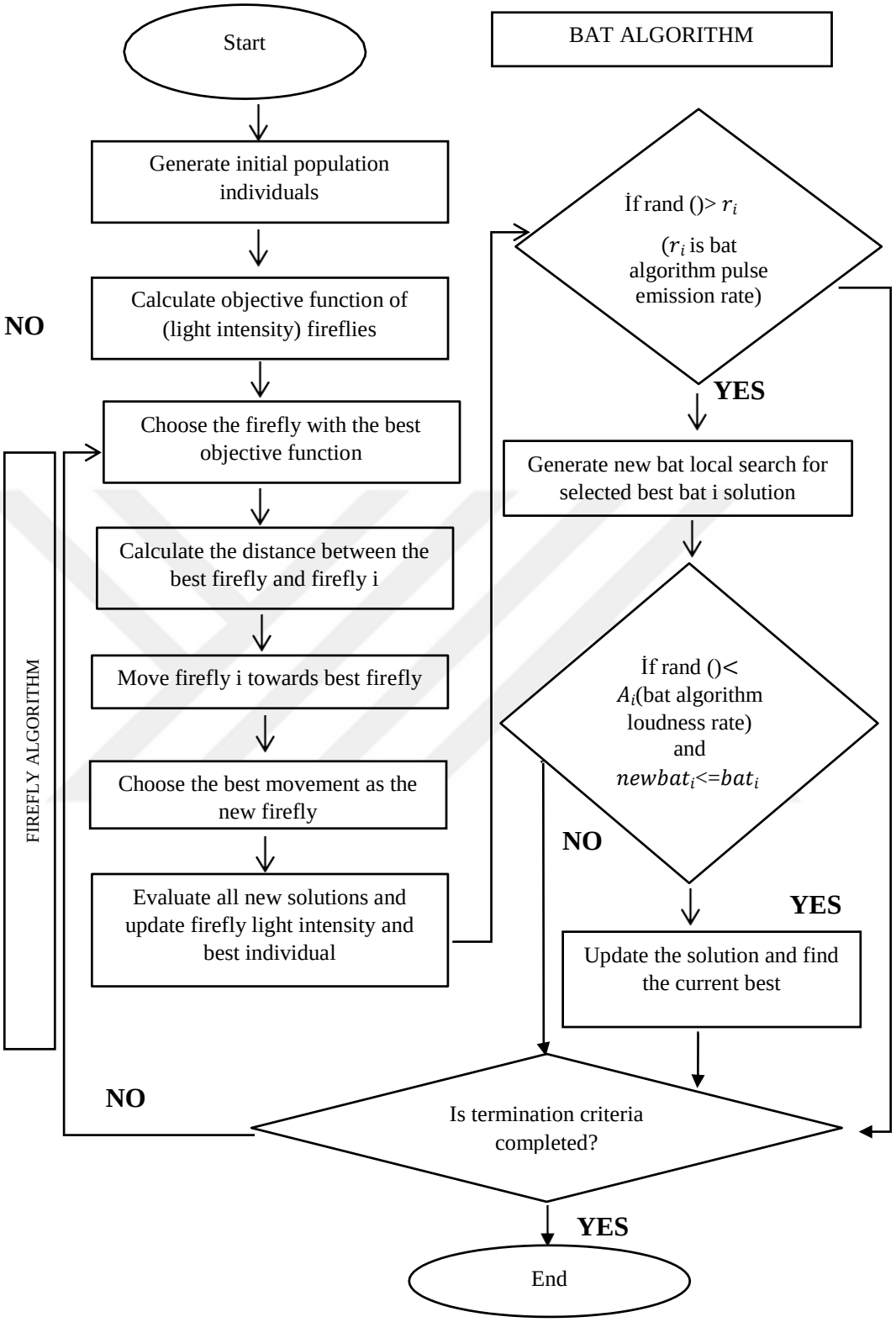


Figure 4.1 Flowchart of the proposed Hybrid FA-BA algorithm

### *4.2.1 Solution Representation*

Individuals consist of the sum of the tasks of the subgraph, chosen by the number of subassemblies that the assembly line balancing problem has. Solution representation for continuous problems is modelled with continuous numbers, while integers can be used for discrete problems in firefly and bat algorithms. Permutation based, binary based, and task-based solution representations have been generally used in the literature while solving discrete problems with metaheuristic algorithms. We used task-based representation as it is the best fit for the ASALBP-1.

In Figure 4.2, the precedence diagram of an example problem, which consist of 2 sub-assemblies with a cycle time of 20, and each sub assembly has 2 subgraphs, is given. The objective function of the ASALBP-1 is to minimize the number machines used. The best solution for the problem given in the below example under cycle time and precedence constraints is due to the choices of subgraph-2 and subgraph-3. The task-based representation of this solution, consist of subgraph-2 and subgraph-3, is shown in Figure 4.3.

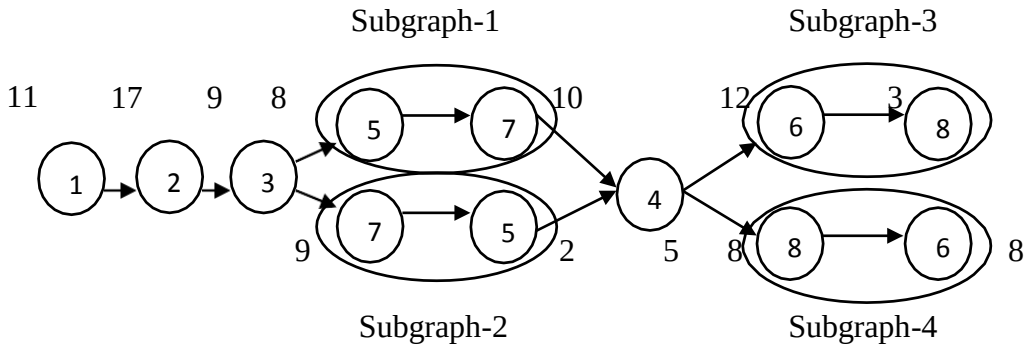


Figure 4.2 Precedence diagram for the example problem

| 1 | 2 | 3 | 7 | 5 | 4 | 6 | 8 |
|---|---|---|---|---|---|---|---|

Figure 4.3 Task based solution representation

Figure 4.2 shows an assembly process with precedence relationships and task times. Each tasks of this assembly process are considered as a dimension of the representation. The tasks are assigned to the workstations in order considering the precedence relations and time constraint of the workstation. When the workstation's cycle time was not enough to receive new job, a new workstation is opened, and all components of the individual is combined task based as shown in the solution with Figure 4.3.

### *4.2.2 Initial Population*

The proposed hybrid firefly-bat algorithm firstly constructs a population with a predefined size. The individual consists of fixed tasks and different tasks that the alternative subgraphs have of different subassemblies. Therefore, subgraph selection is made first to form individuals. Subgraphs' selections are made randomly for every individual. The population is formed from assembly tasks with the same or different subgraphs. All the alternatives option combinations are used at least once for each assembly.

After selecting the alternative subgraphs, the tasks are assigned to the workstations under the precedence and time constraints with the ranked positional weight technique (Helgeson and Birnie, 1961) and randomly for every individual. Since rank positional weight technique and random selection gave good results in previous heuristic studies (Capacho et al., 2006), these two heuristic methods are used to construct initial solutions. In the ranked positional weight technique, the priority weight of each task is calculated by looking at the precedence relations. The weight of a task is the sum of the processing times of its predecessor tasks and itself.

In the random initial solution, it is achieved by paying attention to only precedence and cycle time constraints.

### *4.2.3 Evolution of the Objective Function*

ASALBP-1 aims to assign the tasks of the selected subgraphs to the minimum workstation under the given precedence constraints and cycle time constraints.

### *4.2.4 Distance Calculation*

For first iteration after forming the initial population and later at the beginning of each iteration, every solution is tried to improve via the operators of firefly and bat algorithms. Objective function comparisons are made for each individual. If one individual's objective function is worse than or equal to the other individual's objective function, an improvement is made towards the better individual. While this process is done by comparing all fireflies in the firefly algorithm, we made a comparison only with the best individual in the swarm to reduce the resolution time. The classic firefly hamming distance formula given in Formulation 3.3 was used for distance calculation with the discrete version in the proposed algorithm. In the ASALBP-1, the distance formula is revised to improve the individuals because that problem is a discrete problem. The distance calculation procedure for this discrete problem is explained below.

The assembly orders of tasks for two different individuals are given. Hamming distance is the sum of the number of tasks in different order between the two assembly sequences. Hamming distance is indicated by *r*. Alternative subgraphs in ASALBP-1 can contain the same or different number of tasks. Therefore, compared two individuals may have the same number of tasks as well as a different number of tasks. In Figures 4.4 and 4.5, the distance between the *ith* and *jth* individuals is

calculated with different tasks (shown in bold) in different assembly order. As in Figure 4.4, if it has the same number of tasks, the distance is the same as the number of tasks in the individual whose order is not the same. In accordance with, the distance between the two individuals in Figure 4.4 is 4. If it has different number of tasks as in Figure 4.5, the distance is calculated over the individual with a larger total number of tasks. The missing numbers of tasks in the individual with a small number of tasks are indicated with zero at the end of the sequence. Thus, the distance between the individuals in Figure 4.5 is found as 4.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Individual – i | 1 | 2 | 3 | 5 | 7 | 4 | 6 | 8 |
| Individual – j | 1 | 2 | 3 | 7 | 5 | 4 | 8 | 6 |

Figure 4.4 The hamming distance representation of *i* to *j* from individuals with the same tasks

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Individual – i | 1 | 2 | 3 | 7 | 8 | 9 | 10 |
| Individual – j | 1 | 2 | 3 | 4 | 5 | 6 | 0 |

Figure 4.5 The hamming distance representation of *i* to *j* from individuals with the different tasks

### *4.2.5 Movement Operations*

The movement equation used for the continuous search space in the classic firefly algorithm is given in the Formulation 3.4. However, since proposed FA – BA hybrid algorithm is used for the solution of ASALBP-1, which has the discrete search space, it needs to be modified. For this reason, the movement of individual *i* towards an individual *j* with a better attraction is given by Formulation 4.1 for the proposed FA-

BA hybrid algorithm. The amount of movement of individual $i$, $x_i$, is a random integer chosen between 2 and $r_{ij}$. $r_{ij}$, is the amount of distance between the individuals of $i$ and $j$. In this step, the neighbourhood search algorithm is applied $x_i$ times for the existing solution. There are $x_i$ candidate solutions for the *ith* individual. One of these solutions will be selected for the new candidate solution.

$$x_i = \text{random } (2, r_{ij}) \tag{4.1}$$

The neighbourhood search algorithms applied for every individual in the algorithm are chosen randomly between the swap and insertion heuristics. The tasks to be relocated are selected randomly among the tasks that are suitable for precedence and time constraints. New neighbour solutions are searched in accordance with the structure of the selected neighbour search algorithm. The best solution is selected from the new solutions and compared with the previously determined best solution. If the new solution of individual $i$ is equal to or less than its previous solution, the new individual's position is updated. In the classical firefly algorithm at the end of iteration, new solutions are included in the previous solutions and the new best individuals as many as the population number are accepted for the next iteration. In the proposed algorithm, the objective function of each individual is compared with the objective function of the new individual in iteration, better and equal solutions are accepted, and the population of the next iteration is formed.

Table 4.1 Priority and cycle time constraints for an example problem

| **Tasks** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Priorities** | **-** | **-** | **-** | **1** | **2** | **4** | **5** | **6** | **7** | **8** | **3** |
| | | | | **2** | | | | | | **9** | **10** |

Table 4.2 Solution representation of example problem in table 4.1

| **Assembly order** | **Task time** | **Machine number** |
|---|---|---|
| **2** | **38** | **1** |
| **5** | **10** | **1** |
| **7** | **12** | **2** |
| **9** | **2** | **2** |
| **1** | **4** | **2** |
| **4** | **12** | **2** |
| **6** | **8** | **2** |
| **8** | **10** | **2** |
| **3** | **45** | **3** |
| **10** | **10** | **4** |
| **11** | **34** | **4** |

The movement operation is explained for a small example problem given in Tables 4.1 and 4.2. Producing new solutions through swap and insertion heuristics by paying attention to the constraints is shown in Figures 4.6 and 4.7. In Table 4.1, precedence relations information of an example problem with 11 tasks are given. Cycle time of the example problem is 48. The solution of the problem given in Tables 4.1 is given in Table 4.2. By giving the assembly order of the initial solution in the first column in Table 4.2, it is used to generate new feasible solutions with swap operation in Figure 4.6 and insertion operation in Figure 4.7. In Figure 4.6, the swap operation is formed by the displacement of task 9 and task 3. In Figure 4.7, a new feasible candidate solution is formed by replacing task 9 between tasks 3 and 10.

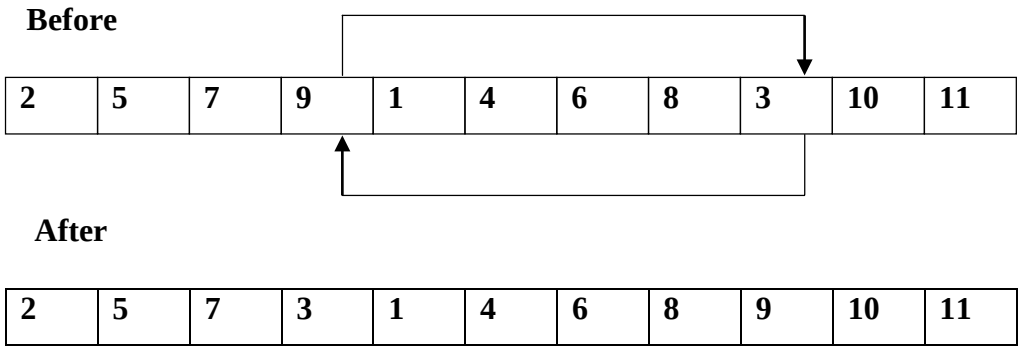


Figure 4.6 Swap operations for example problem in table 4.1 and 4.2

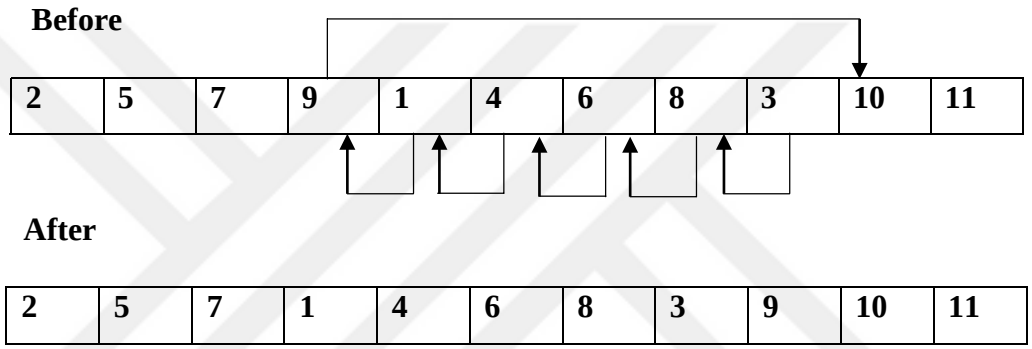


Figure 4.7 Insertion operations for example problem in table 4.1 and 4.2

### *4.2.6 Local Search Operation*

After applying the movement step of the firefly algorithm in the proposed hybrid algorithm, afterwards the local search step of the bat algorithm is applied. The *rand* symbol is a parameter whose value is determined at the beginning of the hybrid algorithm for the bat algorithm. If the random number denoted by $r_i$ determined for every individual is greater than the pulse emission rate is *rand*. For the *ith* individual, local neighbourhood search is applied if this parameter condition is happened. In order to improve candidate new solutions, the movement operation is performed only once in local search step. For neighbourhood search, swap and insertion methods are chosen randomly.

