DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MODELING AND SOLVING MIXED-MODEL ASSEMBLY LINE BALANCING PROBLEM WITH SETUPS

by
Şener AKPINAR

January, 2013
İZMİR

MODELING AND SOLVING MIXED-MODEL ASSEMBLY LINE BALANCING PROBLEM WITH SETUPS

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 Doctor of Philosophy in Industrial Engineering, Industrial Engineering Program

by
Şener AKPINAR

January, 2013
İZMİR

PH.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"MODELING AND SOLVING MIXED-MODEL ASSEMBLY LINE BALANCING PROBLEM WITH SETUPS"** completed by **ŞENER AKPINAR** under supervision of **PROF. DR. ADİL BAYKASOĞLU** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Adil BAYKASOĞLU

Supervisor

Prof. Dr. Can Cengiz ÇELİKOĞLU

Thesis Committee Member

Yard. Doç. Dr. Gökalp YILDIZ

Thesis Committee Member

Prof. Dr. Hadi GÖKÇEN

Examining Committee Member

Yard. Doç. Dr. Ali Serdar TAŞAN

Examining Committee Member

Prof. Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

First, I would like to point out my gratitude to my advisor, Prof. Dr. Adil BAYKASOĞLU for his guidance, continuing support, encouragement and invaluable advice throughout the progress of this dissertation.

I am truly grateful to my previous advisor, Prof. Dr.Günhan Miraç BAYHAN for her guidance at the first two years of this dissertation.

I would also like to thank to my committee members Prof. Dr. Can Cengiz ÇELİKOĞLU and Yard. Doç. Dr. Gökalp YILDIZ for their helpful comments and advice.

I would like to introduce my great thanks to my friend, Atabak ELMİ and all my friends, Neslihan AVCU, Hanefi Okan İŞGÜDER, Alper HAMZADAYI, and Abdurrahman TOSUN, for their support, whenever I need, and listening to my complaints during this period. I would also like to thank to my colleagues for their guidance during my studies at Dokuz Eylul University.

Last, but the most, I would like to emphasize my thankfulness to my parents, Muteber and Hasan Hüseyin AKPINAR, and my elder brothers, Zafer and Taner AKPINAR because of their love, confidence, encouragement and endless support in my whole life.

Şener AKPINAR

İzmir, January, 2013

MODELING AND SOLVING MIXED-MODEL ASSEMBLY LINE BALANCING PROBLEM WITH SETUPS

ABSTRACT

This dissertation concerns the type-I mixed-model assembly line balancing problem with setup times (MMALBPS-I). MMALBPS-I is an extension of classical MMALBP-I in which sequence-dependent setup times between tasks are taken into consideration. The main goal of this dissertation is developing the mathematical formulation of the problem and solving the problem with newly proposed parallel hybrid meta-heuristic approaches.

Within this context, a mixed-integer linear programming (MILP) model for the problem is developed and the capability of our MILP is tested through a set of computational experiments. Due to the complex nature of the problem, parallel hybrid algorithms are proposed in order to tackle the problem.

First, a new hybrid algorithm (ACO-GA), which executes ant colony optimization in combination with genetic algorithm, is developed. The proposed ACO-GA algorithm aims at enhancing the performance of ant colony optimization by incorporating genetic algorithm as a local search strategy for MMALBPS-I. In the proposed hybrid algorithm ACO is conducted to provide diversification, while GA is conducted to provide intensification.

Second, we tackled the problem with Bees Algorithm (BA), which is a relatively new member of swarm intelligence based meta-heuristics and tries to simulate the group behavior of real honey bees. However, the basic BA simulates the group behavior of real honey bees in a single colony; we aim at developing a new BA, which simulates the group behavior of honey bees in a single colony and between multiple colonies. The multiple colony type of BA is more realistic than the single colony type because of the multiple colony structure of the real honey bees.

The performances of the proposed algorithms are tested through a set of computational experiments and computational results indicate that both algorithms have satisfactory performances.

Keywords: Mixed-model assembly line balancing problem, sequence-dependent setup times, mixed-integer linear programming, hybrid meta-heuristics, ant colony optimization, genetic algorithm, bees algorithm

KARMA MODELLİ MONTAJ HATTI DENGELEME PROBLEMİNİN HAZIRLIK ZAMANLARI İLE MODELLENMESİ VE ÇÖZÜLMESİ

ÖZ

Bu tez I. tip karma modelli montaj hattı dengeleme problemini ele almaktadır. Bu problemin kapsamı, işler arasındaki sıra bağımlı hazırlık zamanları da dikkate alınarak genişletilmiştir. Bu tezin temel amacı, problemin matematiksel formülasyonunu geliştirmek ve problemi yeni önerilen paralel hibrit meta-sezgisel algoritmalarla çözmektir.

Bu kapsamda, problem için bir karma tamsayılı doğrusal programlama modeli geliştirilmiş ve modelin performansı bir deney seti üzerinde test edilmiştir. Problemin karmaşık yapısı nedeniyle, problemin çözümü için paralel hibrid algoritmalar önerilmiştir.

İlk olarak, problemin çözümü için karınca kolonisi optimizasyonu ve genetik algoritmanın birlikte çalıştığı yeni bir paralel hibrit algoritma geliştirilmiştir. Önerilen algoritma, genetik algoritmayı lokal arama strateji olarak kullanmayı ve bu şekilde karınca kolonisi optimizasyonunun performansını arttırmayı amaçlamaktadır.Önerilan hibrit algoritmada, genetik algoritma kuvvetlendirme (intensification) sağlarken karınca kolonisi algoritması çeşitlendirme (diversfication) sağlar.

İkinci sırada, sürü zekası tabanlı meta-sezgisel algoritmaların yeni bir üyesi olan ve gerçek bal arılarının grup içi davranışlarının benzetimi ile oluşturulan arılar algoritması ile problem çözülmüştür. Temel arılar algoritmasının tek bir koloni içindeki bal arılarının davranışlarının benzetimi üzerine kurulmuş olmasına rağmen, biz bu çalışma kapsamına bal arılarının tek bir koloni içinde ve çoklu koloniler arasındaki davranışlarının benzetimiyle yeni bir algoritma geliştirmeyi amaçlıyoruz. Çoklu koloni yapısına sahip arı algoritması, tek bir koloniden oluşan arı

algoritmasına göre gerçek bal arılarının çoklu kolonili bir yapıda olmalarından dolayı daha gerçekçidir.

Önerilen algoritmaların performansları bir dizi deneysel çalışma ile test edilmiş ve her iki algoritmanın da tatmin edici performansa sahip oldukları sonucuna varılmıştır.

Anahtar sözcükler: Karma modelli montaj hattı dengeleme problemi, sıra bağımlı hazırlık zamanları, karma tamsayılı doğrusal programlama, hibrit meta sezgiseler, karınca kolonisi optimizasyonu, genetik algoritma, arılar algoritması

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

1.1 Importance of the Problem

In 1913, Henry Ford changed the type of manufacturing system by introducing a moving belt in a factory for the first time. Before the moving belt, workers were able to build one piece of an item at a time instead of an item at a time. This changed type of manufacturing system named as assembly line and reduced the cost of production. Over the years a new problem type, design of efficient assembly lines, increased in importance. Assembly line balancing problem (ALBP) is a well-known assembly design problem, which consist of partitioning the assembly work among the workstations so as to optimize some objective.

Assembly lines are flow-oriented production systems where some operations are performed by some productive units referred to as workstations. The work-pieces (jobs) are moved along the line usually by a conveyor belt so as to successively visit all workstations, so work-pieces are moved from one workstation to another. Certain operations are repeatedly performed regarding the cycle time at each workstation.

Manufacturing a product on an assembly line requires partitioning the total amount of work into a set of elementary operations named tasks. Performing a task j takes a task time t_j and requires certain equipment of machines and/or skills of workers. Due to technological and organizational conditions precedence constraints between the tasks have to be observed. These elements are visualized by a precedence graph. It contains a node for each task, node weights for the task times and arcs for the precedence constraints. Any type of ALBP consists in finding a feasible line balance, i.e., an assignment of each task to exactly one workstation such that the precedence constraints and possibly further restrictions are fulfilled.

Assembly lines were firstly created to produce one single homogeneous product in high volumes. The balancing problem of this type of lines named as simple

assembly balancing problem (SALBP). Single-model assembly lines are the least suited production system for high variety demand scenarios. Current consumer-centric market conditions require high flexibility in manufacturing systems. Hence, assembly lines must be designed so as to satisfy high-mix/low volume manufacturing strategies. Due to high cost to build and maintain an assembly line, the manufacturers produce one model with different features or several models on a single assembly line. This changed type of assembly lines lead to arise the mixed-model assembly line balancing problem, which was handled by Thomopoulos (1967) for the first time in the literature.

Mixed-model assembly lines have mainly two types of balancing problems like traditional single-model assembly lines: design of a new assembly line for which the demand can be easily forecasted (type-I) and redesign of an existing assembly line (type-II) when changes in the assembly process or in the product range occurs. In this study we deal with the type-I mixed-model assembly line balancing problem (MMALBP-I), which has some particular features of the real-world assembly line balancing problems such as parallel workstations, zoning constraints, and sequence dependent setup times between tasks.

The assembly line balancing literature usually assumes that the setups are negligible because of their low times in comparison with operation times for most of the industrial assembly lines. Moreover, setups are considered independently as they are executed just before or after the tasks, hence, their times are added to task times Andrés et al. (2008). In such a situation, it is not an essential issue to determine task performing sequences in a workstation, however, task performing sequences are vital for minimizing the workstation global time, in case of sequence dependent setup times. Furthermore, determining the optimum task performing sequence provides more effectively balanced assembly lines. In other words, the maximum line efficiency, which is one of the most important performance criteria of the assembly lines, can be obtained if the optimum task performing sequences are achieved. Also, considering sequence dependent setup times between tasks becomes more important when cycle time is low, since setup times may represent a high percentage of it.

The main endeavor of this study is to introduce the type-I mixed-model assembly line balancing problem with setups (MMALBPS-I), which is an extension of classical MMALBP-I and takes into consideration the sequence dependent setup times between tasks.

1.2 Framework of the Dissertation

The concept of sequence dependent setup times is an actual framework in assembly line balancing problems (ALBP). Most of the studies on assembly line balancing problem with sequence dependent setup times have focused extensively on single-model lines. Nevertheless, single-model assembly lines are not able to respond the demand for higher product variability, which is an outcome of the current consumer-centric market conditions. Thus, high-mix/low-volume manufacturing strategies substitute for low-mix/high-volume manufacturing strategies. That is to say, mixed-model assembly lines substitute for single-model assembly lines. At this point, the lack of studies dealing with the consideration of setups for mixed-model assembly lines stands out in the existing literature.

The main goal of this study is to introduce the MMALBPS-I, by formally describing the problem and developing solution procedures in order to tackle the problem, since MMALBP-I is NP-hard (Bukchin & Rabinowitch, 2006) then MMALBPS-I is also NP-hard.

Firstly, we developed a mixed integer linear programming (MILP) model, which considers the phenomena of sequence dependent setup times for mixed-model assembly lines for the first time, in order to formally describe the problem. However, due to the NP-Hard nature of the problem the proposed MILP model is not able to solve large scale problems. Therefore, we developed meta-heuristics based hybrid algorithms in order to tackle the problem, since hybrids are believed to benefit from synergy (Blum et al., 2011). Among the meta-heuristics, we selected genetic algorithm (GA), ant colony optimization (ACO), and bess algorithm (BA) and we developed new hybrid algorithms based on these three meta-heuristics.