### *4.2.7 Accepting of New Solutions*

The last phase of the hybrid algorithm is the acceptance stage of new individuals. If a local search operation has been performed for an individual, there are two conditions for the acceptance of the new solution. The first condition is whether the objective function of the new solution formed by the realization of the local search step is equal to or better than the objective functions of the previous solution. The second condition is that the random number (*rand*) is less than the value of the loudness operator $A_i$, which is the bat algorithm parameter. The parameter *rand* is a randomly generated number in the [0, 1] interval. The parameter $A_i$ is a randomly generated number in the [0.7, 1]. If two conditions are met, the new solution is accepted. If the local search step does not occur, the result accepted in the movement step is the new solution. In these two cases, lastly, the solutions are compared with the best solution, and new solutions are accepted as new best solution if they are less than the best solution or equal the best solution.

### *4.2.8 Stopping Condition for the Proposed Hybrid Algorithm*

At this section, it will be explained which criteria the algorithm will be complied with. Since ASALBP has been tested in small, medium and large-scale problems and for the problem of different subgraphs, we used different stopping criteria according to the size of the problem. We set the maximum number of iterations as stopping criteria for small and medium sized problems. In large scale problems, we used time as stopping condition. The stopping criteria used in the algorithm are detailed in Table 4.3.

We determined the maximum number of iterations as 200 for small scale problems and 2000 for medium scale problems. For large-scale problems, we specified a runtime of 18000 seconds or 36000 seconds.

Table 4.3 The stopping conditions used in algorithms

| **Problem** | | | **Stopping conditions** |
|---|---|---|---|
| 5 Subgraph | Small size | | 200 iterations |
| | Medium size | | 2000 iterations |
| | Large size | | 18000 seconds |
| 8 Subgraph | Small size | | 200 iterations |
| | Medium size | | 2000 iterations |
| | Large size | | 18000 seconds |
| 11 Subgraph | Small size | | 200 iterations |
| | Medium size | | 2000 iterations |
| | Large size | Arcus | 18000 seconds |
| | | Bartholdi | 18000 seconds |
| | | Scholl | 36000 seconds |

## 4.3 Computational Study

The proposed algorithms are evaluated on 12 benchmark problems of ASALBP-1. Bat, firefly and the proposed hybrid FA-BA algorithms, for the alternative subgraph assembly line balancing problem, were coded in MATLAB R2013 and run on a PC with Intel Core İ3 2.27 GHz CPU, 4GB RAM, running Windows 7. The proposed algorithms were compared against previously performed 14 heuristics methods (Capacho *et al*., 2006b).

### *4.3.1 Benchmark Problems*

For the experiments, 12 data sets were used of Capacho and Pastor (2006a). The data sets available at assembly line balancing research web site www.assembly line balancing.de. In total, 133 problems with different characteristics have been solved. The data included 13 small-sized problems, 75 medium-sized problems and 45 large-

scale problems. The data consisted of different sub-graphs, tasks and work completion times. The data structure is detailed in the Table 4.4.

Table 4.4 Data sets

| Problem size | Problem | Cycle times | | | | | Number of subgraphs | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | 5 | 8 | 11 |
| | | | | | | | Number of tasks | | |
| Small size | Mansor | 48 | 62 | 94 | - | - | 11 | - | - |
| | Bowman | 20 | - | - | - | - | 10 | - | - |
| | Buxey | 30 | 36 | 54 | - | - | 29 | - | - |
| | Mitchell | 14 | 21 | 35 | - | - | 21 | 21 | - |
| Medium size | Gunther | 41 | 49 | 49 | 61 | 81 | 37 | 37 | 37 |
| | Kilbrid | 51 | 79 | 92 | 138 | 184 | 45 | 46 | 48 |
| | Hahn | 2004 | 2338 | 2806 | 3504 | 4676 | 56 | 56 | 63 |
| | Warnecke | 54 | 62 | 74 | 92 | 111 | 63 | 63 | 67 |
| | Tonge | 160 | 176 | 207 | 251 | 320 | 73 | 75 | 75 |
| Large size | Arc2 | 5785 | 6540 | 7916 | 9400 | 11570 | 115 | 121 | 125 |
| | Bartholdi | 403 | 470 | 564 | 705 | 805 | 151 | 157 | 160 |
| | Scholl | 1394 | 1584 | 1699 | 2049 | 2787 | 299 | 302 | 305 |

### *4.3.2 Parameter Settings*

As explained in previous chapters, classical firefly and bat algorithms are applied to the constraint problem, so some changes were made to the classic FA and BA. The parameters used for firefly Algorithm, Bat Algorithm and hybrid FA-BA are shown in Table 4.5. For FA, BA and hybrid FA-BA: population sizes are calculated problem based considering problem size, population number is related to the number

of tasks in the problem. The more the number of tasks related to the problem, the more the population size should increase in order to better scan the search space.

Table 4.5 Parameter of Hybrid FA-BA for the proposed ASALBP-1.

| Problem | | Parameters of hybrid FA - BA algorithm | | | |
|---|---|---|---|---|---|
| | | Population size | Neighbour search | A | r |
| 1 sub assembly | Small size | 20 | Insertion or Swap Functions | [0.7-1.00] | 0.1 |
| | Medium size | 40 | Insertion or Swap Functions | [0.7-1.00] | 0.1. |
| | Large size | 50 | Insertion or Swap Functions | [0.7-1.00] | 0.1 |
| 2 sub assembly | Small size | 20 | Insertion or Swap Functions | [0.7-1.00] | 0.1 |
| | Medium size | 40 | Insertion or Swap Functions | [0.7-1.00] | 0.1 |
| | Large size | 50 | Insertion or Swap Functions | [0.7-1.00] | 0.1 |
| 3 sub assembly | Small size | 20 | Insertion or Swap Functions | [0.7-1.00] | 0.1 |
| | Medium size | 40 | Insertion or Swap Functions | [0.7-1.00] | 0.1 |
| | Large size | 50 | Insertion or Swap Functions | [0.7-1.00] | 0.1 |

In addition, for BA and hybrid FA-BA; the bat algorithm parameters such as loudness constant *A* and pulse emission rate *r* constant were used. In the classical algorithm, for continuous problems, the loudness value *A* and pulse emission rate *r* in each iteration can change in each iteration of every individual, while we have taken the value of *A* besides *r* as constant for each iteration by referring to the previous studies in the discrete algorithm and making experiments. Pulse emission rate *r* is an integer between 0 and 1 in classical bat algorithm. This rate is an increasing value if prey is found. For proposed hybrid algorithm, to keep the search for solutions continuous, we took the rate *r* as a constant 0.1 for all iterations and individuals.

The loudness rate $A_i$ is random value that normally ranges from [0-1]. In order to increase the acceptance probability of new solutions, we took this rate as a value varying in the range of [0.7-1] for each iteration and individual.

### *4.3.3 Results*

This study aimed to improve the solutions of ASALBP-1 by using different metaheuristic algorithms such as firefly, bat and hybrid FA-BA algorithms. In this section, the operation of the hybrid FA-BA algorithm developed for ASALBP-1 has been evaluated. In addition to hybrid FA-BA algorithm, pure firefly and pure bat algorithms were added to evaluation, and ASALBP-1 was solved with these three algorithms. Proposed hybrid algorithm is formed by two metaheuristic solution methods, FA and BA algorithms. Firefly and bat algorithms were applied with some changes since ASALBP-1 is a discrete problem.

Performance evolution of the solutions of these three algorithms was compared with the work of Capacho et al. (2006b) for medium and large-scale problems. In small scale problems, bat, firefly and hybrid algorithms were evaluated among themselves, because heuristic results could not be reached for small problems.

In Capacho et al. (2006b), ASALBP-1 solution has proposed by 14 different heuristic solutions. These heuristics methods selected subgraphs with randomly and used different decision rules to assign tasks. 14 heuristic results are given in Appendix 2 for all data (Capacho et al., 2006b). We used the best results from these 14 heuristics in our objective function comparison. In addition, the results of FA, BA and hybrid FA-BA algorithms is compared with the two heuristic solution methods with the best results on their own. Also, algorithms were compared with each other; bat firefly and hybrid FA-BA algorithms in terms of objective function and computational time (CPU).

The results of these three algorithms are given in Table 4.6, Table 4.7, Table 4.8, Table 4.9, and Table 4.10. In these tables, the columns from the left show the problem name, the cycle times, the number of tasks, the number of alternative subgraphs, maximum task time plus total number of successors heuristic (maxTTS) and random heuristic running for 600 seconds (Random 600) in medium and large scale problem tables, the best-known heuristic results , and the minimum (Min), maximum (Max), average (Avg.) values, computational time (CPU) result obtained by firefly, bat and hybrid algorithms, respectively. A total of 133 problems consisting of 13 small scale, 75 medium scale and 45 large scale different cycle times and subgraphs were solved with 3 different algorithms.

Although, there are not many differences, it has been observed that the proposed hybrid algorithm is more feasible to achieve good results more times when the FA and BA algorithms are run multiple times. Solutions have been observed that the proposed hybrid algorithm produces good results.

Results with improvement values are shown in bold in Table 4.6, Table 4.7, Table 4.8, Table 4.9, and Table 4.10. Small and medium sized problems were run 10 times, while large scale problems were run 3 times.

In Table 4.6, solutions of 13 problems from small scale problems for ASALBP-1 are given. These small-scale problems solved with firefly, bat and hybrid FA-BA algorithms.

Table 4.6 Results of small-scale problem

| Problem | n | Cycle time | Subgraph | Best solution | Firefly algorithm | | | | Bat algorithm | | | | Hybrid algorithm | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Min | Max | Avg. | CPU | Min | Max | Avg. | CPU | Min | Max | Avg. | CPU |
| Bowman | 10 | 20 | 5 | 5 | 5 | 5 | 5 | 13.478 | 5 | 5 | 5 | 15.678 | 5 | 5 | 5 | 14.648 |
| Mansor | 11 | 48 | 5 | 4 | 4 | 4 | 4 | 19.656 | 4 | 4 | 4 | 21.200 | 4 | 4 | 4 | 21.294 |
| | | 62 | 5 | 3 | 3 | 4 | 3.2 | 18.174 | 3 | 3 | 3 | 21.855 | 3 | 3 | 3 | 20.997 |
| | | 94 | 5 | 2 | 2 | 2 | 2 | 17.955 | 2 | 2 | 2 | 20.311 | 2 | 2 | 2 | 19.936 |
| Mitchell | 21 | 14 | 5 | 8 | 8 | 8 | 8 | 34.445 | 8 | 8 | 8 | 40.856 | 8 | 8 | 8 | 47.985 |
| | | 21 | 5 | 5 | 5 | 5 | 5 | 41.605 | 5 | 5 | 5 | 47.705 | 5 | 5 | 5 | 41.511 |
| | | 35 | 5 | 3 | 3 | 3 | 3 | 37.596 | 3 | 3 | 3 | 42.213 | 3 | 3 | 3 | 42.588 |
| | 21 | 14 | 8 | 8 | 8 | 8 | 8 | 33.961 | 8 | 8 | 8 | 35.365 | 8 | 8 | 8 | 35.989 |
| | | 21 | 8 | 5 | 5 | 5 | 5 | 39.093 | 5 | 5 | 5 | 39.998 | 5 | 5 | 5 | 43.321 |
| | | 35 | 8 | 3 | 3 | 3 | 3 | 36.753 | 3 | 3 | 3 | 37.643 | 3 | 3 | 3 | 40.981 |
| Buxey | 29 | 30 | 8 | 12 | 12 | 12 | 12 | 90.340 | 12 | 12 | 12 | 100.215 | 12 | 12 | 12 | 98.467 |
| | | 36 | 8 | 10 | 10 | 10 | 10 | 87.345 | 10 | 10 | 10 | 96.907 | 10 | 10 | 10 | 97.547 |
| | | 54 | 8 | 7 | 7 | 7 | 7 | 81.182 | 7 | 7 | 7 | 93.163 | 7 | 7 | 7 | 95.706 |

With these three algorithms, the best results known in the literature were achieved at the same rate at a minimum value. The minimum, maximum values of the algorithms were found to be equal; neither of them was superior to the other. But BA and hybrid FA-BA algorithms provided better results at average in Mansor problem with 62 cycle time. Although, there are no obvious differences between the algorithms; bat, firefly and proposed hybrid, in terms of CPU time, the firefly algorithm reached a solution in a shorter time with a small difference.

FA, BA and hybrid algorithm solutions of 1 subassembly medium level problems are given in Table 4.7. Different data and cycle times are evaluated for 25 problems.

Table 4.7 Results of medium-scale problems with 5 subgraphs

| Problem | n | Cycle time | Sub graph | Max TTS | Random 600 | Best solution | Firefly algorithm | | | | Bat algorithm | | | | Hybrid algorithm | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Min | Max | Avg | CPU | Min | Max | Avg | CPU | Min | Max | Avg | CPU |
| Gunther | 37 | 41 | 5 | 14 | 14 | 14 | 14 | 14 | 14 | 2249.253 | 14 | 14 | 14 | 2421.859 | 14 | 14 | 14 | 2425.066 |
| | | 44 | | 12 | 12 | 12 | 12 | 12 | 12 | 2141.255 | 12 | 12 | 12 | 2558.400 | 12 | 12 | 12 | 2331.381 |
| | | 49 | | 11 | 11 | 11 | 11 | 11 | 11 | 2083.630 | 11 | 11 | 11 | 2061.760 | 11 | 11 | 11 | 2405.983 |
| | | 61 | | 9 | 9 | 9 | 9 | 9 | 9 | 2048.607 | 9 | 9 | 9 | 2333.377 | 9 | 9 | 9 | 2350.434 |
| | | 81 | | 7 | 7 | 7 | 7 | 7 | 7 | 2038.140 | 7 | 7 | 7 | 2339.474 | 7 | 7 | 7 | 2375.625 |
| Kilbrid | 45 | 57 | 5 | 10 | 10 | 10 | 10 | 10 | 10 | 3555.031 | 10 | 10 | 10 | 4130.124 | 10 | 10 | 10 | 4085.212 |
| | | 79 | | 8 | 7 | 7 | 7 | 8 | 7.8 | 3683.244 | 7 | 8 | 7.6 | 4132.129 | 7 | 8 | 7.5 | 4050.489 |
| | | 92 | | 7 | 6 | 6 | 6 | 7 | 6.8 | 3718.237 | 6 | 7 | 6.9 | 4395.317 | 6 | 7 | 6.5 | 3957.796 |
| | | 138 | | 4 | 4 | 4 | 4 | 4 | 4 | 3694.570 | 4 | 4 | 4 | 4326.068 | 4 | 4 | 4 | 4130.114 |
| | | 184 | | 4 | 3 | 3 | 3 | 3 | 3 | 3589.148 | 3 | 3 | 3 | 4274.422 | 3 | 3 | 3 | 4002.509 |
| Hann | 56 | 2004 | 5 | 8 | 8 | 8 | 8 | 8 | 8 | 4074.679 | 8 | 8 | 8 | 4539.843 | 8 | 8 | 8 | 4655.636 |
| | | 2338 | | 8 | 8 | 7 | 7 | 7 | 7 | 4247.389 | 7 | 7 | 7 | 4388.966 | 7 | 7 | 7 | 4640.958 |
| | | 2806 | | 6 | 6 | 6 | 6 | 6 | 6 | 4142.116 | 6 | 6 | 6 | 4200.029 | 6 | 6 | 6 | 4629.647 |
| | | 3507 | | 5 | 5 | 5 | 5 | 5 | 5 | 4129.091 | 5 | 5 | 5 | 4444.653 | 5 | 5 | 5 | 4568.565 |
| | | 4676 | | 4 | 4 | 4 | 4 | 4 | 4 | 4137.469 | 4 | 4 | 4 | 4446.464 | 4 | 4 | 4 | 4101.600 |
| Warnecke | 63 | 54 | 5 | 32 | 33 | 32 | 32 | 32 | 32 | 5697.390 | **31** | 32 | **31.3** | 5834.421 | **31** | 32 | **31.3** | 6454.896 |
| | | 62 | | 28 | 28 | 28 | **27** | 28 | **27.7** | 5708.922 | **27** | 28 | **27.6** | 5839.385 | **27** | 28 | **27.3** | 6308.785 |
| | | 74 | | 22 | 23 | 22 | 22 | 22 | 22 | 5692.794 | 22 | 22 | 22 | 5844.610 | 22 | 22 | 22 | 6183.312 |
| | | 92 | | 18 | 18 | 18 | 18 | 18 | 18 | 5712.861 | 18 | 18 | 18 | 5990.444 | 18 | 18 | 18 | 6270.693 |
| | | 111 | | 15 | 15 | 15 | 15 | 15 | 15 | 5639.924 | 15 | 15 | 15 | 6194.957 | 15 | 15 | 15 | 6180.652 |
| Tongue | 73 | 160 | 5 | 24 | 23 | 23 | 23 | 23 | 23 | 8580.201 | 23 | 23 | 23 | 9047.178 | 23 | 23 | 23 | 9243.129 |
| | | 176 | | 21 | 22 | 21 | 21 | 21 | 21 | 8596.613 | 21 | 21 | 21 | 9101.774 | 21 | 21 | 21 | 9629.540 |
| | | 207 | | 18 | 18 | 18 | 18 | 18 | 18 | 8516.756 | 18 | 18 | 18 | 9288.343 | 18 | 18 | 18 | 8995.05 |
| | | 251 | | 16 | 15 | 15 | 15 | 15 | 15 | 8596.613 | 15 | 15 | 15 | 9402.378 | 15 | 15 | 15 | 9493.005 |
| | | 320 | | 12 | 12 | 12 | 12 | 12 | 12 | 8211.231 | 12 | 12 | 12 | 9413.162 | 12 | 12 | 12 | 8296.65 |

In the comparison of FA, BA and hybrid algorithm with the best of the heuristics, there has been improvement in Warnecke problem with cycle times of 54 and 62 for the BA and hybrid algorithm. There has been improvement in Warnecke problem with cycle time of 62 for the FA. The best results of the heuristics are achieved at a rate of % 100 at minimum values and at the rate of % 92 at maximum values for FA, BA and hybrid algorithm. In average values, FA 93.6 percent, BA % 94, hybrid algorithm % 96 reached best heuristic solutions. Although FA, BA and hybrid algorithm find equal results at maximum values, BA and hybrid algorithm found better minimum value than FA for only 1 problem. The hybrid algorithm at the average value is superior to % 12 BA. BA is also % 12 better than FA.

In the comparison of maxTTS heuristic with FA, BA and hybrid FA-BA, there has been improvement, in Kilbrid problem with cycle times of 79, 92 and 184, in Hann problem with cycle time of 2338, in Warnecke problem with cycle times of 54 and 62, in Tongue problem with cycle times of 160 and 251, for the BA and hybrid FA-BA. There has been improvement in Kilbrid problem with cycle times of 79, 92 and 184, in Hann problem with cycle time of 2338, in Warnecke problem with cycle time 62, in Tongue problem with cycle times of 160 and 251 for the FA. The best results of the maxTTS heuristic are achieved at a rate of % 100 at minimum, maximum and average values for FA, BA and hybrid FA-BA.

In the comparison of random_600 heuristic FA, BA and hybrid FA - BA, there has been improvement, in Warnecke problem with cycle times of 54, 62 and 74, in Tongue problem with cycle times of 176, for the FA, BA and hybrid FA-BA. The best results of the random heuristic are achieved at a rate of % 100 at minimum values at the rate of % 92 at maximum values for FA, BA and hybrid FA-BA. In average values, FA 93.6 percent, BA % 94, hybrid algorithm % 96 reached best heuristic solutions. In Table 4.8, the solutions of 2 subassembly medium scale problems with FA, BA and hybrid FA-BA algorithms are given.

Table 4.8 Results of medium-scale problems with 8 subgraphs

| Problem | n | Cycle time | Sub graph | Max TTS | Random 600 | Best solution | Firefly algorithm | | | | Bat algorithm | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Min | Max | Avg | CPU | Min | Max | Avg | CPU | Min | Max | Avg |
| Gunther | 37 | 41 | 8 | 14 | 14 | 14 | 14 | 14 | 14 | 2090.411 | 14 | 14 | 14 | 2847.727 | 14 | 14 | 14 |
| | | 44 | | 12 | 12 | 12 | 12 | 12 | 12 | 2071.680 | 12 | 12 | 12 | 2589.650 | 12 | 12 | 12 |
| | | 49 | | 11 | 11 | 11 | 11 | 11 | 11 | 2168.400 | 11 | 11 | 11 | 2270.762 | 11 | 11 | 11 |
| | | 61 | | 9 | 9 | 9 | 9 | 9 | 9 | 2020.212 | 9 | 9 | 9 | 2242.044 | 9 | 9 | 9 |
| | | 81 | | 7 | 7 | 7 | 7 | 7 | 7 | 1758.908 | 7 | 7 | 7 | 2144.182 | 7 | 7 | 7 |
| Kilbrid | 45 | 57 | 8 | 10 | 10 | 10 | 10 | 10 | 10 | 3715.100 | 10 | 10 | 10 | 4855.206 | 10 | 10 | 10 |
| | | 79 | | 8 | 7 | 7 | 7 | 7 | 7 | 3730.100 | 7 | 7 | 7 | 3525.633 | 7 | 7 | 7 |
| | | 92 | | 7 | 6 | 6 | 6 | 6 | 6 | 3808.022 | 6 | 7 | 6.4 | 3354.050 | 6 | 7 | 6.2 |
| | | 138 | | 4 | 4 | 4 | 4 | 4 | 4 | 3853.088 | 4 | 4 | 4 | 3497.016 | 4 | 4 | 4 |
| | | 184 | | 4 | 3 | 3 | 3 | 3 | 3 | 3766.422 | 3 | 3 | 3 | 3650.430 | 3 | 3 | 3 |
| Hann | 56 | 2004 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 4324.340 | 8 | 8 | 8 | 3904.457 | 8 | 8 | 8 |
| | | 2338 | | 8 | 7 | 7 | 7 | 7 | 7 | 4240.470 | 7 | 7 | 7 | 4524.000 | 7 | 7 | 7 |
| | | 2806 | | 6 | 6 | 6 | 6 | 6 | 6 | 3826.013 | 6 | 6 | 6 | 3260.400 | 6 | 6 | 6 |
| | | 3507 | | 5 | 5 | 5 | 5 | 5 | 5 | 3829.620 | 5 | 5 | 5 | 3806.400 | 5 | 5 | 5 |
| | | 4676 | | 4 | 4 | 4 | 4 | 4 | 4 | 3827.465 | 4 | 4 | 4 | 3660.866 | 4 | 4 | 4 |
| Warnecke | 63 | 54 | 8 | 32 | 33 | 32 | 32 | 32 | 32 | 5779.013 | **31** | 32 | **31.5** | 6251.62 | **31** | 32 | **31.4** |
| | | 62 | | 28 | 28 | 28 | **27** | 28 | **27.9** | 5888.855 | **27** | 28 | **27.5** | 6256.984 | **27** | 28 | **27.4** |
| | | 74 | | 22 | 23 | 22 | 22 | 22 | 22 | 5714.005 | 22 | 22 | 22 | 6240.675 | 22 | 22 | 22 |
| | | 92 | | 18 | 19 | 18 | 18 | 18 | 18 | 4566.563 | 18 | 18 | 18 | 6382.655 | 18 | 18 | 18 |
| | | 111 | | 15 | 15 | 15 | 15 | 15 | 15 | 4856.833 | 15 | 15 | 15 | 6287.013 | 15 | 15 | 15 |
| Tongue | 73 | 160 | 8 | 24 | 23 | 23 | 23 | 23 | 23 | 8589.190 | 23 | 23 | 23 | 8798.979 | 23 | 23 | 23 |
| | | 176 | | 21 | 21 | 21 | 21 | 21 | 21 | 8410.092 | 21 | 21 | 21 | 8608.044 | 21 | 21 | 21 |
| | | 207 | | 18 | 18 | 18 | 18 | 18 | 18 | 7832.844 | 18 | 18 | 18 | 8819.400 | 18 | 18 | 18 |
| | | 251 | | 16 | 15 | 15 | 15 | 15 | 15 | 7593.124 | 15 | 15 | 15 | 8469.12 | 15 | 15 | 15 |
| | | 320 | | 12 | 12 | 12 | 12 | 12 | 12 | 8325.838 | 12 | 12 | 12 | 8835.96 | 12 | 12 | 12 |

In the comparison of FA, BA and hybrid algorithm with the best of the heuristics, better results were found in the solutions of the Warnecke problem with 54, 62 cycle times for BA, hybrid algorithm. In FA, all improvements were performed in the same as other algorithms, except for the Warnecke problem with a cycle time of 54. All three algorithms have found the best of % 100 heuristics at minimum values. At maximum values, FA achieved the best results of heuristics by % 100, BA % 92, and hybrid % 96. In average values, FA reached the best of heuristics by % 100, BA reached at % 97.2 and the hybrid algorithm reached at % 99.2. The hybrid algorithm is % 12 better than BA in average values. FA and BA have the same rate average values.

In the comparison of maxTTS heuristic with FA, BA and hybrid algorithm, better results were found in Kilbrid problem with cycle times of 79, 92 and 184, in Hann problem with cycle time of 2338, in Warnecke problem with cycle times of 54 and 62, in Tongue problem with cycle times of 160 and 251, for BA and hybrid algorithm. All three algorithms have found the best of % 100 heuristic at minimum values. At maximum and average values, FA and hybrid algorithm achieved the best results of heuristic by % 100. In maximum values, BA achieved the best results of heuristic by % 96. In average values, BA reached the best of heuristics by % 99.6.

In the comparison of random_600 heuristics with FA, BA and hybrid algorithm, there has been improvement, in Warnecke problem with cycle times of 54, 62, 74 and 92 for the FA, BA and hybrid algorithm. The best results of the random_600 heuristics are achieved at a rate of % 100 at minimum values for FA, BA and hybrid. While the FA reached the maximum values by % 100 the BA and hybrid algorithm reached % 96. In average values, FA 93.6 percent, BA % 94, hybrid algorithm % 96 reached.

In Table 4.9, the results of 3 subassembly medium scale problems with FA, BA and hybrid FA-BA algorithms are given.