1.3 Outline of the Thesis

Rest of the study involves five chapters. The following chapter contains the problem definition, and a literature survey about the mixed-model assembly line balancing problem, and the concept of sequence-dependent setup times in assembly line balancing.

Chapter three gives the developed MILP model for the type-I mixed-model assembly line balancing problem with sequence dependent setup times, zoning constraints, and parallel workstations.

The fourth and fifth chapters mainly focus on solving MMALBPS-I with parallel workstations and zoning constraints using the proposed hybrid algorithms. Chapter fourth presents a new hybrid algorithm, which executes ant colony optimization in combination with genetic algorithm (ACO-GA), while chapter five presents a new multiple colony hybrid bess algorithm (MCHBA), which simulates the group behavior of honey bees in a single colony and between multiple colonies in a more realistic way than the single colony types.

Finally, the conclusions and the contributions of this study are discussed in chapter six.

CHAPTER TWO
PROBLEM DEFINITION AND LITERATURE SURVEY ON MMALBP-I

2.1 Chapter Introduction

The role of assembly lines in manufacturing systems has been changing through time due to the customer expectations. At the beginning, assembly lines provided manufacturers to produce low variety of products in high volumes. By the way, they gained low production costs, reduced cycle times and accurate quality levels, which are essential advantages for companies in order to remain being competitive in market. The initial designs of assembly lines enabled to produce a single homogenous product. Such assembly lines are the least suited manufacturing systems for the cases of high variety demand scenarios and named as single-model assembly lines. Due to the current competitive and consumer-centric market conditions, a requirement of rearrangement of the single-model assembly lines arises. The newly designed assembly lines must be able to produce different models with different number of features, because customers may prefer a model with regard to their desires and financial capabilities. Hence, manufacturers must produce one model with different features or several models on a single assembly line within the scope of being productive. Under these circumstances, the mixed-model assembly line balancing problem arises to smooth the production and decrease the cost.

The main goal of this chapter is to provide a general understanding about the mixed-model assembly line balancing problem and the consideration of the sequences dependent setup times in assembly line balancing. The rest of this chapter is organized as follows. Following section gives a brief classification of assembly lines. Section 2.3 gives information about mixed-model assembly lines, introduces mixed-model assembly line balancing problem with the consideration of sequence dependent setup times between tasks. In section 2.4, first, a review concerning the existing literature about the assembly line balancing problems with sequence dependent setup times is given. A summarized literature survey on MMALBP-I is also given in Section 2.4.

2.2 Assembly Lines

An assembly line (AL) is a manufacturing process consisting of various workstations connected by a material handling system in which particular tasks are executed in order to produce a final product. Assembly lines are the most suitable manufacturing system in a mass production environment, because they allow the assembly of complex products by workers with limited training, by dedicated machines and/or by robots.

Assembly lines can be categorized by taking into account the number of products to be assembled and the way they are processed (Scholl, 1999). An assembly line can be designed so as to assemble one product or several products with identical production process. These types of assembly lines are named as single-model lines. An assembly line is named as multi-model lines if several products are assembled in batches or named as mixed-model lines if different models of the same base product are assembled simultaneously in the same line in an arbitrarily intermixed sequence not in batches. All these types of assembly lines are illustrated in Figure 2.1, where different models symbolized with different geometrical shapes. For further information about assembly lines, the reader can refer to (Scholl, 1999).

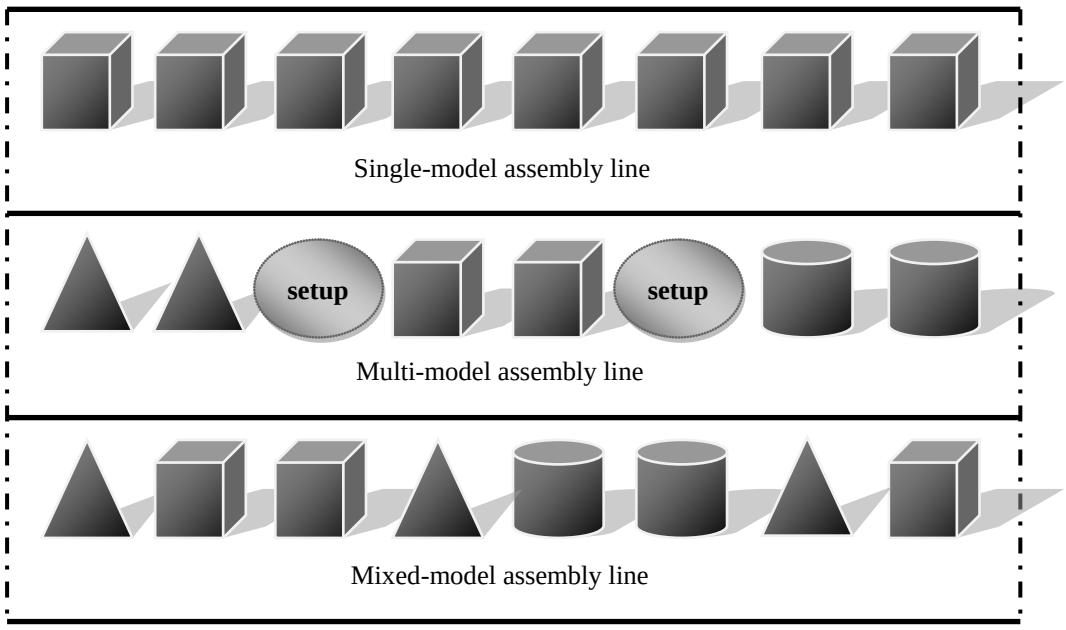


Figure 2.1 Types of assembly lines

In this study, we deal with the mixed-model assembly lines with some particular features of real world problems such as parallel workstations, zoning constraints, and sequences dependent setup times.

2.3 Mixed-Model Assembly Lines

Current markets are characterized as consumer-centric resulted in a growing trend for higher product variability. Hence, it is required to produce several products or different models of the same base product in the same assembly line. Nevertheless, single-model assembly lines, which are the most suited production systems for low variety demand scenarios, are not able to respond the requirements of this new type of manufacturing strategies anymore. Therefore, manufacturers prefer producing one model with different features or several models on a single assembly line in order to avoid the high cost to build and maintain an assembly line for each model. At this point mixed-model assembly lines preferred to multi-model assembly lines, since they provide higher flexibility then multi-model lines.

Zhoa et al. (2004) stated that two points must be considered for mixed-model assembly lines; first at the "design" level and the second at the "operational" level. The entire tasks for the assembly operation have to be assigned to workstations at the design level in order to optimize a given "design measure" and the sequence defines the release order of the models into the line must also be determined at the operational level in order to optimize a given "operational performance measure". The first one refers to the ***balancing problem*** while the second refers to the ***sequencing problem*** of the mixed-model assembly lines. This study deals with the balancing problem of mixed-model assembly lines, which is defined in the following sub-section in details.

2.3.1 Mixed-Model Assembly Line Balancing Problem

The main goal of an assembly line balancing problem is to partition the entire tasks of the assembly operation among workstations so as to optimize a pre-defined

performance measure. The assembly line balancing problems have been classified by the existing literature in various ways (Erel & Sarin, 1998; Becker & Scholl, 2006; Boysen et al., 2007; Boysen et al., 2008). Unlike the single-model lines, different models of a product are assembled on mixed-model assembly lines. The models are launched to the line one after another and moved from workstation to workstation in ordered sequence. Since we deal with the mixed-model assembly line balancing problem within the scope of this study, only the main characteristics of MMALBP which results from the joint assembly of several products are mentioned as below.

- The line is used to produce more than one type of product simultaneously in an intermixed sequence not in batches.
- The assembly of each model requires performing a set of tasks which are connected by precedence relations (precedence graph for each model).
- A subset of tasks is common to all models; the precedence graphs of all models can be combined to a non-cyclical joint precedence graph.
- Tasks which are common to several models are performed by the same station but may have different operation times; zero operation times indicate that a task is not required for a model.
- Fixed total time available for the production during the planning period is known.
- Expected demands for all models (expected model mix) during the planning period are known.

Mixed-model assembly lines have mainly two types of balancing problems like traditional single-model assembly lines: design of a new assembly line for which the demand can be easily forecasted (Type-I) and redesign of an existing assembly line (Type-II) when changes in the assembly process or in the product range occurs. In this study we deal with MMALBP-I, which has some particular features of the real-world assembly line balancing problems such as parallel workstations, zoning constraints, and sequence dependent setup times between tasks.

2.3.2 Mixed-Model Assembly Line Balancing Problem with Setups

The type-I mixed-model assembly line balancing problem which is considered in this study, has the following characteristics in addition to aforementioned characteristics:

- The precedence relationships among tasks for each model are known and the precedence diagrams for all the models can be combined such that the resulting diagram contains the *N* tasks.
- Workstations along the line can be replicated to create parallel workstations, when the demand is such that some tasks have processing times higher than the cycle time.
- Assignment of tasks to a specific workstation can be forced or forbidden through the definition of zoning constraints.

Taking into account these features three types of constraints, precedence, zoning, and capacity constraints, are arisen for the assembly line balancing problem on hand.

Precedence constraints determine the sequence according to which the tasks can be processed. Precedence constraints are usually depicted in a precedence diagram. A task can only be assigned to a workstation if it has no predecessors or if all of its predecessors have already been assigned to a workstation.

Zoning constraints can be positive or negative. Positive zoning constraints force the assignment of certain tasks to a specific workstation. Negative zoning constraints forbid the assignment of tasks to the same workstation.

Capacity constraints provide that the workload of a workstation does not exceed the cycle time. Under some demand conditions, the assembly line may need to be operated with a cycle time such that some of the tasks in the assembly process have processing times higher than cycle time. In this case, the replication of the

workstation to which the tasks with processing time higher than the cycle time are assigned is required, in order for demand to be met.

The mixed-model nature of the problem on hand requires the cycle time (*C*) to be defined by taking into account the different models' demand over the planning horizon. Thus, if a line is required to assemble *M* models each with a demand of D_m units over the planning horizon *(P)*, the cycle time of the line is computed as follows:

$$C = P \div \sum_{m=1}^{M} D_m \qquad m \in \{1, \dots, M\} \qquad (2.1)$$

On the other hand, q_m is the overall proportion of the number of units of model *m* being assembled and calculated by using Equation 2.2.

$$q_m = D_m \Big/ \sum_{m=1}^{M} D_m \qquad m \in \{1, \dots, M\} \qquad (2.2)$$

In this study, balancing of mixed-model assembly line balancing problem with setups (MMALBPS-I) is studied. MMALBPS-I is an extension of classical MMALBP-I in which sequence-dependent setup times between tasks are taken into consideration.

2.3.2.1 Sequence Dependent Setup Times between Tasks

The concept of the sequence-dependent setup times had been considered negligible until the importance of setup times were investigated for the scheduling problems (Allahverdi et al., 1999). Furthermore, setup times were generally considered in low production systems like job shops (Allahverdi et al., 2008). On the other hand, most of the studies about assembly lines also assumed that setup times are negligible, because of their low proportion in comparison with task processing times. The phenomenon of sequence dependent setup times has been a challenging

field in ALBPs, since Andrés et al. (2008) (see the corrigendum to this paper provided by Pastor et al., 2010) dealt with the setup times for the first time for the SALBP.