Table 4.9 Results of medium-scale problems with 11 subgraphs

| Problem | n | Cycle time | Sub graph | Max TTS | Random 600 | Best solution | Firefly algorithm | | | | Bat algorithm | | | | Hybrid algorithm | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Min | Max | Avg | CPU | Min | Max | Avg | CPU | Min | Max | Avg | CPU |
| Gunther | 37 | 41 | 11 | 14 | 14 | 14 | 14 | 14 | 14 | 2115.408 | 14 | 14 | 14 | 2316.613 | 14 | 14 | 14 | 2335.222 |
| | | 44 | | 12 | 12 | 12 | 12 | 12 | 12 | 1894.183 | 12 | 12 | 12 | 2745.926 | 12 | 12 | 12 | 2730.106 |
| | | 49 | | 11 | 11 | 11 | 11 | 11 | 11 | 1862.967 | 11 | 11 | 11 | 1988.469 | 11 | 11 | 11 | 2355.652 |
| | | 61 | | 9 | 9 | 9 | 9 | 9 | 9 | 2050.744 | 9 | 9 | 9 | 2113.207 | 9 | 9 | 9 | 2667.634 |
| | | 81 | | 7 | 7 | 7 | 7 | 7 | 7 | 1457.726 | 7 | 7 | 7 | 2019.278 | 7 | 7 | 7 | 2605.300 |
| Kilbrid | 48 | 57 | 11 | 10 | 10 | 10 | 10 | 10 | 10 | 3639.900 | 10 | 10 | 10 | 4020.200 | 10 | 10 | 10 | 4081.875 |
| | | 79 | | 8 | 7 | 7 | 7 | 7 | 7 | 2531.007 | 7 | 7 | 7 | 3495.399 | 7 | 7 | 7 | 4097.717 |
| | | 92 | | 7 | 6 | 6 | 6 | 7 | 6.5 | 2541.956 | 6 | 7 | 6.4 | 4198.533 | 6 | 7 | 6.5 | 4009.785 |
| | | 138 | | 4 | 4 | 4 | 4 | 4 | 4 | 3876.756 | 4 | 4 | 4 | 3401.752 | 4 | 4 | 4 | 4368.745 |
| | | 184 | | 4 | 3 | 3 | 3 | 3 | 3 | 2842.305 | 3 | 3 | 3 | 4180.887 | 3 | 3 | 3 | 4247.012 |
| Hann | 63 | 2004 | 11 | 8 | 8 | 8 | 8 | 8 | 8 | 4010.730 | 8 | 8 | 8 | 4228.628 | 8 | 8 | 8 | 4711.277 |
| | | 2338 | | 8 | 7 | 7 | 7 | 7 | 7 | 4598.504 | 7 | 7 | 7 | 4134.188 | 7 | 7 | 7 | 4741.615 |
| | | 2806 | | 6 | 6 | 6 | 6 | 6 | 6 | 4224.715 | 6 | 6 | 6 | 3697.388 | 6 | 6 | 6 | 4380.396 |
| | | 3507 | | 5 | 5 | 5 | 5 | 5 | 5 | 4598.834 | 5 | 5 | 5 | 4220.113 | 5 | 5 | 5 | 4701.163 |
| | | 4676 | | 8 | 8 | 8 | **4** | **4** | **4** | 3975.630 | **4** | **4** | **4** | 3851.457 | **4** | **4** | **4** | 3812.980 |
| Warnecke | 67 | 54 | 11 | 32 | 32 | 32 | **31** | 32 | **31.1** | 5797.701 | 32 | 32 | 32 | 6328.753 | **31** | 32 | **31.5** | 6346.220 |
| | | 62 | | 28 | 28 | 28 | 28 | 28 | 28 | 5739.062 | **27** | 28 | **27.9** | 6342.476 | 28 | 28 | 28 | 6451.226 |
| | | 74 | | 22 | 23 | 22 | 22 | 22 | 22 | 5639.090 | 22 | 22 | 22 | 6325.224 | 22 | 22 | 22 | 6298.362 |
| | | 92 | | 18 | 19 | 18 | 18 | 18 | 18 | 6147.042 | 18 | 18 | 18 | 6494.214 | 18 | 18 | 18 | 6831.265 |
| | | 111 | | 15 | 15 | 15 | 15 | 15 | 15 | 6163.969 | 15 | 15 | 15 | 6499.304 | 15 | 15 | 15 | 6721.95 |
| Tongue | 75 | 160 | 11 | 24 | 23 | 23 | 23 | 23 | 23 | 8516.200 | 23 | 23 | 23 | 9233.980 | 23 | 23 | 23 | 6715.851 |
| | | 176 | | 21 | 21 | 21 | 21 | 21 | 21 | 8397.486 | 21 | 21 | 21 | 9197.860 | 21 | 21 | 21 | 8882.373 |
| | | 207 | | 18 | 18 | 18 | 18 | 18 | 18 | 8621.767 | 18 | 18 | 18 | 9229.007 | 18 | 18 | 18 | 8877.450 |
| | | 251 | | 15 | 15 | 15 | 15 | 15 | 15 | 8545.685 | 15 | 15 | 15 | 9249.507 | 15 | 15 | 15 | 9082.217 |
| | | 320 | | 12 | 12 | 12 | 12 | 12 | 12 | 7790.730 | 12 | 12 | 12 | 9477.421 | 12 | 12 | 12 | 9024.122 |

In the Table 4.9, the comparison of FA, BA and hybrid algorithm with the best of the heuristics, Hann problem with 4676 cycle time and Warnecke problem with 54 cycle time better results were produced than best heuristic solutions for FA and hybrid algorithm. In BA, better results were produced in Hann problem with 4676 cycle time and in Warnecke problem with 62 cycle time. FA, BA and hybrid algorithm found the best of heuristics at minimum values of % 100. At maximum values FA, BA and hybrid algorithm found % 92 best of heuristics. For average values, the best results of the heuristics were achieved by FA and hybrid algorithm at % 94.4 and BA at % 95.2. For problems with 11 subgraphs, BA is superior % 8 to FA and hybrid algorithm, and the FA algorithm is superior % 4 to the hybrid algorithm.

The comparison of FA, BA and hybrid algorithm with the maxTTS heuristic, Kilbrid problem with 79, 92, 184 cycle times and Hann problem with 2338, 4676 cycle times results were produced than best heuristic solutions for FA, BA and hybrid algorithm. In addition, FA and hybrid algorithm improved the Warnecke problem with 54 cycle time; while BA improved the problem data with 62 cycle time FA, BA and hybrid algorithm found the best of heuristics at minimum values of % 100. At maximum values FA, BA and hybrid algorithm found % 96 best of heuristics. For average values, the best results of the heuristics were achieved by FA and hybrid algorithm at % 99.2 and BA at % 99.6.

The comparison of FA, BA and hybrid algorithm with the random_600, Hann problem with 4676 cycle time and Warnecke problem with 54, 74, 92 cycle time better results were produced than best heuristic solutions for FA and hybrid algorithm. In BA, better results were produced in Hann problem with 4676 cycle time and in Warnecke problem with cycle times of 62, 74 and 92. FA, BA and hybrid algorithm found the best of heuristics at minimum values of % 100. At maximum values FA, BA and hybrid algorithm found % 96 best of heuristics. For average values, the best results of the heuristics were achieved by FA and hybrid algorithm at % 97.6 and BA at % 98.

In total, 75 medium scale problems were improved in 7 data for medium scale problems with the comparison of the best heuristics' solutions. There has been an improvement of % 9.33 in total. FA, BA and hybrid algorithm achieved the best of heuristics by % 100. At maximum, FA found % 94.67, BA found % 92, hybrid found % 93.33 heuristics. At minimum values, FA showed an improvement of %5.33, while BA and hybrid algorithm improved by % 8. BA and hybrid algorithm gave better result than FA at minimum values. At minimum, the values of BA and hybrid algorithm gave the same results, but in average values, the hybrid algorithm was superior to the BA. In terms of CPU performance evaluation, firefly algorithm gave better results than BA and hybrid algorithms.

The algorithms have different improvements when comparing the algorithms with the maxTTS heuristic. According to this heuristic, the firefly algorithm provided %28 improvements at minimum values with 21 problems, while the bat and hybrid algorithms provided %32 improvement with 24 problems. 100% of the best values were reached at the maximum values.

According to the random heuristic, which was run for 600 seconds, the firefly algorithm provided % 16 improvements with 12 problems in minimum values, while the bat and hybrid algorithms provided %17.33 improvement with 13 problems. At maximum values, the firefly algorithm reached the results of this heuristic with 96 percent, while the bat and hybrid algorithms reached the results of this heuristic at a rate of 94.7 percent.

In Table 4.10, FA, BA and hybrid algorithm solutions of large-scale problems are given.

Table 4.10 Results of large-scale problems

| Problem | n | Cycle time | Sub graph | maxTTS | Random 600 | Best solution | Firefly algorithm | | | | Bat algorithm | | | | Hybrid algorithm | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Min | Max | Avg. | CPU | Min | Max | Avg. | CPU | Min | Max | Avg. | CPU |
| Arcus | 115 | 5785 | 5 | 27 | 28 | 27 | 27 | 27 | 27 | 18000 seconds | 27 | 27 | 27 | 18000 seconds | 27 | 27 | 27 | 18000 seconds |
| | | 6540 | 5 | 25 | 25 | 24 | 24 | 24 | 24 | | 24 | 24 | 24 | | 24 | 24 | 24 | |
| | | 7916 | 5 | 20 | 20 | 20 | 20 | 20 | 20 | | 20 | 20 | 20 | | 20 | 20 | 20 | |
| | | 9400 | 5 | 17 | 17 | 17 | 17 | 17 | 17 | | 17 | 17 | 17 | | 17 | 17 | 17 | |
| | | 11570 | 5 | 14 | 14 | 14 | 14 | 14 | 14 | | 14 | 14 | 14 | | 14 | 14 | 14 | |
| | 121 | 5785 | 8 | 27 | 28 | 27 | 27 | 27 | 27 | | 27 | 27 | 27 | | 27 | 27 | 27 | |
| | | 6540 | 8 | 25 | 25 | 24 | 24 | 24 | 24 | | 24 | 24 | 24 | | 24 | 24 | 24 | |
| | | 7916 | 8 | 20 | 21 | 20 | 20 | 20 | 20 | | 20 | 20 | 20 | | 20 | 20 | 20 | |
| | | 9400 | 8 | 17 | 17 | 17 | 17 | 17 | 17 | | 17 | 17 | 17 | | 17 | 17 | 17 | |
| | | 11570 | 8 | 14 | 14 | 14 | 14 | 14 | 14 | | 14 | 14 | 14 | | 14 | 14 | 14 | |
| | 125 | 5785 | 11 | 27 | 28 | 27 | 27 | 27 | 27 | | 27 | 27 | 27 | | 27 | 27 | 27 | |
| | | 6540 | 11 | 25 | 25 | 24 | 24 | 24 | 24 | | 24 | 24 | 24 | | 24 | 24 | 24 | |
| | | 7916 | 11 | 20 | 20 | 20 | 20 | 20 | 20 | | 20 | 20 | 20 | | 20 | 20 | 20 | |
| | | 9400 | 11 | 17 | 17 | 17 | 17 | 17 | 17 | | 17 | 17 | 17 | | 17 | 17 | 17 | |
| | | 11570 | 11 | 14 | 14 | 14 | 14 | 14 | 14 | | 14 | 14 | 14 | | 14 | 14 | 14 | |
| Bartholdi | 151 | 403 | 5 | 15 | 15 | 15 | 15 | 15 | 15 | | 15 | 15 | 15 | | 15 | 15 | 15 | |
| | | 470 | 5 | 13 | 13 | 13 | 13 | 13 | 13 | | 13 | 13 | 13 | | 13 | 13 | 13 | |
| | | 564 | 5 | 11 | 11 | 11 | 11 | 11 | 11 | | 11 | 11 | 11 | | 11 | 11 | 11 | |
| | | 705 | 5 | 9 | 8 | 8 | 9 | 9 | 9 | | 9 | 9 | 9 | | 9 | 9 | 9 | |
| | | 805 | 5 | 8 | 8 | 8 | 8 | 8 | 8 | | 8 | 8 | 8 | | 8 | 8 | 8 | |
| | 157 | 403 | 8 | 15 | 15 | 15 | 15 | 15 | 15 | | 15 | 15 | 15 | | 15 | 15 | 15 | |
| | | 470 | 8 | 13 | 13 | 13 | 13 | 13 | 13 | | 13 | 13 | 13 | | 13 | 13 | 13 | |
| | | 564 | 8 | 11 | 11 | 10 | 11 | 11 | 11 | | 11 | 11 | 11 | | 11 | 11 | 11 | |
| | | 705 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | | 8 | 8 | 8 | | 8 | 8 | 8 | |
| | | 805 | 8 | 8 | 7 | 7 | 8 | 8 | 8 | | 8 | 8 | 8 | | 8 | 8 | 8 | |
| | 160 | 403 | 11 | 15 | 15 | 15 | 15 | 15 | 15 | | 15 | 15 | 15 | | 15 | 15 | 15 | |
| | | 470 | 11 | 13 | 13 | 13 | 13 | 13 | 13 | | 13 | 13 | 13 | | 13 | 13 | 13 | |
| | | 564 | 11 | 10 | 11 | 10 | 11 | 11 | 11 | | 11 | 11 | 11 | | 11 | 11 | 11 | |
| | | 705 | 11 | 8 | 8 | 8 | 8 | 8 | 8 | | 8 | 8 | 8 | | 8 | 8 | 8 | |
| | | 805 | 11 | 15 | 15 | 15 | **8** | **8** | **8** | | **8** | **8** | **8** | | **8** | **8** | **8** | |
| Scholl | 299 | 75 | 5 | 52 | 53 | 51 | 51 | 51 | 51 | | 51 | 51 | 51 | | 51 | 51 | 51 | |
| | | 83 | 5 | 45 | 46 | 45 | 46 | 46 | 46 | | 46 | 46 | 46 | | 46 | 46 | 46 | |
| | | 97 | 5 | 42 | 43 | 42 | 42 | 42 | 42 | | 42 | 42 | 42 | | 42 | 42 | 42 | |
| | | 118 | 5 | 35 | 35 | 35 | 35 | 35 | 35 | | 35 | 35 | 35 | | 35 | 35 | 35 | |
| | | 150 | 5 | 26 | 26 | 26 | 26 | 26 | 26 | | 26 | 26 | 26 | | 26 | 26 | 26 | |
| | 302 | 75 | 8 | 52 | 52 | 51 | 51 | 51 | 51 | | 51 | 51 | 51 | | 51 | 51 | 51 | |
| | | 83 | 8 | 45 | 46 | 45 | 45 | 45 | 45 | | 45 | 45 | 45 | | 45 | 45 | 45 | |
| | | 97 | 8 | 42 | 43 | 42 | 42 | 42 | 42 | | 42 | 42 | 42 | | 42 | 42 | 42 | |
| | | 118 | 8 | 35 | 35 | 35 | 35 | 35 | 35 | | 35 | 35 | 35 | | 35 | 35 | 35 | |
| | | 150 | 8 | 26 | 26 | 26 | 26 | 26 | 26 | | 26 | 26 | 26 | | 26 | 26 | 26 | |
| | 305 | 75 | 11 | 51 | 53 | 51 | 51 | 51 | 51 | 36000 | 51 | 51 | 51 | 36000 | 51 | 51 | 51 | 36000 |
| | | 83 | 11 | 45 | 46 | 45 | 45 | 45 | 45 | | 45 | 45 | 45 | | 45 | 45 | 45 | |
| | | 97 | 11 | 42 | 43 | 42 | 42 | 42 | 42 | | 42 | 42 | 42 | | 42 | 42 | 42 | |
| | | 118 | 11 | 35 | 35 | 35 | 35 | 35 | 35 | | 35 | 35 | 35 | | 35 | 35 | 35 | |
| | | 150 | 11 | 26 | 26 | 26 | 26 | 26 | 26 | | 26 | 26 | 26 | | 26 | 26 | 26 | |

When the metaheuristic results of the large-scale problems given in Table 4.10 are compared with the best of the heuristics, an improvement was seen in the Bartholdi problem, which has only 11 subgraphs and a cycle time of 805 seconds among the three metaheuristic algorithms. In the large- scale problem, the best results of the heuristics were achieved at % 88.89 with 40 problems.

In large- scale problems, according to the maxTTS heuristic, there was a %13.33 improvement with 6 problems for the FA, BA and hybrid algorithms, and the best results of this heuristic were % 80. According to the random heuristic run for 600 seconds, there was a %37.78 improvement with 17 problems in the FA, BA and hybrid algorithms, and the best results of this heuristic were % 100.

Lastly, for all medium and large-scale problems, %95 achieved the best heuristic results. There was improvement in 8 problems from 120 problems in FA, BA and hybrid algorithm for minimum values. In addition, all improvement problems' solutions were shown with Appendix 1. In addition, task assignments in detail are given in Appendix 3.

# CHAPTER FIVE
# CONCLUSION

Today, assembly lines are a major component of mass production. Assembly line balancing problems have an important place among optimization problems. In the application of assembly line balancing problems, it is more important to adapt them to real life problems. Therefore, in this thesis, the alternative subgraph assembly line balancing problem type-1 was studied.

With ASALBP, it is aimed to select selection and line balancing of assembly alternatives with different tasks, task times and priority relations. ASALBP-1 aims to minimize the number of workstations with the most suitable mounting alternatives selected. As the size of ASALBP-1 grows, its solution becomes more difficult. In order to improve the solution of the problem, its solution with metaheuristic methods has been proposed.

In this study, metaheuristic solution methods are proposed for the solution of ASALBP-1. It was aimed to expand the search space and increase the working time with metaheuristic solution approaches. Three metaheuristic algorithms were used to solve the problem. Modified firefly and bat algorithms were used, and the hybrid FA-BA algorithm was also proposed for ASALBP-1. Performance evaluations of the algorithms were carried out over 133 benchmark problems with different cycle time and assembly alternatives. The results of FA, BA and hybrid FA-BA algorithms are compared with the best results known in the literature. Medium and large-scale problems compared with the results of 14 heuristics Capacho et al. (2006b).

Overall, the larger the problem data, the larger the solution time, the harder it was to achieve the best results, and the less improvement. For 13 benchmark small problems, BA and proposed hybrid algorithms found better results than FA.

%100 of the best-known results was achieved for 75 benchmark medium – sized problems. The hybrid algorithm found better results than BA and FA in average values. İmprovement was achieved in %9, 33 (7 of 75 problem) of medium- sized problems for best heuristic results.

Improvement and optimizing for large-scale problems gave the same result with FA, BA and hybrid algorithms. The best result of %88, 89 (40 of 45 problem) was achieved in all algorithms. Improvement was achieved in %2.22 (1 of 45 problem) of large-scale problems.

A total of 120 problems of medium and large-scale problems were compared with the best-known heuristic results. %95 (114 of 120 problem) of the best results were achieved. The general improvement was 6.67 (8 of 120 problems).

Table 5.1 Comparison of firefly, bat and hybrid firefly – bat algorithms for best heuristic results

| Algorithms | Minimum values improvement | Maximum values improvement | Average values improvement |
|---|---|---|---|
| Firefly algorithm | %4.17 | %1.67 | %2.75 |
| Bat algorithm | %6.67 | %1.67 | %3.5 |
| Hybrid FA-BA algorithm | %6.67 | %1.67 | %4.17 |

Table 5.1 shows the improvement rates achieved in minimum, maximum and average values with FA, BA and hybrid algorithms in medium and large scale 120 problems. The minimum improvement rate is superior to the FA of bat and hybrid algorithms. BA and hybrid algorithm have equal improvement in minimum improvement. The improvement in maximum is the same for FA, BA and hybrid FA-BA. In average improvement, hybrid algorithm is superior to FA and BA. BA is superior from FA.

## REFERENCES


Baybars, I. (1986). A survey of exact algorithms for the simple assembly line balancing problem. *Management Science, 32,* 909–932.

Battaia, O., & Dolgui, A., (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics, 142*, 259-277.

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research, 168,* 694−715.

Boysen, N., Fliedner, M., & Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research, 183*, 674−693.

Bukchın, J. & Tzur, M. (2000). Designs of flexible assembly line minimize equipment cost. *Institute of Industrial Engineers Transactions, 32,* 585−598.

Baykasoglu, A., & Özsoydan, F.B., (2014). An improved firefly algorithm for solving dynamic multidimensional knapsack problems. *Expert Systems with Applications, 41,* 3712-3725.

Blum, C., & Roli, A., (2008). Hybrid metaheuristic: an emerging approach to optimization. *Studies in Computational İntelligence, 114,* 1-30.

Capacho, L. & Pastor, R. (2005). ASALBP: The alternative subgraphs assembly line balancing problem. Technical Report: IOC−DT−P−2005−5. UPC. Barcelona, Spain.

Capacho, L. & Pastor, R. (2006a). The ASALB problem with processing alternatives involving different tasks: definition, formalization and resolution. *Lecture Notes in Computer Science, 3982*, 554−563.

Capacho, L., Guschinskaya, O., Dolgui, A., & Pastor, R. (2006a). Approximation methods to solve the alternative subgraphs assembly line balancing problem. *Research Report: G2I-EMSE 2006-500-003,* Ecole des Mines, SE, France, April 2006.

Capacho, L., Guschinskaya, O., Dolgui, A., & Pastor, R. (2006b). A comprehensive comparative analysis of heuristic methods for the alternative subgraphs assembly line balancing problem. *Research Report: G2I-EMSE 2006-500-005,* Ecole des Mines de Saint Etienne, France, 2006.

Capacho, L., & Pastor, R. (2006b). Heuristic methods to solve the alternative subgraphs assembly line balancing problem. *Institute Electrical and Electronics Engineers Conference on Automation Science and Engineering,* Shanghai, China, October 7-10, 2006.

Capacho, L., Pastor, R., Dolgui, A., & Guschinskaya, O. (2009). An evaluation of constructive heuristic methods for solving the alternative subgraphs assembly line balancing problem. *Journal of Heuristics, 15*(2), 109–132.

Capacho, L., & Pastor, R, (2011). *A metaheuristic approach to solve the alternative subgraphs assembly line balancing problem.* Retrieved November 28, 2018, from https://www.researchgate.net/publication/221914628.

Das, S. & Nagendra, P., (1997). Selection of routes in a flexible manufacturing facility. *International Journal of Production Economics, 48*, 237-247.

Erel, E. & Sarin, S. (1998). A survey of the assembly line balancing procedures. *Production Planning & Control*, *9*, 414−434.

Fister, I., Fister Jr, I., Yang, X-S., & Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*, *13*, 34-46.

Gandomi, A. H., Yang, X-S., Alavi, & A. H. (2011). Mixed structural optimization using firefly algorithm. *Computers and Structures*, *89*, 2325-2336.

Gandomi A.H., Yang X-S., Alavi A.H., & Talatahari S. (2013). Bat algorithm for constrained optimization tasks. *Neural Computing and Applications*, *22*(6), 1239–1255.

Guo, L., Wang, G-G., Wang, H., & Wang, D. (2013). An effective hybrid firefly algorithm with harmony search for global numerical optimization. *The Scientific World Journal*,*2013*, 1-9.

Helgeson, N. B., & Birnie, D. P., (1961). Assembly line balancing using the ranked positional weight technique. *Journal of Industrial Engineering*, *12*(6), 394-398.

Homem de Mello, L. S., & Sanderson, A. C. (1990). *AND/OR graph representation of assembly plans. Institute Electrical and Electronics Engineers* Transactions on Robotics and Automation, *6*(2), 188–199.

Jati, G. K., & Suyonto (2011). Evolutionary discrete firefly algorithm for travelling salesman problem. *Abdelhamid Bouchachia (Edition), International Conference on Adaptive and Intelligent Systems, Lecture Notes in Computer Science*, *6943*, 393-43.

Koc, A., Sabuncuoglu, I., & Erel, E. (2009). Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph. *Institute of Industrial Engineers Transactions, 41*, 866–881.

Kongkaew, W. (2017). Bat algorithm in discrete optimization: A review of recent applications. *Songklanakarin Journal of Science and Technology*, *39*(5), 641-650.

Kota, L. (2012). Optimization of the supplier selection problem using discrete firefly algorithm. *Advanced Logistic Systems, 6*(1), 117-126.

Li, M., Zhang, Y., Zeng, B., Zhou, H., & Liu, J., (2015). The modified firefly algorithm considering fireflies' visual range and its application in assembly sequence planning. *The International Journal of Advanced Manufacturing Technology*, *82,* 1381-1403.

Luo, Q., Zhou, Y., Xie, J., Ma, M., & Li, L., (2014). Discrete bat algorithm for optimal problem of permutation flow shop scheduling. *The Scientific World Journal, 2014,* 1-15.

Marichelvam, M.K., Prahabaran, T., & Yang, X-S. (2014). A discrete firefly algorithm for the multi-objective hybrid flow shop scheduling problems. *Institute Electrical and Electronics Engineers Transactions on Evolutionary Computation, 18*(2), 301 - 305.

Mirjalili, S., Mirjalili, S. M., & Yang, X-S. (2014). Binary bat algorithm. *Neural Computing and Applications, 25,* 663- 681.

Osaba, E., Carbelledo, R., Yang, X.-S., & Diaz, F. (2016). An evolutionary discrete firefly algorithm with novel operators for solving the vehicle routing problem with time windows. *X-S. Yang (ed.), Nature – Inspired Computation in Engineering, Studies in Computational Intelligence, 637*, 21-41.

Osaba, E., Yang, X.-S., Diaz, F. Lopez-Garcia, P. & Carballedo, R. (2016b). An improved discrete bat algorithm for symmetric and asymmetric traveling salesman. *Engineering Applications of Artificial Intelligence*, 59–71.

Osaba, E., Yang, X.-S. Fister, I., Del Ser, J., Lopez-Garcia, P., Vazquez-Pardavil, et al (2019). A discrete and improved bat algorithm for solving a medical goods distribution problem with pharmacological waste collection. *Swarm and Evolutionary Computation*, 273-286.

Osaba, E., Yang, X-S., Diaz, F., Onieva, E., Masegosa, AD., & Perallas, A. (2017). A discrete firefly algorithm to, solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy. *Soft Computing, 21,* 5295-5308

Park, K., Park, S. & Kim, W. (1997). A heuristic for an assembly line balancing problem with incompatibility, range, and partial precedence constraints. *Computers & Industrial Engineering*, *32*(2), 321−332.

Pinto, P., Dannenbring, D. & Khumawala, B. (1983). Assembly line balancing with processing alternatives: an application. *Management Science*, *29*, 817−830.

Rahmani, A., & MirHassani, S.A. (2014). A hybrid firefly- genetic algorithm for the capacitated facility location problem. *Information Sciences, 283*, 70-78.

Saji, Y., & Riffi, M. E. (2016). A novel discrete bat algorithm for solving the travelling salesman problem. *Neural Computing and Applications, 27*(7), 1853-1866.

Salum, L., & Supciller. A. A. (2008). Rule-based modeling of assembly constraints for line balancing. *Lecture Notes in Artificial Intelligence, 5227*, 783–789.

Sayadi, MK., Ramazanian, R., & Ghaffari, N. (2010). A discrete firefly metaheuristic with local search for makespan minimization in permutation flow shop scheduling problem. *International Journal of Industrial Engineering Computations, 1*, 1-10.

Scholl, A. (1999). *Balancing and sequencing assembly lines* (2nd. Ed.). Heidelberg: *Physica–Verlag.*

Scholl, A., Boysen, N., & Fliedner, M. (2009). Optimally solving the alternative subgraphs assembly line balancing problem. *Annals of Operations Research, 172,* 243–258.

Senin, N., Gropetti, R., & Wallace, D. R. (2000). Concurrent assembly planning with genetic algorithms. *Robotics and Computer Integrated Manufacturing, 16*, 65-72.

Tilahun, S. L., & Ngnotchouye, J. M. T. (2017). Firefly algorithm for discrete optimization problems: a survey. *Korean Society of Civil Engineering Journal of Civil Engineering, 21*(2), 535-545.

Topaloglu, S., Salum, L., & Supciller, A. A. (2012). Rule-based modeling and constraint programming-based solution of the assembly line balancing problem. *Expert Systems with Applications, 39*, 3484–3493.

Yang, X.-S. & He, X. (2013). Bat algorithm: literature review and applications. *International Journal of Bio-Inspired Computation, 5*(3), 141–149.

Yang, X., S. (2008). *Nature – inspired metaheuristic algorithm* (2nd ed.). England: Luniver Press.

Yang, X., S. (2010a). *Engineering optimization an introduction with metaheuristic applications* (1st ed.). New Jersey: A John Wiley & Sons, Inc., Publication.

Yang, X., S. (2010b). A new metaheuristic bat – inspired algorithm. *Computational Intelligence, 284*, 65-74.

Yang, X., S, (2014). Cuckoo search and firefly algorithm theory and applications. *Computational Intelligence, 516*, 12-26.

Zhou, L., Ding, L., Qiang, X., & Luo, Y. (2015). An improved discrete firefly algorithm for the travelling salesman problem. *Journal of Computational and Theoretical Nanoscience*, *12*(7), 1184-1189.

Zhou, Y., Xie, J., & Zheng, H. (2013). A hybrid bat algorithm with path relinking for capacitated vehicle routing problem. *Mathematical Problems in Engineering*, *2013*, 1-10.

Zhou, L., Ding, L., & Qiang, X. (2014). A multi population discrete firefly algorithm to solve tsp. L. Pan et al., (Ed.), *Bio – Inspired Computing- Theories and Applications* (648 -653). New York: Springer.