For the assembly line balancing applications setups were considered independently as they executed just before or after the tasks. Thus, their times were added to task times (Andrés et al., 2008). In such situations, it is not required to determine intra-stations schedules; however, they have considerable effect on the workload of a workstation in case of sequence dependent setup times between tasks. In other words, different intra-station schedules mean different workloads for a workstation. Since the aim of assembly line studies is achieving effectively balanced lines, determining optimum intra-station schedules become much more important. That is to say, determining the optimum task performing sequences provides the maximum line efficiency, which is one of the most important performance criteria of the assembly lines. Besides, if the cycle time is low, considering sequence dependent setup times between tasks becomes more important, because setup times may represent a high percentage of cycle time.

Scholl et al. (2011) modified the consideration of the sequence dependent setup times for assembly line balancing problems by introducing the phenomena of backward and forward setups in addition to (Andrés et al., 2008).

"....The term *forward setup* refers to a situation where task *j* is executed directly after task *i* in the *same cycle*, i.e., at the same workpiece, observing a (forward) setup time $\tau_{ij} \geq 0$. A *backward setup* occurs if task *i* is the last one executed at the workpiece of a cycle *p* and the worker has to move to the *next workpiece* which is to be assembled in cycle *p*+1. This transfer causes a (backward) setup time $\mu_{ij} \geq 0$ which must be finished by the end of cycle *p* in order to start execution of task *j* just when cycle *p* + 1 begins. Note that since stations are supposed to be independent and exclusively operated by a single (team of) worker(s), forward and backward setups are only considered among tasks at the same station, not between adjacent stations" (Scholl et al. 2011).

MMALBPS adds sequence-dependent setup time considerations to the classical mixed-model assembly line balancing problem as follows: whenever a task *j* is assigned immediately after another task *i* for model *m* at the same workstation, a forward setup time (FST_{ij}^{m}) must be added due to forward setup operation (FS_{ij}^{m}) to compute the global workstation time for model *m*, thereby providing the task sequence inside each workstation. Furthermore, if a task *i* is the last one assigned to the workstation in which task *j* was the first task assigned for model *m*, then a backward setup time (BST_{ij}^{m}) must also be considered due to backward setup operation (BS_{ij}^{m}). This is because the tasks are repeated cyclically; the last task in one cycle of the workstation is performed just before the first task in the next cycle. Hence, MMALBPS consists of assigning a set of tasks for a set of models to an ordered sequence of workstations, such that the precedence constraints between tasks are maintained, the setup times between tasks for all models are considered and a given efficiency measure is optimized. Within the context of this study, we deal with the MMALBPS-I, which aims at minimizing the number of workstations for a given cycle time and a given set of *M* models with sequence dependent setup times between tasks for all models.

As an example, we can take a case in which there are two models (*A* and *B*) assembled at the same line over a planning horizon of 480 time units. The demands for each model *A* and *B* are deterministically known; 20 and 28 units, respectively. Hence, the cycle time (*C*) is equal to $480 \div (20 + 28) = 10$ time units. On the other hand, q_A is equal to $20 \div 48 = 0.42$ and q_B is equal to $28 \div 48 = 0.58$. Combined precedence diagram originally used by Gokcen & Erel (1998), for these two models is depicted in Figure 2.2.

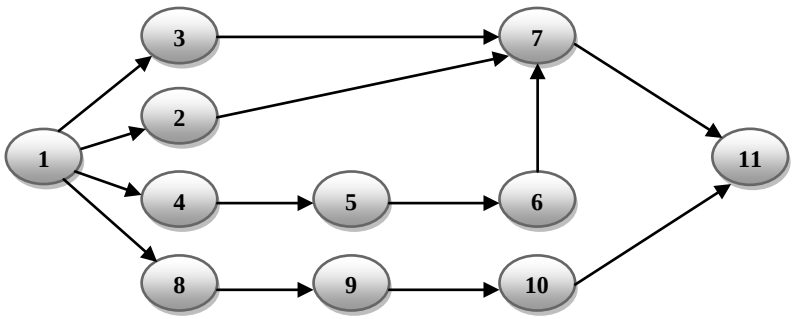


Figure 2.2 Combined precedence diagram

The processing times of tasks, forward and backward setup times between tasks for the models *A* and *B* are shown in Tables 2.1 and 2.2, respectively.

Table 2.1 Task, forward and backward setup time matrixes for model A

Setup	Task	1	2	3	4	5	6	7	8	9	10	11
Forward Setups Times	1	0.62	0.14	**0.33**	**0.11**	0.54	0.46	0.46	0.41	0.39	0.27	0.74
	2	0.59	0.39	0.11	0.56	**0.59**	0.19	0.11	0.48	**0.19**	0.24	0.44
	3	0.22	0.44	0.41	**0.38**	0.39	0.32	0.47	0.18	0.13	0.21	0.38
	4	0.18	0.50	**0.13**	0.45	0.13	0.64	0.46	0.49	0.44	0.28	0.19
	5	0.42	**0.31**	0.26	0.44	0.44	0.53	0.16	**0.49**	0.52	0.24	0.30
	6	0.36	0.46	0.13	0.55	0.25	0.42	**0.49**	0.53	0.18	0.23	0.47
	7	0.24	0.40	0.26	0.13	0.34	0.47	0.28	0.21	0.48	**0.53**	**0.25**
	8	0.20	0.57	0.11	0.59	**0.27**	0.60	0.57	0.33	**0.48**	0.16	0.51
	9	0.41	0.36	0.21	0.31	0.58	0.36	0.23	0.31	0.32	0.18	0.12
	10	0.18	0.54	0.17	0.23	0.12	**0.60**	0.20	0.37	0.18	0.57	**0.49**
	11	0.43	0.52	0.27	0.58	0.29	0.26	0.26	0.54	0.39	0.14	0.60
Backward Setups Times	1	0.45	0.53	0.66	0.30	0.74	0.27	0.49	0.15	0.46	0.61	0.58
	2	0.74	0.83	0.40	0.45	0.52	0.46	0.36	0.57	0.46	0.37	0.34
	3	**0.14**	0.42	0.70	0.30	0.40	0.41	0.49	0.56	0.63	0.45	0.35
	4	**0.46**	0.51	0.44	0.53	0.68	0.58	0.61	0.65	0.57	0.65	0.18
	5	0.44	0.67	0.51	0.48	0.70	0.51	0.40	0.71	0.62	0.57	0.37
	6	0.50	0.34	0.38	0.58	0.73	0.60	0.73	0.71	0.39	0.70	0.65
	7	0.49	0.51	0.61	0.59	0.33	0.37	0.15	0.51	0.42	0.77	0.53
	8	0.36	0.39	0.39	0.47	0.60	0.37	0.77	0.53	0.47	0.45	0.76
	9	0.36	**0.50**	0.56	0.66	0.68	0.61	0.53	**0.33**	0.46	0.40	0.46
	10	0.46	0.47	0.17	0.48	0.35	0.71	0.69	0.60	0.56	0.71	0.73
	11	0.57	0.59	0.40	0.70	0.63	**0.51**	0.51	0.20	0.53	**0.70**	0.44
Task time (T_A)		2.75	1.25	3.00	3.00	2.25	1.80	2.10	2.30	2.10	2.00	2.00

Table 2.2 Task, forward and backward setup time matrixes for model B

Setup	Task	1	2	3	4	5	6	7	8	9	10	11
Forward Setups Times	1	0.42	0.47	**0.38**	**0.17**	0.33	0.12	0.33	0.31	0.37	0.54	0.41
	2	0.22	0.28	0.26	0.32	**0.42**	0.56	0.31	0.35	**0.19**	0.25	0.44
	3	0.12	0.21	0.53	**0.26**	0.46	0.41	0.43	0.21	0.21	0.30	0.32
	4	0.31	0.49	**0.11**	0.21	0.15	0.36	0.29	0.51	0.51	0.22	0.27
	5	0.42	**0.11**	0.25	0.31	0.20	0.43	0.19	**0.25**	0.52	0.41	0.25
	6	0.27	0.39	0.37	0.48	0.18	0.33	**0.27**	0.19	0.02	0.28	0.46
	7	0.35	0.26	0.18	0.36	0.35	0.38	0.36	0.17	0.43	**0.39**	**0.36**
	8	0.42	0.37	0.41	0.13	**0.12**	0.46	0.41	0.17	**0.47**	0.33	0.54
	9	0.29	0.30	0.15	0.46	0.13	0.43	0.38	0.42	0.49	0.19	0.45
	10	0.37	0.49	0.21	0.33	0.28	**0.41**	0.29	0.21	0.37	0.31	**0.17**
	11	0.48	0.45	0.20	0.48	0.23	0.54	0.36	0.23	0.31	0.40	0.40
Backward Setups Times	1	0.47	0.45	0.43	0.68	0.38	0.57	0.46	0.67	0.57	0.46	0.64
	2	0.39	0.54	0.38	0.61	0.61	0.62	0.47	0.71	0.50	0.51	0.15
	3	**0.20**	0.47	0.45	0.60	0.70	0.72	0.34	0.70	0.20	0.44	0.15
	4	**0.46**	0.33	0.45	0.80	0.61	0.70	0.48	0.51	0.39	0.53	0.53
	5	0.55	0.61	0.40	0.53	0.61	0.51	0.35	0.67	0.45	0.68	0.49
	6	0.40	0.46	0.33	0.69	0.45	0.40	0.71	0.21	0.60	0.58	0.49
	7	0.14	0.56	0.39	0.71	0.19	0.45	0.69	0.58	0.55	0.61	0.51
	8	0.60	0.68	0.46	0.70	0.63	0.58	0.60	0.53	0.61	0.65	0.53
	9	0.58	**0.18**	0.44	0.42	0.59	0.50	0.56	**0.17**	0.74	0.57	0.58
	10	0.65	0.51	0.50	0.53	0.61	0.61	0.71	0.48	0.46	0.65	0.75
	11	0.20	0.61	0.47	0.56	0.50	**0.19**	0.73	0.40	0.47	**0.62**	0.63
Task time (T_B)		2.75	1.50	3.00	3.15	2.50	2.00	2.30	2.50	2.00	2.10	2.00

Figure 2.3 represents two different solutions with a unique assignment of tasks to 3 different workstations. That is, the number of workstations is equal to 3 and the assignment of task to workstations in both solutions are the same, and two different intra-station schedules of tasks caused two different solutions. The intra-station schedules of tasks are displayed with discontinuous lines. As pointed out in Figure 2.3, different intra-station schedules lead to different work-loads (*WL*) for the workstations (*WS*). This situation indicates the importance of considering sequence dependent setup times between tasks.