# APPENDICES

## APPENDIX – 1: Results of Problems with Improvement in Algorithms

Table A.1 Results of problems with improvement in algorithms

| Algorithm | Problem | Cycle time | Number of Sub graphs | Best Heuristic solutions | Minimum | Maximum | Average | CPU |
|---|---|---|---|---|---|---|---|---|
| FIREFLY ALGORITHM | Warnecke | 54 | 11 | 32 | **31** | 32 | **31.1** | 5797.701 |
| | Warnecke | 62 | 5 | 28 | **27** | 28 | **27.7** | 5708.922 |
| | Warnecke | 62 | 8 | 28 | **27** | 28 | **27.9** | 5888.855 |
| | Hann | 4676 | 11 | 8 | **4** | 4 | **4** | 3975.630 |
| | Bartholdi | 805 | 11 | 15 | **8** | 8 | **8** | 18000 |
| BAT ALGORITHM | Warnecke | 54 | 5 | 32 | **31** | 32 | **31.3** | 5834.421 |
| | Warnecke | 62 | 5 | 28 | **27** | 28 | **27.6** | 5839.385 |
| | Warnecke | 54 | 8 | 32 | **31** | 32 | **31.5** | 6251.620 |
| | Warnecke | 62 | 8 | 28 | **27** | 28 | **27.5** | 6256.984 |
| | Warnecke | 62 | 11 | 28 | **27** | 28 | **27.9** | 6342.476 |
| | Hann | 4676 | 11 | 8 | **4** | 4 | **4** | 3851.457 |
| | Bartholdi | 805 | 11 | 15 | **8** | 8 | **8** | 18000 |
| HYBID ALGORITHM | Warnecke | 54 | 5 | 32 | **31** | 32 | **31.3** | 6454.896 |
| | Warnecke | 62 | 5 | 28 | **27** | 28 | **27.3** | 6308.785 |
| | Warnecke | 54 | 8 | 32 | **31** | 32 | **31.4** | 6296.294 |
| | Warnecke | 62 | 8 | 28 | **27** | 28 | **27.4** | 6044.954 |
| | Warnecke | 54 | 11 | 32 | **31** | 32 | **31.5** | 6346.220 |
| | Hann | 4676 | 11 | 8 | **4** | 4 | **4** | 3812.980 |
| | Bartholdi | 805 | 11 | 15 | **8** | 8 | **8** | 18000 |

## APPENDIX – 2: Results of Heuristic Algorithms

Table A.2 Results of heuristic algorithms

| Problem name | Cycle time | Subgraph numbers | Best solution | Heuristic solution methods | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Maximum rank positional weight | Maximum task time | Minimum earliest workstation | Maximum latest workstation | Minimum task number | Minimum slack | Min. tasks time divided by Latest workstation | Max. number of immediate successors | Max. number of total successors | Max. task time plus total number of successors | Max. successors time / total successors | Max. number of total successors / slack | Min. latest workstation / total successors | Random _1 second | Random _5 seconds | Random _25 seconds | Random _60 seconds | Random _600 seconds |
| Gunther | 7 | 5 | 14 | 15 | 14 | 15 | 15 | 16 | 15 | 14 | 15 | 15 | 14 | 14 | 16 | 15 | 14 | 14 | 14 | 14 | 14 |
| | | 8 | 14 | 15 | 14 | 15 | 15 | 16 | 15 | 14 | 15 | 15 | 14 | 14 | 16 | 15 | 14 | 14 | 14 | 14 | 14 |
| | | 11 | 14 | 14 | 14 | 15 | 14 | 16 | 14 | 14 | 15 | 14 | 14 | 14 | 16 | 14 | 14 | 14 | 14 | 14 | 14 |
| | 44 | 5 | 12 | 13 | 12 | 13 | 13 | 13 | 13 | 12 | 14 | 13 | 12 | 13 | 13 | 13 | 12 | 12 | 12 | 12 | 12 |
| | | 8 | 12 | 13 | 12 | 13 | 13 | 13 | 13 | 12 | 14 | 13 | 12 | 13 | 13 | 13 | 12 | 12 | 12 | 12 | 12 |
| | | 11 | 12 | 12 | 12 | 13 | 12 | 13 | 13 | 12 | 14 | 13 | 12 | 13 | 13 | 13 | 12 | 12 | 12 | 12 | 12 |
| | 49 | 5 | 11 | 12 | 11 | 12 | 11 | 12 | 11 | 11 | 12 | 12 | 11 | 11 | 12 | 12 | 11 | 11 | 11 | 11 | 11 |
| | | 8 | 11 | 12 | 11 | 12 | 11 | 12 | 11 | 11 | 12 | 12 | 11 | 11 | 12 | 12 | 11 | 11 | 11 | 11 | 11 |
| | | 11 | 11 | 12 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 12 | 11 | 11 | 11 | 12 | 11 | 11 | 11 | 11 | 11 |
| | 61 | 5 | 9 | 9 | 9 | 10 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
| | | 8 | 9 | 9 | 9 | 10 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
| | | 11 | 9 | 9 | 9 | 10 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
| | 81 | 5 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| | | 8 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| | | 11 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| Kilbrid | 57 | 5 | 10 | 10 | 10 | 11 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 11 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| | | 8 | 10 | 10 | 10 | 11 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| | | 11 | 10 | 10 | 10 | 11 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| | 79 | 5 | 7 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 7 | 7 | 7 |
| | | 8 | 7 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 7 | 7 | 7 | 7 |
| | | 11 | 7 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 7 | 7 | 7 |
| | 92 | 5 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 6 | 6 |
| | | 8 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 6 | 6 | 7 | 6 | 6 |
| | | 11 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 6 |
| | 138 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 |
| | | 8 | 4 | 4 | 5 | 5 | 4 | 4 | 5 | 5 | 5 | 5 | 4 | 5 | 4 | 5 | 4 | 4 | 4 | 4 | 4 |
| | | 11 | 4 | 4 | 5 | 5 | 4 | 4 | 5 | 5 | 5 | 5 | 4 | 5 | 4 | 5 | 4 | 4 | 4 | 4 | 4 |
| | 184 | 5 | 3 | 4 | 3 | 4 | 4 | 4 | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 |
| | | 8 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 |
| | | 11 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 |

Table A.2 continues

| Problem name | Cycle time | Subgraph numbers | Best solution | Heuristic solution methods | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Maximum rank positional weight | Maximum task time | Minimum earliest workstation | Maximum latest workstation | Minimum task number | Minimum slack | Min. tasks time divided by Latest workstation | Max. number of immediate successors | Max. number of total successors | Max. task time plus total number of successors | Max. successors time divided by total successors | Max. number of total successors divided by slack | Min. latest workstation divided by total successors | Random _1 second | Random _5 seconds | Random _25 seconds | Random _60 seconds | Random _600 seconds |
| Hann | 2004 | 5 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| | | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| | | 11 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| | 2338 | 5 | 7 | 7 | 8 | 8 | 7 | 8 | 7 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| | | 8 | 7 | 7 | 8 | 8 | 7 | 8 | 7 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 7 |
| | | 11 | 7 | 7 | 8 | 8 | 7 | 8 | 7 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 7 |
| | 2806 | 5 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| | | 8 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| | | 11 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| | 3507 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| | | 8 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| | | 11 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| | 4676 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| | | 8 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| | | 11 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| Warnecke | 54 | 5 | 32 | 33 | 34 | 35 | 32 | 35 | 34 | 32 | 35 | 33 | 32 | 34 | 35 | 32 | 34 | 34 | 33 | 33 | 33 |
| | | 8 | 32 | 33 | 34 | 35 | 32 | 35 | 34 | 32 | 35 | 32 | 32 | 34 | 35 | 32 | 34 | 33 | 33 | 33 | 33 |
| | | 11 | 32 | 33 | 33 | 35 | 32 | 35 | 34 | 32 | 35 | 32 | 32 | 34 | 35 | 32 | 33 | 33 | 33 | 33 | 32 |
| | 62 | 5 | 28 | 29 | 29 | 31 | 29 | 30 | 30 | 29 | 30 | 29 | 28 | 29 | 30 | 29 | 29 | 28 | 28 | 28 | 28 |
| | | 8 | 28 | 29 | 29 | 31 | 29 | 30 | 30 | 29 | 30 | 29 | 28 | 29 | 30 | 29 | 29 | 29 | 28 | 28 | 28 |
| | | 11 | 28 | 29 | 29 | 31 | 29 | 30 | 30 | 29 | 30 | 29 | 28 | 29 | 30 | 29 | 29 | 28 | 28 | 28 | 28 |
| | 74 | 5 | 22 | 23 | 24 | 25 | 23 | 25 | 24 | 23 | 24 | 23 | 22 | 24 | 25 | 23 | 23 | 23 | 23 | 23 | 23 |
| | | 8 | 22 | 23 | 24 | 25 | 23 | 25 | 24 | 23 | 24 | 23 | 22 | 24 | 25 | 23 | 23 | 23 | 23 | 23 | 23 |
| | | 11 | 22 | 23 | 24 | 25 | 23 | 25 | 24 | 23 | 24 | 23 | 22 | 24 | 25 | 23 | 23 | 24 | 23 | 23 | 23 |
| | 92 | 5 | 18 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 18 | 19 | 18 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 18 |
| | | 8 | 18 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 18 | 19 | 18 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 19 |
| | | 11 | 18 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 18 | 19 | 18 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 19 |
| | 111 | 5 | 15 | 15 | 15 | 16 | 15 | 15 | 16 | 15 | 16 | 15 | 15 | 16 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| | | 8 | 15 | 15 | 15 | 16 | 15 | 15 | 16 | 15 | 16 | 15 | 15 | 16 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| | | 11 | 15 | 15 | 15 | 16 | 15 | 15 | 16 | 15 | 15 | 15 | 15 | 16 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |

Table A.2 continues

| Problem name | Cycle time | Subgraph numbers | Best solution | Heuristic solution methods | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Maximum rank positional weight | Maximum task time | Minimum earliest workstation | Maximum latest workstation | Minimum task number | Minimum slack | Min. tasks time divided by Latest workstation | Max. number of immediate successors | Max. number of total successors | Max. task time plus total number of successors | Max. successors time divided by total successors | Max. number of total successors divided by slack | Min. latest workstation divided by total successors | Random _1 second | Random _5 seconds | Random _25 seconds | Random _60 seconds | Random _600 seconds |
| Tongue | 160 | 5 | 23 | 23 | 23 | 26 | 24 | 25 | 24 | 23 | 25 | 23 | 24 | 25 | 25 | 23 | 24 | 23 | 23 | 23 | 23 |
| | | 8 | 23 | 23 | 23 | 26 | 24 | 25 | 24 | 23 | 25 | 23 | 24 | 25 | 25 | 23 | 24 | 24 | 23 | 23 | 23 |
| | | 11 | 23 | 23 | 23 | 26 | 24 | 25 | 24 | 23 | 25 | 23 | 24 | 25 | 25 | 23 | 24 | 24 | 24 | 23 | 23 |
| | 176 | 5 | 21 | 22 | 21 | 23 | 22 | 23 | 22 | 21 | 23 | 22 | 21 | 22 | 23 | 22 | 22 | 22 | 22 | 22 | 22 |
| | | 8 | 21 | 22 | 21 | 23 | 22 | 23 | 22 | 21 | 23 | 22 | 21 | 22 | 23 | 22 | 22 | 22 | 22 | 22 | 21 |
| | | 11 | 21 | 21 | 21 | 23 | 22 | 23 | 22 | 21 | 23 | 22 | 21 | 22 | 23 | 22 | 22 | 22 | 22 | 22 | 21 |
| | 207 | 5 | 18 | 18 | 18 | 19 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 19 | 18 | 18 | 18 | 18 | 18 | 18 | 18 |
| | | 8 | 18 | 18 | 18 | 19 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 19 | 18 | 18 | 18 | 18 | 18 | 18 | 18 |
| | | 11 | 18 | 18 | 18 | 19 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 19 | 18 | 18 | 18 | 18 | 18 | 18 | 18 |
| | 251 | 5 | 15 | 15 | 15 | 16 | 15 | 15 | 15 | 15 | 15 | 15 | 16 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| | | 8 | 15 | 15 | 15 | 16 | 15 | 15 | 15 | 15 | 15 | 15 | 16 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| | | 11 | 15 | 15 | 15 | 16 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| | 320 | 5 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| | | 8 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| | | 11 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| Arcus | 5785 | 5 | 27 | 27 | 27 | 30 | 28 | 29 | 28 | 27 | 29 | 28 | 27 | 27 | 29 | 28 | 29 | 28 | 28 | 28 | 28 |
| | | 8 | 27 | 27 | 27 | 30 | 27 | 29 | 28 | 27 | 29 | 28 | 27 | 27 | 29 | 28 | 28 | 28 | 28 | 28 | 28 |
| | | 11 | 27 | 27 | 27 | 30 | 27 | 29 | 28 | 27 | 28 | 28 | 27 | 27 | 29 | 28 | 29 | 29 | 28 | 28 | 28 |
| | 6540 | 5 | 24 | 24 | 25 | 27 | 24 | 27 | 24 | 25 | 27 | 24 | 25 | 25 | 27 | 24 | 25 | 25 | 25 | 25 | 25 |
| | | 8 | 24 | 24 | 25 | 27 | 24 | 27 | 24 | 25 | 27 | 24 | 25 | 25 | 27 | 24 | 26 | 25 | 25 | 25 | 25 |
| | | 11 | 24 | 24 | 25 | 27 | 24 | 27 | 24 | 25 | 26 | 24 | 25 | 25 | 27 | 24 | 25 | 25 | 25 | 25 | 25 |
| | 7916 | 5 | 20 | 20 | 20 | 23 | 20 | 22 | 20 | 20 | 22 | 20 | 20 | 20 | 22 | 20 | 21 | 21 | 21 | 21 | 20 |
| | | 8 | 20 | 20 | 20 | 23 | 20 | 22 | 20 | 20 | 22 | 20 | 20 | 20 | 22 | 20 | 21 | 21 | 21 | 21 | 21 |
| | | 11 | 20 | 20 | 20 | 23 | 20 | 22 | 20 | 20 | 22 | 20 | 20 | 20 | 22 | 20 | 22 | 21 | 21 | 21 | 20 |
| | 9400 | 5 | 17 | 17 | 17 | 18 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 |
| | | 8 | 17 | 17 | 17 | 18 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 |
| | | 11 | 17 | 17 | 17 | 18 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 |
| | 11570 | 5 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 |
| | | 8 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 |
| | | 11 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 |