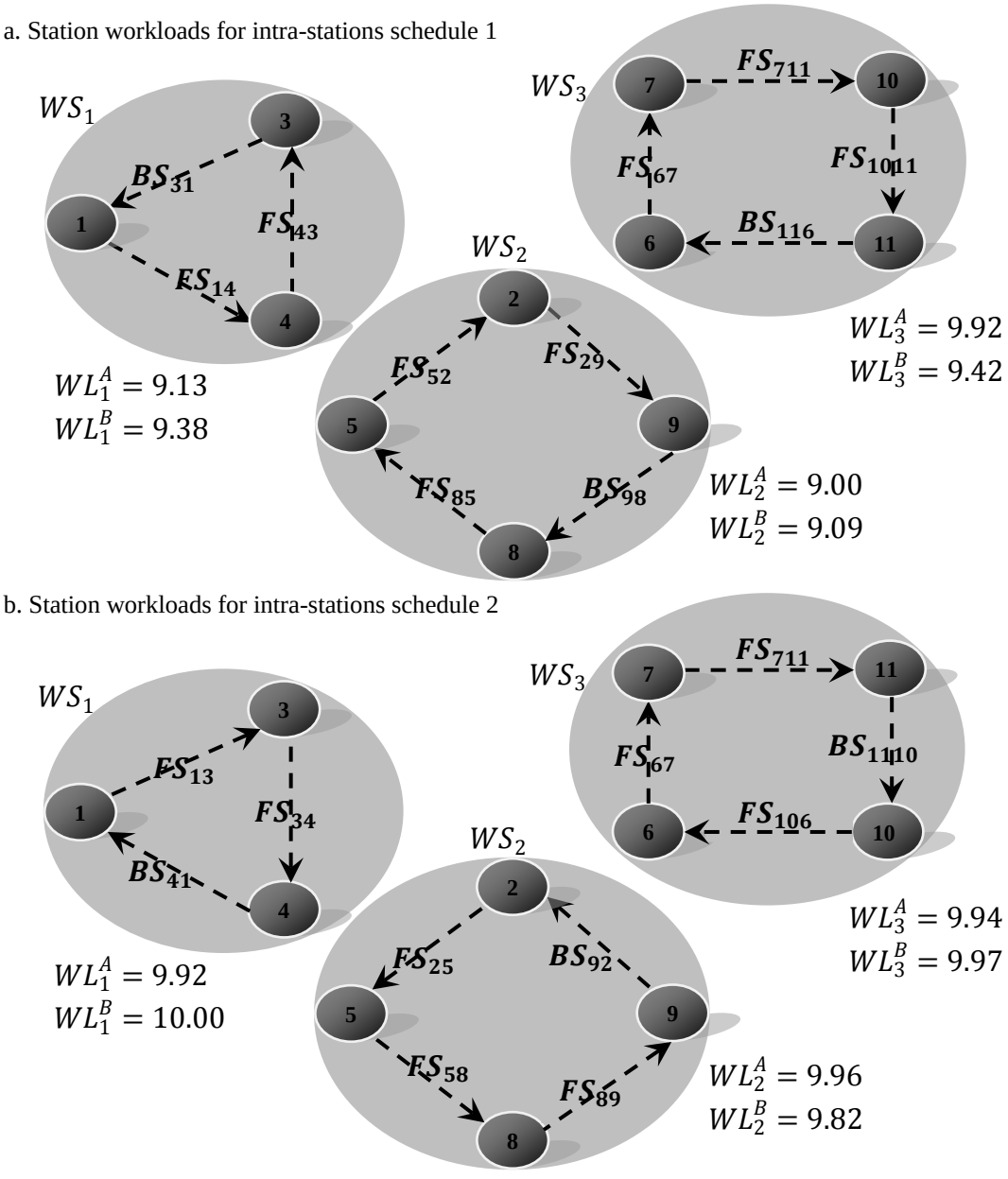


Figure 2.3 Station work-loads for different intra-station schedules

The reader must also be noted the following assumptions, which are directly related to the MMALBPS-I.

- Task processing times and setup times between tasks are known deterministically.
- Processing and setup times are independent on the workstation in which tasks are processed.

2.4 Literature Survey

The existing literature about the assembly line balancing problems with sequence dependent setup times has extensively dealt with single-model lines. Andrés et al. (2008) extended the simple version of the ALBPs by considering the sequence dependent setup times between tasks for the first time and they referred to as general assembly line balancing problem with setups (GALBPS). The authors developed the mathematical programming model of the problem. Due to the high combinatorial nature of the problem they provided some heuristics and a GRASP algorithm to tackle the innovative problem. Moreover, Martino & Pastor (2010) developed heuristic procedures based on priority rules in order to solve the same problem; however the performance of their procedures were not effective in high-size tests. A similar problem was introduced by Scholl et al. (2008) and they formulated several versions of a mixed-integer program for the problem. As a result of their experiments, the authors stated that it is not effective enough modeling and solving the problem with MIP standard software. Scholl et al. (2011) modified the problem by introducing the phenomena of backward and forward setups and the triangle inequality for the setup times. They formulated the modified problem as a mixed-binary linear model and developed effective solution procedures for the problem. Yolmeh & Kianfar (2012) dealt also with single-model lines with sequence dependent setup times between tasks. They proposed a hybrid genetic algorithm for solving the problem. Hamta et al. (2012) enriched the SALBP by adding some realistic relevant aspects such as sequence dependent setup times. They developed a mathematical model for the problem and the problem was tackled by a combination

of particle swarm optimization (PSO) algorithm with variable neighborhood search (VNS). Seyed-Alagheband et al. (2011) addressed type-II SALBP, which was enriched by considering sequence-dependent setup times between tasks (GALBPS-II). They proposed a mathematical model based on Andres et al.'s (2008) model and the authors developed a novel simulated annealing (SA) algorithm to tackle the problem. Özcan & Toklu (2010) handled the two-sided assembly line balancing problem with setups (TALBPS). The authors proposed a mixed integer program in order to solve and model the problem. The proposed model minimizes the number of mated-stations as a primary objective and minimizes the number of stations as a secondary objective. A heuristic approach was also presented.

This study concerns the type-I mixed-model assembly line balancing problem with sequences dependent setup times between tasks (MMALBPS-I). MMALBPS-I is an extension of classical MMALBP-I in which sequence-dependent setup times are taken into consideration and handled by Akpınar et al. (2013) for the first time to the best of our knowledge. MMALBPS-I aims at assigning a set of tasks for a set of models to an ordered sequence of workstations and determining the intra-stations schedules. The problem seeks the optimum value of the number of workstations so as to maintain the precedence constraints and to consider sequence dependent setup times between tasks for a predefined cycle time.

The relevant literature about the solution procedures of the mixed-model assembly lines was initiated by the approaches of Thomopoulos (1970) and can be divided into three groups: mathematical programming, heuristics and meta-heuristics, and hybrid approaches. Heuristic and meta-heuristic approaches were widely used in order to cope with the problem. The field of hybrid approaches has become very popular among researchers because of the insufficient performance of heuristics and pure meta-heuristics while exploring the solution space effectively as problems get larger and more complex as in real life. Mathematical programming approaches are used to formally describe the problem. We summarized the published papers by taking into account the line configuration, the methodology, and the employed data to test the performance of the proposed approach and the summary is presented in Table 2.3.

Table 2.3 An overview of the approaches in the literature on MMALBP-I

Publications	Line Configuration	Methodology	Test Problem
Askin & Zhou (1997)	Straight line, Parallel stations	Nonlinear Integer Programming, Heuristic	Randomly generated
McMullen & Frazier (1997)	Straight line, Parallel stations	Heuristic, Simulation	Randomly generated
Gokcen & Erel (1997)	Straight line	Binary Goal Programming	More than one
McMullen & Fraizer (1998)	Straight line, Parallel stations	Simulated Annealing	Randomly generated
Gokcen & Erel (1998)	Straight line	Binary Integer Programming	More than one
Sparling & Miltenburg (1998)	U-line	Approximate Solution Algorithm, Mathematical Model	Only one problem
Erel & Gokcen (1999)	Straight line	Network Programming	Only one problem
Merengo et al. (1999)	Paced and unpaced lines	Heuristic	Randomly generated
Vilarinho & Simaria (2002)	Straight line, Parallel stations	Mathematical Model, Simulated Annealing	Randomly generated
Buckhin et al. (2002)	Straight line	Mathematical Model, Heuristic	Only one problem
Miltenburg (2002)	U-line	Genetic Algorithm	Randomly generated
McMullen & Tarasewich (2003)	Straight line, Parallel stations	Ant Colony Optimization, Simulation	Benchmark problems
Zhao et al. (2004)	Paced line	Heuristic	Randomly generated
Mendes et al. (2005)	Straight line, Parallel stations	Heuristic, Simulation	Case study
Hop (2006)	Straight line	Fuzzy Binary Linear Programming, Heuristic	Randomly generated
Bock (2006)	Straight line	Distributed Search Procedures	More than one
Buckhin & Rabinowitch (2006)	Straight line	Branch and Bound Algorithm based Heuristic, Mathematical Model	Randomly generated
Noorul Haq et al. (2006)	Straight line	**Hybrid Genetic Algorithm**	More than one
Vilarinho & Simaria (2006)	Straight line, Parallel stations	Ant Colony Optimization	Benchmark problems
Kara et al. (2007)	U-line	Simulated Annealing, Mathematical Model	Randomly generated
Bock (2008)	Straight line	Tabu Search	Randomly generated
Simaria and Vilarinho (2009)	Two-sided line	Ant Colony Optimization, Mathematical Model	Benchmark problems
Özcan & Toklu (2009)	Two-sided line	Mathematical Model, Simulated Annealing	Benchmark problems
Hwang & Katamaya (2009)	U-line	Genetic Approach	Benchmark problems
Özcan et al. (2010)	Parallel lines	Simulated Annealing	Benchmark problems
Hwang & Katamaya (2010)	Straight and U-line	Evolutionary Approach	Case study
Yagmahan (2011)	Straight line	Ant Colony Optimization	Randomly generated
Kazemi et al. (2011)	U-line	Genetic Algorithm, Mathematical Model	Benchmark problems
Akpınar & Bayhan (2011)	Straight line, Parallel stations	**Hybrid Genetic Algorithm**	Benchmark set
Kara et al. (2011)	Straight line	Integer Goal and Fuzzy Goal Programming	Randomly generated
Hamzadayi & Yildiz (2012)	U-line	Genetic Algorithm, Simulated Annealing	Benchmark set
Akpınar et al. (2013)	Straight line	**Hybrid Ant Colony Optimization-Genetic Algorithm**	Randomly generated

This summarized review reveals there is only one paper (Akpınar et al., 2013) handled MMALBP-I with the sequence dependent setup times between tasks. On the other hand, there are only three hybrid approaches (Noorul Haq et al., 2006; Akpınar & Bayhan, 2011; Akpınar et al, 2013) dealing with MMALBP-I between the years 1997 and 2013. Noorul Haq et al. (2006) combined GA with only modified version of ranked positional weight technique (RPWT), while Akpınar & Bayhan (2011) presented a new hybrid GA in which the RPWT, Kilbridge & Wester Heuristic and Phase-I of Moodie & Young Method are sequentially hybridized with GA. Both of the hybrid approaches belong to the class of sequential hybrid algorithms, and are based on hybridizing problem specific heuristics with meta-heuristics. The study of Akpınar et al. (2013), developed a new hybrid algorithm belongs to the class of parallel hybrid algorithms and combines two well known meta-heuristics, ant colony optimization and genetic algorithm. From this review, it can be noticed that there is lack of mathematical models about mixed-model assembly line balancing problem with sequence dependent setups between tasks in the existing literature. The following chapter of this study aims at removing this lack of the existing literature by developing a mixed integer linear mathematical programming model for mixed-model assembly line balancing problem with setups.

CHAPTER THREE
A MIXED INTEGER LINEAR PROGRAMMING MODEL FOR MMALBPS-I

3.1 Chapter Introduction

Assembly lines were firstly created to produce one single homogeneous product in high volumes. The balancing problem of this type of lines named as simple assembly balancing problem (SALBP), which was first mathematically formulated by Salveson (1955). Single-model assembly lines are the least suited production system for high variety demand scenarios.

Current consumer-centric market conditions require high flexibility in manufacturing systems. Hence, assembly lines must be designed so as to satisfy high-mix/low volume manufacturing strategies. Due to high cost to build and maintain an assembly line, the manufacturers produce one model with different features or several models on a single assembly line. This changed type of assembly lines lead to arise the mixed-model assembly line balancing problem, which was handled by Thomopoulos (1967) for the first time in the literature.

The relevant literature about the solution procedures of the mixed-model assembly lines was initiated by the approaches of Thomopoulos (1970) and can be divided into three groups: mathematical programming, heuristics and meta-heuristics, and hybrid approaches. For more detailed information, the reader can refer to Battaïa & Dolgui's (2012b) recent survey.

Heuristic and meta-heuristic approaches were widely used in order to cope with the problem. The field of hybrid approaches has become very popular among researchers because of the insufficient performance of heuristics and pure meta-heuristics while exploring the solution space effectively as problems get larger and more complex as in the real life. On the other hand, mathematical programming approaches are used to formally describe the problem. In this study we proposed a new mathematical programming model for type-I mixed-model assembly line

balancing with sequence dependent setup times between tasks (MMALBPS-I). To the best of our knowledge, this is the first attempt to model type-I mixed-model assembly line balancing problem while considering the sequence dependent setup times in the literature.

Akpınar et al. (2013) summarized the published papers related to type-I mixed-model assembly line balancing problem (MMALBP-I) between the years 1997 and 2011 by taking into account the line configuration, the methodology, and the employed data to test the performance of the proposed approach. From their summary, it is observed that few papers dealt with mathematically modeling of the MMALBP-I and none of these studies handled the sequence dependent setup times between tasks.

Askin & Zhou (1997) proposed a non-linear integer mathematical model for MMALBP-I. Their model allows using parallel workstations if required. By the way, the authors relaxed the splitting restriction for the first time.

Gokcen & Erel (1997) modeled the MMALBP-I as a binary goal program. They considered several conflicting goals and their model provides flexibility to the decision maker. Their model also allow to the use of zoning constraints. Moreover, Gokcen & Erel (1998) developed a binary integer programming model for the MMALBP-I. The authors stated that their model may be used as a validation tool for the heuristic procedures for the MMALBP-I. On the hand, Erel & Gokcen (1999) proposed a shortest-route formulation of the MMALBP-I.

Vilarinho & Simaria (2002) combined the concepts of parallel workstations assignment and zoning constraints in their mathematical programming model. Their model aims at minimizing the number of workstations as a primary goal, and balancing the workloads between and within workstations as a secondary goal.

The literature about the mixed-model assembly line balancing problem (MMALBP) use a restriction ensures that assigning common tasks of different

models to the same workstation. This restriction has been relaxed by Bukchin et al. (2002), and Bukchin & Rabinowitch (2006) and they allow the assignment of a common task for multiple products to different workstations. The same relaxation was also used by Kara et al. (2011). They proposed a new binary mathematical programming model based on the Bukchin & Rabinowitch's (2006) model and have also developed two goal programming approaches, one with precise and the other with fuzzy goals. Hop (2006) dealt also with fuzzy concept and handled the MMALBP with fuzzy processing times and formulated the problem as a fuzzy binary linear programming model, which was transformed to a mixed zero-one program.

Simaria & Vilarinho (2009) dealt with the MMALBP-I with a different line configuration, two-sided assembly line and developed a mathematical programming model covers the parallel workstations assignment and zoning constraints. The phenomenon of two-sided assembly lines was also handled by Ozcan & Toklu (2009). They also proposed a mathematical programming model for the two-sided MMALBP-I.

On the other hand, Sparling & Miltenburg (1998), and Kazemi at al. (2011) handled the U-line MMALBP-I. They all developed mathematical programming models for the problem.

In this study, we deal with the MMALBP-I with some particular features of the real world problems such as parallel workstations and zoning constraints. Furthermore, we extend the problem by adding sequence dependent setup times between tasks, which is a new concept for assembly line balancing problem. We developed a mixed integer linear programming (MILP) model for formally describing the extended problem.

The rest of this chapter is organized as follows. The proposed MILP model is given in Section 3.2. An illustrative example is solved in Section 3.3. Computational experiments are given in Section 3.4. Finally, the discussions and conclusions are presented in Section 3.5.

3.2 The Mixed Integer Linear Programming Model

To the best of our knowledge, the proposed MILP considers the phenomena of sequence dependent setup times for mixed-model assembly lines for the first time. Our MILP is a general model when considered some characteristics of assembly lines. Table 3.1 contains a comparison of our model with some recent publications. Some of them proposed mathematical models for MMALBP-I and some others attempted to formulate the SALBP by considering sequence dependent setup times. Moreover, this comparison covers some particular features of real world problems such as parallel workstations and zoning constraints. Since SALBP is special case of MMALBP, the proposed MILP model is able to solve SALBP as well as other mathematical models around MMALBP.

Table 3.1 Model characteristics considered in different researches

Research	Characteristics					Line Configuration
	MM	SM	SDST	ZC	PW	
Proposed Model	✓	✓	✓	✓	✓	Straight
Askin & Zhou (1997)	✓	✓			✓	Straight
Gokcen & Erel (1997)	✓	✓		✓		Straight
Gokcen & Erel (1998)	✓	✓				Straight
Sparling & Miltenburg (1998)	✓	✓				U-line
Erel & Gokcen (1999)	✓	✓				Straight
Vilarinho & Simaria (2002)	✓	✓		✓	✓	Straight
Bukchin et al. (2002)	✓	✓				Straight
Bukchin & Rabinowitch (2006)	✓	✓				Straight
Hop (2006)	✓	✓				Straight
Simaria & Vilarinho (2009)	✓	✓		✓		Two-sided
Ozcan & Toklu (2009)	✓	✓		✓		Two-sided
Kazemi at al. (2011)	✓	✓				U-line
Kara et al. (2011)	✓	✓				Straight
Andrés et al. (2008)		✓	✓			Straight
Scholl et al. (2008)		✓	✓			Straight
Özcan & Toklu (2010)		✓	✓			Straight
Scholl et al. (2011)		✓	✓			Straight
Seyed-Alagheband et al. (2011)		✓	✓			Straight
Hamta et al. (2012)		✓	✓			Straight

MM: Mixed-model; SM; Single-model; SDST; Sequence dependent setup times;
ZC: Zoning constraints; PW; Parallel Workstations

In order to describe the proposed model more clearly, the stated assumptions and defined notations (Table 3.2) are mentioned in the following.

Assumptions:

- A set of similar models of a product assembled on a straight line.

- The combined precedence diagram, which is a combination of all the precedence diagrams for all the models, contains the *N* tasks.
- It is allowed to create parallel workstations along the line, if there are tasks having processing times higher than cycle time due to demand.
- Zoning constraints can force/forbid the assignment of tasks to a specific workstation.
- A task can be assigned to only one workstation.
- Common tasks to several models must be performed on the same workstations.
- Processing time of a common task may be different among the models.
- Task processing times and sequence-dependent setup times between tasks are known deterministically.
- Processing and setup times are not dependent on the workstations.

Table 3.2 Model Notations

	Notation	Definition
Indices	N	Total number of tasks,
	M	Total number of models simultaneously assembled at the line,
	WS	Maximum number of workstations,
	i	Set of tasks $i \in \{1,2,\dots,N\}$,
	s	Set of stations $s \in \{1,2,\dots,WS\}$,
	m	Set of models $m \in \{1,2,\dots,M\}$,
Parameters	C	Cycle time,
	$maxp$	Maximum number of replicas for a workstation (Set as 2),
	α	A pre-defined proportion (%α) of the cycle time,
	$bigM$	A very large number,
	T_i	Processing time of task *i* on model *m*,
	$TT_{im} \in \{0,1\}$	Equals to 1 if processing time of task *i* is greater than zero for model *m* and 0 otherwise,
	FST_{ijm}	Forward set-up time between task *i* and *j* on model *m*,
	BST_{ijm}	Backward set-up time between task *i* and *j* on model *m*,
	$PR_{ij} \in \{0,1\}$	Equals to 1 if task *i* must precede task *j* and 0 otherwise,
	$ZP_{ij} \in \{0,1\}$	Equals to 1 if tasks *i* and *j* must be assigned to the same workstation, 0 otherwise,
	$ZN_{ij} \in \{0,1\}$	Equals to 1 if tasks *i* and *j* must be assigned to different workstations, 0 otherwise,
Decision Variables	$Y_{is} \in \{0,1\}$	Equals to 1 if task *i is* assigned to workstation *s* and 0 otherwise,
	$A_s \in \{0,1\}$	Equals to 1 if station *s* is active, 0 otherwise,
	$R_{sm} \in \{0,1\}$	Equals to 1 if workstation *s* is duplicated due to model *m* and 0 otherwise,
	$R_s \in \{0,1\}$	Equals to 1 if workstation *s* is duplicated, 0 otherwise,
	$w_{ijs} \in \{0,1\}$	Equals to 1 if task *i* precede task *j* at workstation *s* and 0 otherwise,
	$FS_{ijms} \in \{0,1\}$	Equals to 1 if task *j* directly follows task *i* on model *m* in the forward direction in workstation *s* and 0 otherwise,
	$BS_{ijms} \in \{0,1\}$	Equals to 1 if *i* is the last and *j* is the first tasks of model *m* in workstation *s* and 0 otherwise,
	N_{WS}	Total number of workstations including replicas.

The constraints of the model can be grouped into seven sets: assignment, precedence, zoning, workstation parallelization, sequence dependency, capacity, and stations. All these sets of constraints are explained in details in the following subsections.

3.2.1 Assignment Constraints

This set of constraints ensures the assignment of each task to exactly one workstation and can be written as follows:

$$\sum_{s=1}^{WS} Y_{is} = 1 \qquad i \in \{1, \dots, N\} \qquad (3.1)$$

3.2.2 Precedence Constraints

A task can only be assigned if all its predecessors were assigned to an earlier station or to the current station. This assignment restriction ensures processing a task after the completion of all its predecessors, and this set of constraints can be expressed as below:

$$bigM \times \left(1 - Y_{is} \times PR_{ij}\right) + \sum_{t|(t \geq s)}^{WS} Y_{jt} \geq 1 \qquad i, j \in \{1, \dots, N\}; s \in \{1, \dots, WS\} \quad (3.2)$$

3.2.3 Zoning Constraints

Zoning constraints are used to force or forbid the assignment of different tasks into the same station. The forcing set is called as positive (compatible) zoning constraints and verified by the set of constraints (3.3), while the forbidding set is called as negative (incompatible) zoning constraints and guaranteed by the set of constraints (3.4).

$$Y_{js} + bigM \times \left(1 - \left(Y_{is} \times ZP_{ij}\right)\right) \geq 1 \qquad i,j \in \{1,\dots,N\}; s \in \{1,\dots,WS\} \qquad (3.3)$$

$$Y_{js} - bigM \times \left(1 - \left(Y_{is} \times ZN_{ij}\right)\right) \leq 0 \qquad i,j \in \{1,\dots,N\}; s \in \{1,\dots,WS\} \qquad (3.4)$$

3.2.4 Workstation Parallelization Constraints

Total processing time of tasks assigned to a workstation defines the workload of the relevant workstation and it is not allowed to exceed the workstation's capacity defined by the cycle time. Some demand scenarios may cause to have some tasks greater processing times than a certain proportion (α %) of the cycle time. In such situations the workload restriction must be relaxed in such a way that two or more identical replicas of a workstation can perform the same set of tasks. Our proposed model allows paralleling a workstation if it performs a task with processing time larger than a certain proportion (α %) of the cycle time for at least one of the models. The set of constraints (3.5) determines which model requires parallelization (due to the assigned tasks processing times) and so the set of constraints (3.6) creates the parallel workstation for this model.