Table A.2 continues

| Problem name | Cycle time | Subgraph numbers | Best solution | Heuristic solution methods | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Maximum rank positional weight | Maximum task time | Minimum earliest workstation | Maximum latest workstation | Minimum task number | Minimum slack | Min. tasks time divided by Latest workstation | Max. number of immediate successors | Max. number of total successors | Max. task time plus total number of successors | Max. successors time divided by total successors | Max. number of total successors divided by slack | Min. latest workstation divided by total successors | Random _1 second | Random _5 seconds | Random _25 seconds | Random _60 seconds | Random _600 seconds |
| Bartholdi | 403 | 5 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| | | 8 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| | | 11 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| | 470 | 5 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 |
| | | 8 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 |
| | | 11 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 |
| | 564 | 5 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| | | 8 | 10 | 11 | 11 | 11 | 11 | 11 | 11 | 10 | 11 | 11 | 11 | 10 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| | | 11 | 10 | 11 | 11 | 11 | 11 | 11 | 11 | 10 | 11 | 11 | 10 | 10 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| | 705 | 5 | 8 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 8 | 9 | 9 | 9 | 9 | 9 | 8 | 8 |
| | | 8 | 8 | 8 | 9 | 9 | 9 | 9 | 9 | 8 | 9 | 8 | 8 | 8 | 9 | 9 | 9 | 9 | 9 | 9 | 8 |
| | | 11 | 8 | 8 | 9 | 9 | 9 | 9 | 9 | 8 | 8 | 8 | 8 | 8 | 9 | 9 | 9 | 9 | 9 | 8 | 8 |
| | 805 | 5 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| | | 8 | 7 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 7 | 8 | 8 | 8 | 7 | 8 | 8 | 8 | 8 | 7 |
| | | 11 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| Scholl | 1394 | 5 | 51 | 51 | 52 | 54 | 52 | 53 | 52 | 53 | 52 | 52 | 52 | 54 | 53 | 52 | 53 | 53 | 53 | 53 | 52 |
| | | 8 | 51 | 51 | 52 | 54 | 52 | 53 | 52 | 52 | 52 | 52 | 52 | 53 | 53 | 52 | 53 | 53 | 53 | 53 | 53 |
| | | 11 | 51 | 51 | 52 | 54 | 52 | 53 | 52 | 52 | 52 | 52 | 51 | 53 | 53 | 52 | 53 | 53 | 53 | 53 | 53 |
| | 1584 | 5 | 45 | 46 | 46 | 47 | 46 | 46 | 46 | 46 | 46 | 46 | 45 | 48 | 46 | 46 | 46 | 46 | 47 | 46 | 46 |
| | | 8 | 45 | 46 | 46 | 47 | 46 | 46 | 46 | 46 | 46 | 46 | 45 | 47 | 46 | 46 | 47 | 47 | 46 | 46 | 46 |
| | | 11 | 45 | 45 | 45 | 47 | 46 | 46 | 46 | 45 | 46 | 46 | 45 | 46 | 46 | 46 | 47 | 46 | 46 | 46 | 46 |
| | 1699 | 5 | 42 | 42 | 43 | 44 | 42 | 43 | 43 | 42 | 42 | 43 | 42 | 43 | 43 | 43 | 44 | 44 | 43 | 43 | 43 |
| | | 8 | 42 | 42 | 43 | 44 | 42 | 43 | 43 | 42 | 42 | 42 | 42 | 43 | 43 | 42 | 44 | 44 | 43 | 43 | 43 |
| | | 11 | 42 | 42 | 43 | 44 | 42 | 43 | 43 | 42 | 42 | 42 | 42 | 43 | 43 | 42 | 44 | 44 | 43 | 43 | 43 |
| | 2049 | 5 | 35 | 35 | 35 | 36 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 36 | 35 | 35 | 37 | 36 | 35 | 35 | 35 |
| | | 8 | 35 | 35 | 35 | 36 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 37 | 36 | 35 | 35 | 35 |
| | | 11 | 35 | 35 | 35 | 36 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 36 | 36 | 35 | 35 | 35 |
| | 2787 | 5 | 26 | 26 | 26 | 27 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 27 | 26 | 26 | 26 | 26 | 26 | 26 | 26 |
| | | 8 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 |
| | | 11 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 |

## APPENDIX – 3: Detailed Solutions of Improving Problems

Table A.3 Results of Hann problem with 3 sub-assembly line number and cycle time 4676

| Sub assembly number | 3 | | |
|---|---|---|---|
| Subgraph numbers | 5-8-10 | | |
| Problem name- cycle time | Hann- 4676 | | |
| Task order | Workstation | Task time | Cumulative task time |
| 1 | 1 | 971 | 971 |
| 54 | 1 | 660 | 1631 |
| 55 | 1 | 660 | 2291 |
| 56 | 1 | 667 | 2958 |
| 11 | 1 | 82 | 3040 |
| 12 | 1 | 60 | 3100 |
| 13 | 1 | 112 | 3212 |
| 15 | 2 | 1556 | 1556 |
| 17 | 2 | 259 | 1815 |
| 14 | 2 | 420 | 2235 |
| 19 | 2 | 601 | 2836 |
| 22 | 2 | 70 | 2906 |
| 25 | 2 | 105 | 3011 |
| 20 | 2 | 80 | 3091 |
| 23 | 2 | 89 | 3180 |
| 26 | 2 | 330 | 3510 |
| 27 | 2 | 132 | 3642 |
| 28 | 2 | 69 | 3711 |
| 16 | 2 | 236 | 3947 |
| 21 | 2 | 80 | 4027 |
| 29 | 2 | 99 | 4126 |
| 24 | 2 | 89 | 4215 |
| 31 | 2 | 70 | 4285 |
| 34 | 2 | 70 | 4355 |
| 33 | 2 | 191 | 4546 |
| 30 | 2 | 70 | 4616 |
| 2 | 3 | 142 | 142 |
| 32 | 3 | 158 | 300 |
| 3 | 3 | 142 | 442 |
| 35 | 3 | 53 | 495 |
| 36 | 3 | 50 | 545 |
| 37 | 3 | 125 | 670 |
| 38 | 3 | 353 | 1023 |
| 39 | 3 | 70 | 1093 |

Table A.3 continues

| | | | |
|---|---|---|---|
| 40 | 3 | 128 | 1221 |
| 41 | 3 | 65 | 1286 |
| 18 | 3 | 125 | 1411 |
| 57 | 3 | 1800 | 3211 |
| 58 | 3 | 300 | 3511 |
| 59 | 3 | 300 | 3811 |
| 60 | 3 | 300 | 4111 |
| 61 | 3 | 300 | 4411 |
| 62 | 4 | 900 | 900 |
| 63 | 4 | 800 | 1700 |

Table A.4 Results of Warnecke problem with 1 sub-assembly line number and cycle time 54

| Sub assembly number | 1 | | |
|---|---|---|---|
| Subgraph | 4 | | |
| Problem name - cycle time | Warnecke - 54 | | |
| Task order | Workstation | Task time | Cumulative task time |
| 5 | 1 | 35 | 35 |
| 28 | 1 | 12 | 47 |
| 1 | 2 | 10 | 10 |
| 3 | 2 | 41 | 51 |
| 22 | 3 | 37 | 37 |
| 13 | 3 | 12 | 49 |
| 6 | 4 | 17 | 17 |
| 16 | 4 | 33 | 50 |
| 14 | 5 | 52 | 52 |
| 18 | 6 | 7 | 7 |
| 12 | 6 | 34 | 41 |
| 15 | 6 | 12 | 53 |
| 9 | 7 | 14 | 14 |
| 11 | 7 | 33 | 47 |
| 33 | 8 | 34 | 34 |
| 19 | 8 | 15 | 49 |
| 2 | 9 | 53 | 53 |
| 17 | 10 | 44 | 44 |
| 32 | 11 | 47 | 47 |
| 27 | 12 | 16 | 16 |
| 20 | 12 | 13 | 29 |
| 30 | 12 | 22 | 51 |
| 21 | 13 | 29 | 29 |
| 8 | 13 | 23 | 52 |

Table A.4 continues

| | | | |
|---|---|---|---|
| 23 | 14 | 43 | 43 |
| 26 | 14 | 9 | 52 |
| 25 | 15 | 24 | 24 |
| 24 | 15 | 23 | 47 |
| 29 | 16 | 26 | 26 |
| 34 | 16 | 23 | 49 |
| 31 | 17 | 51 | 51 |
| 36 | 18 | 52 | 52 |
| 44 | 19 | 37 | 37 |
| 37 | 19 | 12 | 49 |
| 7 | 20 | 34 | 34 |
| 41 | 20 | 15 | 49 |
| 40 | 21 | 7 | 7 |
| 39 | 21 | 44 | 51 |
| 42 | 22 | 13 | 13 |
| 35 | 22 | 12 | 25 |
| 43 | 22 | 29 | 54 |
| 45 | 23 | 43 | 43 |
| 46 | 24 | 23 | 23 |
| 49 | 24 | 16 | 39 |
| 50 | 24 | 12 | 51 |
| 47 | 25 | 24 | 24 |
| 51 | 25 | 26 | 50 |
| 38 | 26 | 33 | 33 |
| 52 | 26 | 12 | 45 |
| 48 | 26 | 9 | 54 |
| 53 | 27 | 52 | 52 |
| 54 | 28 | 12 | 12 |
| 55 | 28 | 33 | 45 |
| 56 | 29 | 44 | 44 |
| 57 | 29 | 7 | 51 |
| 58 | 30 | 15 | 15 |
| 4 | 30 | 36 | 51 |
| 10 | 31 | 52 | 52 |

Table A.5 Results of Warnecke problem with 1 sub-assembly line number and cycle time 62

| Sub assembly number | 1 | | |
|---|---|---|---|
| Subgraph number | 3 | | |
| Problem name - cycle time | Warnecke - 62 | | |
| Task order | Workstation | Task time | Cumulative task time |
| 3 | 1 | 41 | 41 |
| 6 | 1 | 17 | 58 |
| 1 | 2 | 10 | 10 |
| 22 | 2 | 37 | 47 |
| 9 | 2 | 14 | 61 |
| 27 | 3 | 16 | 16 |
| 28 | 3 | 12 | 28 |
| 11 | 3 | 33 | 61 |
| 24 | 4 | 23 | 23 |
| 12 | 4 | 34 | 57 |
| 13 | 5 | 12 | 12 |
| 15 | 5 | 12 | 24 |
| 16 | 5 | 33 | 57 |
| 2 | 6 | 53 | 53 |
| 14 | 7 | 52 | 52 |
| 18 | 7 | 7 | 59 |
| 17 | 8 | 44 | 44 |
| 19 | 8 | 15 | 59 |
| 20 | 9 | 13 | 13 |
| 23 | 9 | 43 | 56 |
| 30 | 10 | 22 | 22 |
| 33 | 10 | 34 | 56 |
| 21 | 11 | 29 | 29 |
| 26 | 11 | 9 | 38 |
| 34 | 11 | 23 | 61 |
| 5 | 12 | 35 | 35 |
| 29 | 12 | 26 | 61 |
| 31 | 13 | 51 | 51 |
| 36 | 14 | 52 | 52 |
| 32 | 15 | 47 | 47 |
| 37 | 15 | 12 | 59 |
| 7 | 16 | 34 | 34 |
| 41 | 16 | 15 | 49 |
| 42 | 16 | 13 | 62 |
| 39 | 17 | 44 | 44 |

Table A.5 continues

| | | | |
|---|---|---|---|
| 35 | 17 | 12 | 56 |
| 8 | 18 | 23 | 23 |
| 44 | 18 | 37 | 60 |
| 40 | 19 | 7 | 7 |
| 43 | 19 | 29 | 36 |
| 25 | 19 | 24 | 60 |
| 45 | 20 | 43 | 43 |
| 49 | 20 | 16 | 59 |
| 46 | 21 | 23 | 23 |
| 47 | 21 | 24 | 47 |
| 50 | 21 | 12 | 59 |
| 38 | 22 | 33 | 33 |
| 51 | 22 | 26 | 59 |
| 4 | 23 | 36 | 36 |
| 52 | 23 | 12 | 48 |
| 53 | 24 | 52 | 52 |
| 48 | 24 | 9 | 61 |
| 54 | 25 | 12 | 12 |
| 56 | 25 | 44 | 56 |
| 55 | 26 | 33 | 33 |
| 57 | 26 | 7 | 40 |
| 58 | 26 | 15 | 55 |
| 10 | 27 | 52 | 52 |

Table A.6 Results of Warnecke problem with 2 sub-assembly line number and cycle time 54

| Sub assembly number | 2 | | |
|---|---|---|---|
| Subgraph numbers | 4 -7 | | |
| Problem name - cycle time | Warnecke - 54 | | |
| Task order | Workstation | Task time | Cumulative task time |
| 7 | 1 | 34 | 34 |
| 1 | 1 | 10 | 44 |
| 3 | 2 | 41 | 41 |
| 15 | 2 | 12 | 53 |
| 11 | 3 | 33 | 33 |
| 13 | 3 | 12 | 45 |
| 16 | 4 | 33 | 33 |
| 18 | 4 | 7 | 40 |
| 28 | 4 | 12 | 52 |

Table A.6 continues

| | | | |
|---|---|---|---|
| 22 | 5 | 37 | 37 |
| 27 | 5 | 16 | 53 |
| 5 | 6 | 35 | 35 |
| 19 | 6 | 15 | 50 |
| 14 | 7 | 52 | 52 |
| 32 | 8 | 47 | 47 |
| 12 | 9 | 34 | 34 |
| 9 | 9 | 14 | 48 |
| 17 | 10 | 44 | 44 |
| 20 | 11 | 13 | 13 |
| 6 | 11 | 17 | 30 |
| 30 | 11 | 22 | 52 |
| 24 | 12 | 23 | 23 |
| 21 | 12 | 29 | 52 |
| 23 | 13 | 43 | 43 |
| 26 | 13 | 9 | 52 |
| 34 | 14 | 23 | 23 |
| 29 | 14 | 26 | 49 |
| 31 | 15 | 51 | 51 |
| 36 | 16 | 52 | 52 |
| 2 | 17 | 53 | 53 |
| 37 | 18 | 12 | 12 |
| 33 | 18 | 34 | 46 |
| 41 | 19 | 15 | 15 |
| 8 | 19 | 23 | 38 |
| 35 | 19 | 12 | 50 |
| 44 | 20 | 37 | 37 |
| 42 | 20 | 13 | 50 |
| 39 | 21 | 44 | 44 |
| 40 | 21 | 7 | 51 |
| 43 | 22 | 29 | 29 |
| 25 | 22 | 24 | 53 |
| 45 | 23 | 43 | 43 |
| 48 | 23 | 9 | 52 |
| 46 | 24 | 23 | 23 |
| 47 | 24 | 24 | 47 |
| 51 | 25 | 26 | 26 |

Table A.6 continues

| | | | |
|---|---|---|---|
| 49 | 25 | 16 | 42 |
| 50 | 25 | 12 | 54 |
| 38 | 26 | 33 | 33 |
| 52 | 26 | 12 | 45 |
| 53 | 27 | 52 | 52 |
| 54 | 28 | 12 | 12 |
| 55 | 28 | 33 | 45 |
| 56 | 29 | 44 | 44 |
| 57 | 29 | 7 | 51 |
| 4 | 30 | 36 | 36 |
| 58 | 30 | 15 | 51 |
| 10 | 31 | 52 | 52 |