$$R_{sm} - bigM \times \sum_{i|(T_{im} > \propto * C)}^{N} Y_{is} \leq 0 \qquad s \in \{1,\dots,WS\};\ m \in \{1,\dots,M\} \qquad (3.5)$$

$$R_{sm} \geq Y_{is} \qquad i \in \{1,\dots,N\} | T_{im} > \propto * C;\ s \in \{1,\dots,WS\};\ m \in \{1,\dots,M\} \qquad (3.6)$$

In the same way, the set of constraints (3.7) ensures the parallelization of a workstation if parallelization required for at least one of the models in any workstation. So, the set of constraints (3.8) creates parallel one of this workstation in general.

$$R_s - bigM \times \sum_{m=1}^{M} R_{sm} \leq 0 \qquad s \in \{1,\dots,WS\} \qquad (3.7)$$

$$R_s \geq R_{sm} \qquad s \in \{1, \dots, WS\};\ m \in \{1, \dots, M\} \qquad (3.8)$$

3.2.5 Sequence Dependency Constraints

Sequence dependency constraints were based on three decision variables; w_{ijs}, FS_{ijms}, BS_{ijms}. The variable of w_{ijs} is used to determine the performing order of tasks (sequence of tasks) in a workstation. As we have considered sequence dependent setup times we need to determine immediate follower of each task in a workstation. Therefore, extra decision variables have been defined for determining the immediate followers of tasks whiles, FS_{ijms} and BS_{ijms} are used for the immediately following tasks in forward direction and for backward direction (transition from the last to the first tasks) in any workstation *s*. As a result of these types of decision variables, sequence dependency constraints may be classified into two groups: constraints (3.9-3.14) and correlations (a-i) determining the sequences of tasks, constraints (3.15-3.26) and correlations (j-u) determining the setup operations tasks.

3.2.5.1 Constraints Sets for the Sequences of Tasks

Considering a workstation *s*, the tasks *i*, *j*, *k*, and *l* are executed on a work-piece in this workstation. From the sequence dependent point of view it is necessary to determine the sequence of tasks in this workstation. As pointed out in Figure 3.1, it is possible to derive the sequence of these tasks due to the variable w_{ijs}. In other words, the variable w_{ijs} provides us the positions of tasks in a workstation.

The following correlations ensure that two tasks would be ordered if both of them have been assigned to the same workstation. These sets of correlations prevent ordering tasks in two situations for a workstation; only one of them assigned to the related workstation (the correlations *a*, and *b*), none of them assigned to the related workstation (correlations *c*).

$$w_{ijs} + w_{jis} - bigM \times \left(1 - Y_{is} + Y_{js}\right) \leq 0 \qquad i, j \in \{1, \dots, N\}; s \in \{1, \dots, WS\} \qquad (a)$$

$$w_{ijs} + w_{jis} - bigM \times \left(1 + Y_{is} - Y_{js}\right) \leq 0 \qquad i, j \in \{1, \dots, N\}; s \in \{1, \dots, WS\} \qquad (b)$$

$$w_{ijs} + w_{jis} - bigM \times \left(Y_{is} + Y_{js}\right) \leq 0 \qquad i, j \in \{1, \dots, N\}; s \in \{1, \dots, WS\} \qquad (c)$$

$$w_{iis} = 0 \qquad i \in \{1, \dots, N\}; s \in \{1, \dots, WS\} \qquad (3.9)$$

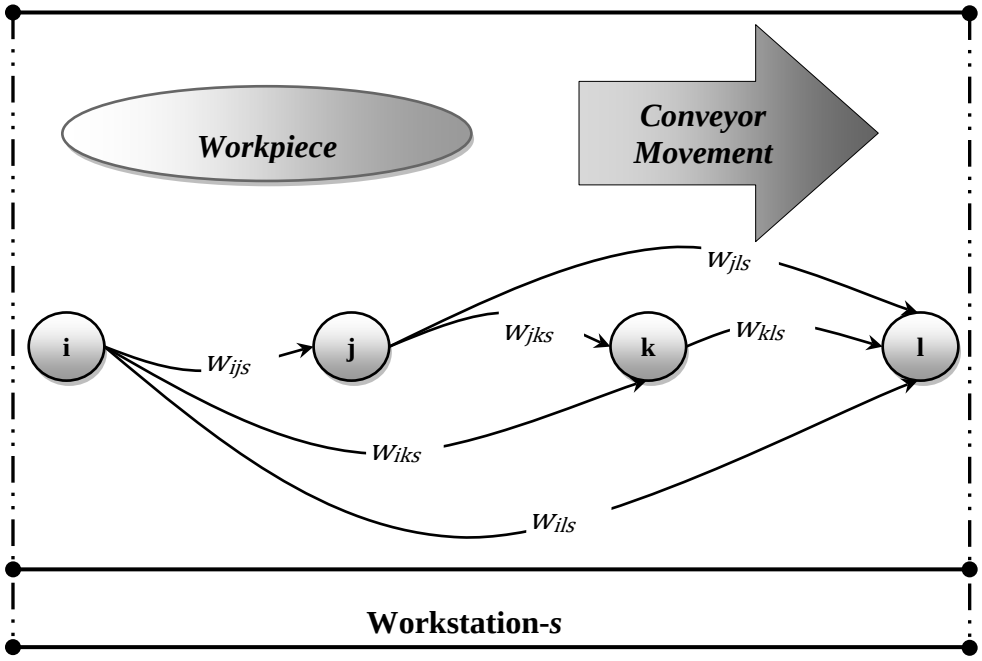


Figure 3.1 Sequence of tasks in a workstation

Remark 1: Considering F number of assigned tasks for any workstation s, to guarantee the ordering of tasks in a right way the variable w_{ijs} should be provided to take a value 1 in necessary situations. In what follows, we provide the sufficient and necessary conditions to tasks orders constraints by two cases in constraint sets (3.10-3.13) and constraints (3.14). The constraint sets (3.12) and (3.13) ensure that any two tasks would be ordered if both of them have been assigned to the same workstation.

$$w_{ijs} + w_{jis} + bigM \times \left(2 - Y_{is} - Y_{js}\right) \geq 1 \qquad i = 1, \dots, N; j \in \{1, \dots, N\} | j \neq i; s \in \{1, \dots, WS\} \qquad (3.10)$$

$$w_{ijs} + w_{jis} - bigM \times \left(2 - Y_{is} - Y_{js}\right) \leq 1 \qquad i = 1, \dots, N; j \in \{1, \dots, N\} | j \neq i; s \in \{1, \dots, WS\} \qquad (3.11)$$

It must be noted that any two tasks have to be ordered due to their precedence relations. The set of constraints (3.12) guarantees the mentioned precedence relations between tasks in any workstation.

$$w_{ijs} + bigM \times \left(3 - Y_{is} - Y_{js} - PR_{ij}\right) \geq 1 \quad i,j \in \{1,\dots,N\}; s \in \{1,\dots,WS\} \quad (3.12)$$

The aforementioned constraints determine the performing order of tasks in any workstation. As realized by constraint set (3.13), if task *i* has been performed before task *k* and task *k* has been performed before task *j*, then task *i* would be performed before task *j* too.

$$w_{ijs} + bigM \times \left(2 - w_{iks} - w_{kjs}\right) \geq 1 \quad i,j,k \in \{1,\dots,N\}; s \in \{1,\dots,WS\} \quad (3.13)$$

Lemma 1: If task *i* is in position *f* of the tasks order of the workstation *s* where *F* number of tasks are assigned to workstation *s* then:

$$\sum_{j=1}^{N} w_{ijs} = F - f \quad f \in \{1,\dots,F\}; i \in \{1,\dots,N\} \quad (d)$$

Proof: Considering *F* number of tasks in a workstation, if any task is in position *f*:

$$f = 1 \quad \rightarrow \quad \sum_{j=1}^{N} w_{ijs} = F - 1 \quad (e)$$

$$f = 2 \quad \rightarrow \quad \sum_{j=1}^{N} w_{ijs} = F - 2 \quad (f)$$

.

.

.

$$f = F - 1 \quad \rightarrow \quad \sum_{j=1}^{N} w_{ijs} = F - (F - 1) \quad (g)$$

$$f = F \quad \rightarrow \quad \sum_{j=1}^{N} w_{ijs} = F - F \quad (h)$$

Due to the aforementioned lemma we can also derive correlation (i).

$$\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ijs} = \sum_{i|(i\leq F-1)}^{N} i \qquad s = 1, \dots, WS \qquad (i)$$

The total number of the tasks in any workstation ($\sum_{k=1}^{N} Y_{ks}$) must be considered to provide assigning necessary performing orders between tasks. Within this context, the set of constraints (3.14) provides necessary number of assigned orders between tasks in any workstation. Due to the correlation (*i*), and substituting $F \leftarrow \sum_{k=1}^{N} Y_{ks}$ we can provide the set of constraints (3.14) for the s^{th} workstation:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ijs} = \sum_{i|\left(i<\sum_{k=1}^{N} Y_{ks}\right)}^{N} i \qquad s \in \{1, \dots, WS\} \qquad (3.14)$$

3.2.5.2 Constraints Sets for the Setup Operations

The variable w_{ijs} is not sufficient enough while determining the setup operations in a workstation. As pointed out in Figure 3.2, two types of setup operations exist in a workstation, forward and backward setup operations (Scholl et al., 2011). A forward setup operation occurs when a task *j* is performed directly after task *i* in the same cycle at the same workstation. A backward setup operation occurs between the last task and the first task of a workstation. Whenever the last task at the work-piece of cycle *p* is completed in a workstation, the employee has to move to the next work-piece of cycle *p+1*. For that reason, the variables of FS_{ijms} (for forward setups) and BS_{ijms} (for backward setups) are defined in order to determine the setup operations.

Remark 2: Considering *F* number of assigned tasks for any workstation *s* to determine the sequence of required setup operations we define a variable S_{ijs} which would be provided to take a value 1 if task *j* would be performed immediately after task *i* in the s^{th} workstation. Whiles, the mentioned setup operations are cyclic in any workstation.