Table A.7 Results of Warnecke problem with 2 sub-assembly line number and cycle time 62

| Sub assembly numbers | 2 | | |
|---|---|---|---|
| Subgraph numbers | 4 - 6 | | |
| Problem name - cycle time | Warnecke - 62 | | |
| Task order | Workstation | Task time | Cumulative task time |
| 1 | 1 | 10 | 10 |
| 14 | 1 | 52 | 62 |
| 3 | 2 | 41 | 41 |
| 6 | 2 | 17 | 58 |
| 24 | 3 | 23 | 23 |
| 12 | 3 | 34 | 57 |
| 13 | 4 | 12 | 12 |
| 15 | 4 | 12 | 24 |
| 22 | 4 | 37 | 61 |
| 11 | 5 | 33 | 33 |
| 28 | 5 | 12 | 45 |
| 9 | 5 | 14 | 59 |
| 17 | 6 | 44 | 44 |
| 20 | 6 | 13 | 57 |
| 16 | 7 | 33 | 33 |
| 21 | 7 | 29 | 62 |
| 2 | 8 | 53 | 53 |
| 18 | 8 | 7 | 60 |
| 27 | 9 | 16 | 16 |
| 30 | 9 | 22 | 38 |
| 8 | 9 | 23 | 61 |

Table A.7 continues

| | | | |
|---|---|---|---|
| 26 | 10 | 9 | 9 |
| 44 | 10 | 37 | 46 |
| 19 | 10 | 15 | 61 |
| 23 | 11 | 43 | 43 |
| 5 | 12 | 35 | 35 |
| 34 | 12 | 23 | 58 |
| 33 | 13 | 34 | 34 |
| 29 | 13 | 26 | 60 |
| 31 | 14 | 51 | 51 |
| 36 | 15 | 52 | 52 |
| 32 | 16 | 47 | 47 |
| 37 | 16 | 12 | 59 |
| 25 | 17 | 24 | 24 |
| 7 | 17 | 34 | 58 |
| 39 | 18 | 44 | 44 |
| 41 | 18 | 15 | 59 |
| 42 | 19 | 13 | 13 |
| 35 | 19 | 12 | 25 |
| 40 | 19 | 7 | 32 |
| 43 | 19 | 29 | 61 |
| 45 | 20 | 43 | 43 |
| 49 | 20 | 16 | 59 |
| 4 | 21 | 36 | 36 |
| 46 | 21 | 23 | 59 |
| 50 | 22 | 12 | 12 |
| 47 | 22 | 24 | 36 |
| 51 | 22 | 26 | 62 |
| 48 | 23 | 9 | 9 |
| 38 | 23 | 33 | 42 |
| 52 | 23 | 12 | 54 |
| 53 | 24 | 52 | 52 |
| 54 | 25 | 12 | 12 |
| 56 | 25 | 44 | 56 |
| 55 | 26 | 33 | 33 |
| 57 | 26 | 7 | 40 |
| 58 | 26 | 15 | 55 |
| 10 | 27 | 52 | 52 |

Table A.8 Results of Warnecke problem with 3 sub-assembly line number and cycle time 54

| Sub assembly number | 3 | | |
|---|---|---|---|
| Subgraph numbers | 5- 6- 9 | | |
| Problem name - cycle time | Warnecke - 54 | | |
| Task order | Workstation | Task time | Cumulative task time |
| 2 | 1 | 53 | 53 |
| 28 | 2 | 12 | 12 |
| 3 | 2 | 41 | 53 |
| 22 | 3 | 37 | 37 |
| 27 | 3 | 16 | 53 |
| 5 | 4 | 35 | 35 |
| 6 | 4 | 17 | 52 |
| 32 | 5 | 47 | 47 |
| 16 | 6 | 33 | 33 |
| 18 | 6 | 7 | 40 |
| 20 | 6 | 13 | 53 |
| 24 | 7 | 23 | 23 |
| 8 | 7 | 23 | 46 |
| 33 | 8 | 34 | 34 |
| 19 | 8 | 15 | 49 |
| 30 | 9 | 22 | 22 |
| 21 | 9 | 29 | 51 |
| 26 | 10 | 9 | 9 |
| 23 | 10 | 43 | 52 |
| 34 | 11 | 23 | 23 |
| 29 | 11 | 26 | 49 |
| 31 | 12 | 51 | 51 |
| 36 | 13 | 52 | 52 |
| 37 | 14 | 12 | 12 |
| 59 | 14 | 42 | 54 |
| 41 | 15 | 15 | 15 |
| 7 | 15 | 34 | 49 |
| 44 | 16 | 37 | 37 |
| 42 | 16 | 13 | 50 |
| 39 | 17 | 44 | 44 |
| 35 | 18 | 12 | 12 |
| 40 | 18 | 7 | 19 |
| 43 | 18 | 29 | 48 |
| 45 | 19 | 43 | 43 |

Table A.8 continues

| | | | |
|---|---|---|---|
| 63 | 20 | 43 | 43 |
| 46 | 21 | 23 | 23 |
| 25 | 21 | 24 | 47 |
| 47 | 22 | 24 | 24 |
| 51 | 22 | 26 | 50 |
| 50 | 23 | 12 | 12 |
| 62 | 23 | 42 | 54 |
| 48 | 24 | 9 | 9 |
| 38 | 24 | 33 | 42 |
| 52 | 24 | 12 | 54 |
| 53 | 25 | 52 | 52 |
| 54 | 26 | 12 | 12 |
| 61 | 26 | 42 | 54 |
| 60 | 27 | 42 | 42 |
| 49 | 28 | 16 | 16 |
| 55 | 28 | 33 | 49 |
| 56 | 29 | 44 | 44 |
| 57 | 29 | 7 | 51 |
| 58 | 30 | 15 | 15 |
| 4 | 30 | 36 | 51 |
| 10 | 31 | 52 | 52 |

Table A.9 Results of Warnecke problem with 3 sub-assembly line number and cycle time 62

| Sub assembly number | 3 | | |
|---|---|---|---|
| Subgraph numbers | 2-7-10 | | |
| Problem name - cycle time | Warnecke - 62 | | |
| Task order | Workstation | Task time | Cumulative task time |
| 28 | 1 | 12 | 12 |
| 6 | 1 | 17 | 29 |
| 16 | 1 | 33 | 62 |
| 18 | 2 | 7 | 7 |
| 2 | 2 | 53 | 60 |
| 1 | 3 | 10 | 10 |
| 3 | 3 | 41 | 51 |
| 22 | 4 | 37 | 37 |
| 24 | 4 | 23 | 60 |

Table A.9 continues

| | | | |
|---|---|---|---|
| 9 | 5 | 14 | 14 |
| 19 | 5 | 15 | 29 |
| 11 | 5 | 33 | 62 |
| 14 | 6 | 52 | 52 |
| 23 | 7 | 43 | 43 |
| 15 | 7 | 12 | 55 |
| 12 | 8 | 34 | 34 |
| 13 | 8 | 12 | 46 |
| 27 | 8 | 16 | 62 |
| 17 | 9 | 44 | 44 |
| 20 | 9 | 13 | 57 |
| 21 | 10 | 29 | 29 |
| 26 | 10 | 9 | 38 |
| 30 | 10 | 22 | 60 |
| 33 | 11 | 34 | 34 |
| 29 | 11 | 26 | 60 |
| 5 | 12 | 35 | 35 |
| 34 | 12 | 23 | 58 |
| 35 | 13 | 12 | 12 |
| 32 | 13 | 47 | 59 |
| 31 | 14 | 51 | 51 |
| 36 | 15 | 52 | 52 |
| 7 | 16 | 34 | 34 |
| 37 | 16 | 12 | 46 |
| 41 | 16 | 15 | 61 |
| 42 | 17 | 13 | 13 |
| 39 | 17 | 44 | 57 |
| 8 | 18 | 23 | 23 |
| 44 | 18 | 37 | 60 |
| 40 | 19 | 7 | 7 |
| 25 | 19 | 24 | 31 |
| 43 | 19 | 29 | 60 |
| 45 | 20 | 43 | 43 |
| 49 | 20 | 16 | 59 |
| 38 | 21 | 33 | 33 |
| 46 | 21 | 23 | 56 |
| 4 | 22 | 36 | 36 |

Table A.9 continues

| | | | |
|---|---|---|---|
| 47 | 22 | 24 | 60 |
| 10 | 23 | 52 | 52 |
| 48 | 23 | 9 | 61 |
| 64 | 24 | 50 | 50 |
| 65 | 25 | 50 | 50 |
| 66 | 26 | 50 | 50 |
| 50 | 26 | 12 | 62 |
| 67 | 27 | 43 | 43 |
| 58 | 27 | 15 | 58 |

Table A.10 Results of Bartholdi problem with 3 sub-assembly line number and cycle time 805

| Sub assembly number | 3 | | |
|---|---|---|---|
| Subgraph number | 2 - 8 - 9 | | |
| Problem name - cycle time | Bartholdi - 805 | | |
| Task number | Workstation | Task time | Cumulative task time |
| 2 | 1 | 30 | 30 |
| 3 | 1 | 7 | 37 |
| 4 | 1 | 47 | 84 |
| 1 | 1 | 16 | 100 |
| 6 | 1 | 8 | 108 |
| 8 | 1 | 37 | 145 |
| 9 | 1 | 32 | 177 |
| 10 | 1 | 29 | 206 |
| 5 | 1 | 29 | 235 |
| 7 | 1 | 39 | 274 |
| 14 | 1 | 15 | 289 |
| 16 | 1 | 53 | 342 |
| 15 | 1 | 53 | 395 |
| 17 | 1 | 8 | 403 |
| 19 | 1 | 24 | 427 |
| 18 | 1 | 24 | 451 |
| 20 | 1 | 8 | 459 |
| 24 | 1 | 13 | 472 |
| 23 | 1 | 14 | 486 |
| 22 | 1 | 8 | 494 |
| 21 | 1 | 7 | 501 |
| 26 | 1 | 25 | 526 |
| 32 | 1 | 10 | 536 |
| 33 | 1 | 14 | 550 |

Table A.10 continues

| | | | |
|---|---|---|---|
| 34 | 1 | 41 | 591 |
| 35 | 1 | 42 | 633 |
| 28 | 1 | 25 | 658 |
| 25 | 1 | 10 | 668 |
| 27 | 1 | 11 | 679 |
| 29 | 1 | 11 | 690 |
| 31 | 1 | 25 | 715 |
| 36 | 1 | 47 | 762 |
| 61 | 1 | 3 | 765 |
| 62 | 1 | 8 | 773 |
| 37 | 1 | 7 | 780 |
| 152 | 2 | 403 | 403 |
| 45 | 2 | 80 | 483 |
| 38 | 2 | 80 | 563 |
| 39 | 2 | 7 | 570 |
| 40 | 2 | 41 | 611 |
| 74 | 2 | 40 | 651 |
| 59 | 2 | 14 | 665 |
| 75 | 2 | 101 | 766 |
| 46 | 2 | 7 | 773 |
| 47 | 3 | 41 | 41 |
| 56 | 3 | 28 | 69 |
| 154 | 3 | 105 | 174 |
| 153 | 3 | 104 | 278 |
| 155 | 3 | 403 | 681 |
| 64 | 3 | 33 | 714 |
| 54 | 3 | 25 | 739 |
| 70 | 3 | 11 | 750 |
| 73 | 3 | 40 | 790 |
| 90 | 4 | 19 | 19 |
| 63 | 4 | 16 | 35 |
| 65 | 4 | 8 | 43 |
| 111 | 4 | 23 | 66 |
| 112 | 4 | 162 | 228 |
| 141 | 4 | 151 | 379 |
| 144 | 4 | 170 | 549 |
| 145 | 4 | 137 | 686 |
| 66 | 4 | 18 | 704 |

Table A.10 continues

| | | | |
|---|---|---|---|
| 99 | 4 | 51 | 755 |
| 156 | 5 | 115 | 115 |
| 60 | 5 | 3 | 118 |
| 67 | 5 | 10 | 128 |
| 68 | 5 | 14 | 142 |
| 113 | 5 | 11 | 153 |
| 98 | 5 | 34 | 187 |
| 91 | 5 | 115 | 302 |
| 120 | 5 | 31 | 333 |
| 95 | 5 | 20 | 353 |
| 132 | 5 | 36 | 389 |
| 72 | 5 | 25 | 414 |
| 134 | 5 | 20 | 434 |
| 142 | 5 | 148 | 582 |
| 116 | 5 | 31 | 613 |
| 100 | 5 | 39 | 652 |
| 71 | 5 | 118 | 770 |
| 55 | 5 | 7 | 777 |
| 76 | 5 | 5 | 782 |
| 133 | 5 | 23 | 805 |
| 105 | 6 | 58 | 58 |
| 117 | 6 | 32 | 90 |
| 121 | 6 | 32 | 122 |
| 92 | 6 | 35 | 157 |
| 128 | 6 | 8 | 165 |
| 147 | 6 | 78 | 243 |
| 148 | 6 | 78 | 321 |
| 135 | 6 | 46 | 367 |
| 101 | 6 | 30 | 397 |
| 129 | 6 | 11 | 408 |
| 58 | 6 | 52 | 460 |
| 86 | 6 | 21 | 481 |
| 130 | 6 | 27 | 508 |
| 42 | 6 | 16 | 524 |
| 103 | 6 | 13 | 537 |
| 43 | 6 | 32 | 569 |
| 122 | 6 | 26 | 595 |
| 102 | 6 | 26 | 621 |

Table A.10 continues

| | | | |
|---|---|---|---|
| 127 | 6 | 55 | 676 |
| 44 | 6 | 66 | 742 |
| 118 | 6 | 26 | 768 |
| 123 | 6 | 19 | 787 |
| 96 | 7 | 31 | 31 |
| 126 | 7 | 48 | 79 |
| 11 | 7 | 17 | 96 |
| 49 | 7 | 47 | 143 |
| 136 | 7 | 64 | 207 |
| 77 | 7 | 28 | 235 |
| 157 | 7 | 105 | 340 |
| 12 | 7 | 11 | 351 |
| 13 | 7 | 32 | 383 |
| 30 | 7 | 29 | 412 |
| 138 | 7 | 15 | 427 |
| 93 | 7 | 26 | 453 |
| 88 | 7 | 23 | 476 |
| 137 | 7 | 22 | 498 |
| 139 | 7 | 34 | 532 |
| 114 | 7 | 19 | 551 |
| 140 | 7 | 22 | 573 |
| 124 | 7 | 14 | 587 |
| 115 | 7 | 14 | 601 |
| 125 | 7 | 19 | 620 |
| 87 | 7 | 47 | 667 |
| 104 | 7 | 45 | 712 |
| 131 | 7 | 18 | 730 |
| 41 | 7 | 47 | 777 |
| 78 | 7 | 8 | 785 |
| 119 | 8 | 55 | 55 |
| 48 | 8 | 13 | 68 |
| 57 | 8 | 12 | 80 |
| 94 | 8 | 46 | 126 |
| 89 | 8 | 13 | 139 |
| 97 | 8 | 19 | 158 |