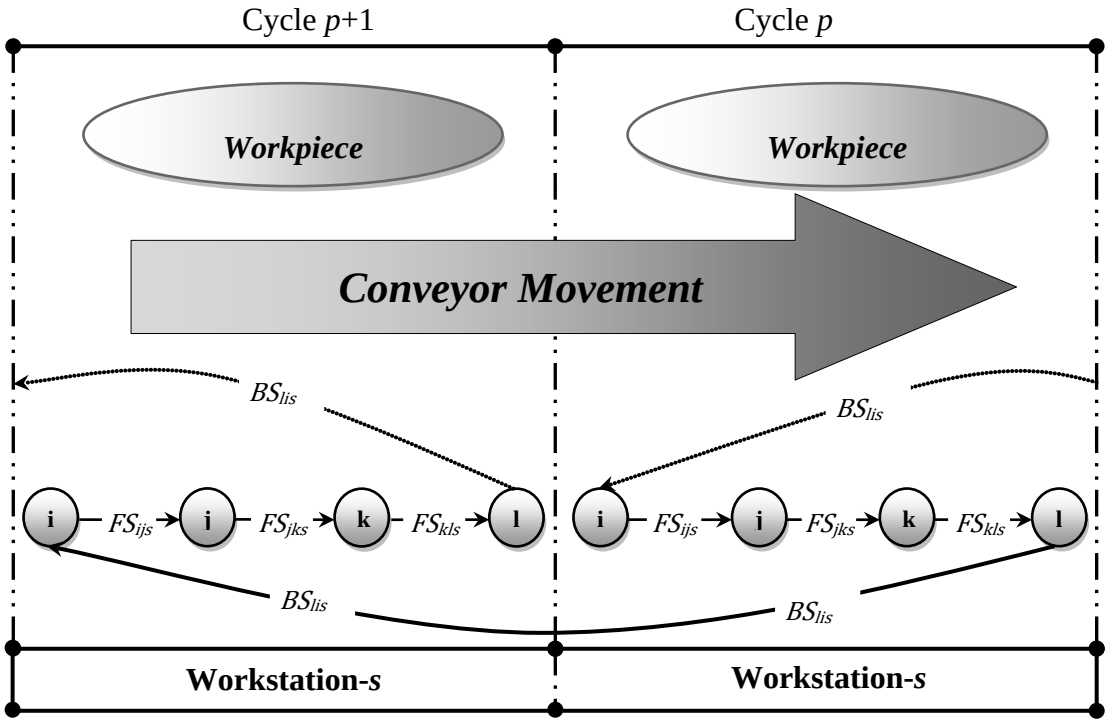


Figure 3.2 Forward and backward setup operations in a workstation

Lemma 2: In any workstation the total number of setup operations is equal to the total number of assigned tasks (F) to the related workstation (s) as:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} S_{ijs} = F \qquad s \in \{1, \dots, WS\} \qquad (j)$$

Proof:

$$F = 1 \quad \rightarrow \quad \sum_{j=1}^{N} S_{1js} = 1 \quad \Rightarrow \quad \sum_{i=1}^{1}\sum_{j=1}^{N} S_{ijs} = 1 \qquad (k)$$

$$F = 2 \quad \rightarrow \quad \left.\begin{matrix} \sum_{j=1}^{N} S_{1js} = 1 \\ \sum_{j=1}^{N} S_{2js} = 1 \end{matrix}\right\} \Rightarrow \quad \sum_{i=1}^{2}\sum_{j=1}^{N} S_{ijs} = 2 \qquad (l)$$

$$F = 3 \quad \rightarrow \quad \left.\begin{matrix} \sum_{j=1}^{N} S_{1js} = 1 \\ \sum_{j=1}^{N} S_{2js} = 1 \\ \sum_{j=1}^{N} S_{3js} = 1 \end{matrix}\right\} \Rightarrow \quad \sum_{i=1}^{3}\sum_{j=1}^{N} S_{ijs} = 3 \qquad (m)$$

$$F = N \quad \rightarrow \quad \left.\begin{array}{c} \sum_{j=1}^{N} S_{1js} = 1 \\ \sum_{j=1}^{N} S_{2js} = 1 \\ \vdots \\ \sum_{j=1}^{N} S_{(N-1)js} = 1 \\ \sum_{j=1}^{N} S_{Njs} = 1 \end{array}\right\} \Rightarrow \sum_{i=1}^{N} \sum_{j=1}^{N} S_{ijs} = \mathrm{N} \qquad (n)$$

On the other hand, if a task is not performed, then there would not be any setup operation in any workstation related to this task. The set of constraints (3.15) ensure the mentioned conditions in the model.

$$FS_{ijms} + BS_{ijms} - bigM \times \left(TT_{im} \times TT_{jm}\right) \leq 0$$
$$i, j \in \{1, \dots, N\}; m \in \{1, \dots, M\}; s \in \{1, \dots, WS\} \qquad (3.15)$$

If a backward setup operation has been assigned between any tasks there would not be any forward setup operation. The set of constraints (3.16) ensures this restriction.

$$FS_{ijms} - bigM \times \left(1 - BS_{ijms}\right) \leq 0$$
$$i, j \in \{1, \dots, N\}; m \in \{1, \dots, M\}; s \in \{1, \dots, WS\} \qquad (3.16)$$

As mentioned previously a forward setup operation occurs when a task executed directly after another task in the same cycle of a workstation. Similar to correlations (a, b, and c) the following correlations ensure that two tasks would be ordered if both of them have been assigned to the same workstation.

$$FS_{ijms} + FS_{jims} - bigM \times \left(1 - Y_{is} + Y_{js}\right) \leq 0$$
$$i, j \in \{1, \dots, N\}; s \in \{1, \dots, WS\} \qquad (o)$$

$$FS_{ijms} + FS_{jims} - bigM \times \left(1 + Y_{is} - Y_{js}\right) \leq 0$$
$$i, j \in \{1, \dots, N\}; s \in \{1, \dots, WS\} \qquad (p)$$

$$FS_{ijms} + FS_{jims} - bigM \times \left(Y_{is} + Y_{js}\right) \leq 0$$
$$i, j \in \{1, \dots, N\}; s \in \{1, \dots, WS\} \qquad (r)$$

It must be noted that each task in any workstation would have at most one immediate follower. Set of constraints (3.17) ensures this situation for all tasks for all workstations.

$$\sum_{j=1}^{N} \sum_{s=1}^{WS} FS_{ijms} \leq 1 \qquad i \in \{1, \dots, N\}; m \in \{1, \dots, M\} \qquad (3.17)$$

If a forward setup operation was done between tasks *i(j)* and *j(i)* then any forward setup operation between task *j(i)* and *i(j)* must be prohibited. That is to say it is possible to do only one forward setup operation between any pair of tasks. The mentioned restriction was provided by the constraints set of (3.18).

$$FS_{ijms} + FS_{jims} \leq 1 \qquad i, j \in \{1, \dots, N\}; m \in \{1, \dots, M\};\ s = \{1, \dots, WS\} \qquad (3.18)$$

Set of constraints (3.19) prevents assigning any forward setup operation between a task and itself. Forward setup operations may be executed between any different two tasks in the same workstation.

$$\sum_{s=1}^{WS} FS_{iims} = 0 \qquad i \in \{1, \dots, N\}; m \in \{1, \dots, M\} \qquad (3.19)$$

Considering Lemma 1 we can determine adjacent tasks in any workstation through the variable w_{ijs}. Set of constraints (3.20) provides us adjacent tasks in any workstation, so provides us the required forward setup operations.

$$FS_{ijms} + bigM \times \left(4 - Y_{is} - Y_{js} - TT_{im} - TT_{jm}\right) +$$
$$bigM \times \left| \sum_{k=1}^{N} \left(w_{iks} \times TT_{km}\right) - \sum_{l=1}^{N} \left(w_{jls} \times TT_{lm}\right) - 1 \right| \geq 1$$
$$i, j \in \{1, \dots, N\}; m \in \{1, \dots, M\}; s \in \{1, \dots, WS\} \qquad (3.20)$$

Backward setup operations occur between the last and the first tasks of any workstation as mentioned previously. Similar to correlations (a, b, and c) and (o, p, and r) the following correlations ensure that there may be a backward setup operation between any two tasks if both of them have been assigned to the same workstation.

$$BS_{ijms} + BS_{jims} - bigM \times \left(1 - Y_{is} + Y_{js}\right) \leq 0$$
$$i, j \in \{1, \dots, N\}; s \in \{1, \dots, WS\} \qquad (s)$$

$$BS_{ijms} + BS_{jims} - bigM \times \left(1 + Y_{is} - Y_{js}\right) \leq 0$$
$$i, j \in \{1, \dots, N\}; s \in \{1, \dots, WS\} \qquad (t)$$

$$BS_{ijms} + BS_{jims} - bigM \times \left(Y_{is} + Y_{js}\right) \leq 0$$
$$i, j \in \{1, \dots, N\}; s \in \{1, \dots, WS\} \qquad (u)$$

Due to the similarities between correlations (a, b, and c), (o, p, and r), and (s, t, and u) the sets of constraints (3.21), (3.22) and (3.23) have been generated as follows.

$$w_{ijs} + w_{jis} + FS_{ijms} + FS_{jims} + BS_{ijms} + BS_{jims} - bigM \times \left(1 - Y_{is} + Y_{js}\right) \leq 0$$
$$i, j \in \{1, \dots, N\}; m \in \{1, \dots, M\}; s \in \{1, \dots, WS\} \qquad (3.21)$$

$$w_{ijs} + w_{jis} + FS_{ijms} + FS_{jims} + BS_{ijms} + BS_{jims} - bigM \times \left(1 + Y_{is} - Y_{js}\right) \leq 0$$
$$i, j \in \{1, \dots, N\}; m \in \{1, \dots, M\}; s \in \{1, \dots, WS\} \qquad (3.22)$$

$$w_{ijs} + w_{jis} + FS_{ijms} + FS_{jims} + BS_{ijms} + BS_{jims} - bigM \times \left(Y_{is} + Y_{js}\right) \leq 0$$
$$i, j \in \{1, \dots, N\}; m \in \{1, \dots, M\}; s \in \{1, \dots, WS\} \qquad (3.23)$$

As mentioned previously, backward setup operations occur between the last task and the first task assigned to the same workstation. After performing the last task, the worker has to move to the first task assembled in the same workstation. Therefore,

each workstation would have just one backward setup operation. This restriction is realized by the sets of constraints (3.24).

$$\sum_{i=1}^{N}\sum_{j=1}^{N} BS_{ijms} \leq 1 \qquad m \in \{1,\dots,M\}; s \in \{1,\dots,WS\} \tag{3.24}$$

Considering Lemma 1 we can determine the first and the last tasks in any workstation thanks to the variable w_{ijs}. So, the third and fourth terms in the left side of constraint set (3.25) provide us to determine the backward setup operation between the last and the first tasks in any workstation.

$$BS_{ijms} + bigM \times \left(4 - Y_{is} - Y_{js} - TT_{im} - TT_{jm}\right) + bigM \times \left(\sum_{k=1}^{N} w_{iks} \times TT_{km}\right) + bigM \times \left|\sum_{l=1}^{N} (Y_{ls} \times TT_{ls}) - \sum_{n=1}^{N} \left(w_{jns} \times TT_{ns}\right) - 1\right| \geq 1$$
$$i,j \in \{1,\dots,N\}; m \in \{1,\dots,M\}; s \in \{1,\dots,WS\} \tag{3.25}$$

3.2.6 Capacity Constraints

Capacity constraints ensure that the workload of a workstation does not exceed the pre-defined cycle time for all the models being assembled. The workload of a workstation consists of the summation of the tasks processing times, forward and backward setup operations times within a workstation. The set of constraints (3.26) ensures this capacity restriction if a workstation has more than one task.

$$\sum_{i=1}^{N}\left(Y_{is} \times T_{im} + \sum_{j=1}^{N}\left(FS_{ijms} \times FST_{ijm} + BS_{ijms} \times BST_{ijm}\right)\right) \leq C \times \left(1 + R_s \times (maxp - 1)\right)$$
$$s \in \{1,\dots,WS\};\ m \in \{1,\dots,M\} \tag{3.26}$$

3.2.7 Stations Constraints

It is necessary to utilize the same number of workstations for all models. That is to say, if any workstation has any task from any model it would be considered as an active workstation for the other models too. Constraint sets (3.27) and (3.28) provide us to determine the active workstations due to their number of assigned tasks.

$$A_s + bigM \times (1 - Y_{is}) \geq 1 \qquad i \in \{1, \dots, N\};\ \ s \in \{1, \dots, WS\} \qquad (3.27)$$

$$A_s - bigM \times \sum_{i=1}^{N} Y_{is} \leq 0 \qquad s \in \{1, \dots, WS\} \qquad (3.28)$$

Set of constraints (3.29) provides the active workstations to be in an ordered sequence. So, if a workstation has been activated its preceding workstations should have been activated already.

$$\left| \left(\sum_{s=1}^{t} A_s \right) - t \right| - bigM \times (1 - A_t) \leq 0 \qquad t \in \{1, \dots, WS\} \qquad (3.29)$$

3.2.8 Objective Function

The aim of our proposed MILP model is to minimize the total number of workstations (N_{WS}). Where, N_{WS} determines the total number of active workstations as well as their parallels.

$$N_{WS} = \left(\sum_{s=1}^{\sum_{t=1}^{WS} A_t} R_s + A_s \right) \qquad (3.30)$$

3.3 An Illustrative Example

In this section, a numerical example with the following characteristics has been used to illustrate the proposed MILP model.

- A line is used to simultaneously assemble two models A and B over a planning horizon of 480 time units where, the demands for each model are 20 and 28 units, respectively. Thus, the cycle time (C) is equal to $480 \div (20 + 28) = 10$.
- Precedence diagram for 11 numbers of tasks is the same as Figure 2.2 (see Section 2.3.2.1).
- The task processing times for the models A and B are given in Table 3.3.
- Tasks 7 and 8 cannot be executed on the same workstation.
- A workstation can be replicated if it performs a task with a processing time greater than cycle time.

Table 3.3 Task processing times of the numerical example

Task	1	2	3	4	5	6	7	8	9	10	11
T_A	3.0	3.1	1.9	8.4	3.1	11.2	8.8	8.7	2.5	5.2	4.4
T_B	0.0	3.1	1.9	8.4	3.1	9.9	0.0	8.7	2.5	0.0	4.4

T_A: Task time for model A; T_B: Task time for model B

As it can be seen from Table 3.3, all tasks would be performed for model A whiles, tasks 1, 7, and 10 are not required for model B. Since, tasks are assigned according to combined precedence diagram; this situation has no effect on the decision variables of Y_{is} and w_{ijs}, however, it directly influences the decision variables of FS_{ijms} and BS_{ijms}. Thus, this situation must be taken into consideration in order to determine the correct values of FS_{ijms} and BS_{ijms} regarding to the required setup operations. The sets of constraints (3.20) and (3.26) have been developed within this context in order to properly determine the required setup operations for all models in any workstation according to the task assignments. Table 3.4 represents the tasks assignments for the numerical example. Since, processing times of tasks 1,

7, and 10 have been considered as zero for model B. There are some conditions that the setup operations and their sequences differ due to the model type in any workstations. Hence, tasks have been assigned to the workstations by considering both model types. Moreover, tasks are performed according to the given sequences in Table 3.4, where, these sequences have been outlined from the decision variable w_{ijs}.

Table 3.4 Task assignments of the numerical example

Workstation	Tasks	Task Sequence	
		Model A	Model B
1	1, 2	1, 2	2
2	8	8	8
3	4	4	4
4	5, 3, 9	5, 3, 9	5, 3, 9
5	10, 6	10, 6	6
6	7	7	-
7	11	11	11

Also, considering the task performing sequences in Table 3.4, the setup operations have been derived for all models in all workstation as illustrated in Table 3.5.

Table 3.5 Setup operations of the numerical example

Workstation	Forward Setup Operations				Backward Setup Operations			
	Model A		Model B		Model A		Model B	
	Tasks		Tasks		Tasks		Tasks	
	From	To	From	To	From	To	From	To
1	1	2	-		2	1	2	2
2	-		-		8	8	8	8
3	-		-		4	4	4	4
4	5 3	3 9	5 3	3 9	9	5	9	5
5	10	6	-		6	10	6	6
6	-		-		7	7	-	
7	-		-		11	11	11	11

As pointed out in Table 3.5, each workstation contains exactly one backward setup operation for each model if a workstation contains at least one task for each model.

3.4 Computational Experiments

In order to evaluate the performance of the proposed MILP model a set of benchmark problems have been used. Some of the problems (problems 1-4) have been taken from the literature and some of them (problems 5-10) have been generated randomly. Problems 1-4 were used originally by Vilarinho & Simaria (2002) and Akpınar & Bayhan (2011). For the other 6 problems, precedence relations were taken from the existing literature as presented in Table 3.6, and task processing times and setup times were generated randomly. The main characteristics of the test problems are given in Table 3.6 where, *N*, *M*, and *C* denote the number of tasks of the combined precedence diagram, the number of models, and the cycle time of the assembly line, respectively. All problems were enriched by adding sequences dependent setup times between tasks.

Table 3.6 Main characteristics of the test problems

Problem No	N	M	C	Precedence Relations	Problem No	N	M	C	Precedence Relations
1	8	2	10	Bowman	6	12	3	10	Ponnambalam et al. (1999)
2	8	3	10	Bowman	7	14	2	10	Simaria & Vilarinho (2009)
3	11	2	10	Gokcen & Erel	8	14	3	10	Simaria & Vilarinho (2009)
4	11	3	10	Gokcen & Erel	9	15	2	10	Buckhin et al. (2002)
5	12	2	10	Ponnambalam et al. (1999)	10	15	3	10	Buckhin et al. (2002)

N: Number of tasks; M: number of models; C: Cycle time

As mentioned in Scholl et al. (2011), setup times have been generated in two types: forward and backward setups. Where, both forward and backward setup times have been generated according to a uniform discrete distribution $U[0, 0.5 \times (\min T_i)]$ (Andrés et al., 2008). Moreover, to fulfill the triangle inequality mentioned

by Scholl et al. (2011) Equation 3.31 have been considered in setup times generations.

$$FST_{ih} + T_h + FST_{hj} \geq FST_{ij} \qquad and \qquad FST_{ih} + T_h + BST_{hj} \geq BST_{ij}$$
$$i, j, h \in \{1, \dots, N\} \qquad (3.31)$$

Moreover, considering any task may be a single element of a workstation, the following pre-condition have to be satisfied for all tasks.

$$T_i + BST_{ii} \leq C \qquad i \in \{1, \dots, N\} \qquad (3.32)$$

The proposed MILP model has been coded in IBM ILOG CPLEX 12.1.0. We have attempted to solve the test problems on Core(TM) i7-3820 CPU 3.60GHz personal computer and the run time has been limited up to 5 hours for each problem. The computational results are given in Table 3.7.

Table 3.7 Summary of the computational results on test problems

Problem No	Objective Value		NV	NBV	NC	CPU
	Optimum	Feasible				
1	5	-	6930	2656	17576	46.49
2	9	-	9882	3688	22640	1.73
3	8	-	17778	6820	49478	114.36
4	7	-	25533	9493	62634	643.17
5	9	-	23018	8832	65928	266.20
6	10	-	33110	12300	83004	240.96
7	10	-	36402	13972	110096	5622.37
8	11	-	52488	19474	137256	4810.45
9	No feasible solution		44702	17160	138735	-
10	No feasible solution		64517	23925	172110	-

NV: Number of Variables; NBV: Number of Binary Variables; NC: Number of Constraints; CPU: Computational Time

As it can be seen from Table 3.7, the proposed MILP model found optimal solutions for problems 1-8, however it was not able to solve problems 9 and 10 to optimality. From these computational results we can conclude that our MILP is able

to solve up to 14 tasks instances to optimality. Furthermore, the computational times of finding the optimal solutions depend on the number of variables and the number of constraints as well as the problem data (Table 3.7).

3.5 Chapter Conclusions

In this chapter we aimed at developing a mixed-integer linear programming model for type-I mixed-model assembly line balancing problem enriched with the sequences dependent setup times between tasks for the first time. The MILP provides us the formally formulation of MMALBPS-I. Moreover, our MILP can solve the problem with and without sequence dependent setup times, parallel workstation assignments, and zoning constraints. Since, the SALBP-I is a special case of MMALBP-I, our MILP is able to solve SALBP-I with and without the aforementioned characteristics. Thus, we can conclude that our MILP is a general model for some of the assembly line balancing problems.

Due to the complex nature of the model, it is not able to solve optimality as the problem size increased. For that reason, meta-heuristic approaches were developed in order to tackle the problem and the developed algorithms are explained in the following chapters in details.

CHAPTER FOUR
HYBRID ANT COLONY OPTIMIZATION-GENETIC ALGORITHM FOR MMALBPS-I

4.1 Chapter Introduction

Current markets are characterized as consumer-centric resulted in a growing trend for higher product variability. As a result of this, high-mix/low-volume manufacturing strategies substitute for low-mix/high-volume manufacturing strategies. Single-model assembly lines, which are the most suited production systems for low variety demand scenarios, are not able to respond the requirements of this new type of manufacturing strategies anymore. Therefore, manufacturers prefer producing one model with different features or several models on a single assembly line (mixed-model assembly line, which was handled by Thomopoulos (1967) for the first time in the literature) in order to avoid the high cost to build and maintain an assembly line for each model. In the relevant literature several approaches have been presented to cope with MMALBP-I. These approaches can be divided into three groups: mathematical programming, heuristics/meta-heuristics and hybrid approaches. Besides assembly line balancing problems, hybrid algorithms were used for solving several combinatorial optimization problems and usually developed by integrating meta-heuristics with problem specific heuristic algorithms or meta-heuristics with meta-heuristics. On the other hand, hybrid algorithms have showed their ability to provide local optima of high quality. For more comprehensive reviews on hybrid meta-heuristics the reader can refer to the papers of Preux & Talbi (1999), Talbi (2002), Raidl (2006) and Blum et al. (2011). In this study, we attempt to hybridize ACO (Dorigo et al., 1991; Dorigo et al., 1996; Dorigo & Gambardella, 1996; Dorigo & Gambardella, 1997; Dorigo et al., 1999; Stützle & Dorigo, 1999) with GA (Holland, 1975; Goldberg, 1989) in parallel (operation parallelization), which belongs to the class of parallel hybrid meta-heuristics (Crainic & Toulouse, 2003). These algorithms are sufficiently complex to provide powerful adaptive search approaches, and usually can be embedded with other approaches to speed up the search performance (Lee et al., 2008). The rationale why we attempt to hybridize

ACO with GA is to exploit the complementary character of different optimization strategies, that is, hybrids are believed to benefit from synergy (Blum et al., 2011). Viz, our proposed hybrid algorithm integrates the positive feedback mechanism and the satisfactory performance of ACO with the faster speed of GA. Thus, the proposed hybrid ACO-GA algorithm attempt to overcome the slower speed of ACO and the poor searching capability of GA, especially for large sized problems, by embedding GA into ACO as a local search. Furthermore, ACO-GA utilizes the synergy of GA as an improvement procedure and ACO as a constructive procedure.

The rest of this chapter[*] is organized as follows. The proposed hybrid ACO-GA algorithm is defined in Section 4.2. Comparative study is given in Section 4.3. Finally, the discussions and conclusions are presented in Section 4.4.

4.2 The proposed Hybrid ACO-GA Algorithm

The procedure of the proposed ACO-GA algorithm applies selection procedure, measurement of solution qualities, genetic algorithm, and pheromone evaporation and release strategies respectively for MMALBPS-I. The proposed hybrid ACO-GA algorithm is depicted in Figure 4.1. The algorithm starts by generating a pre-defined number of solutions by the task selection strategy and each solution turned as a chromosome. After that, the solution quality measures are determined. At this point, the obtained set of chromosomes is set as the initial population of GA and the solution qualities are set as their fitness values. Selection, crossover and mutation operators are applied to produce new chromosomes (offspring chromosomes). According to their fitness values, a strategy called elitism survives the best fit chromosomes to next generation. After a predefined termination criterion is met, the final generation releases a certain amount of pheromone. The hybrid ACO-GA algorithm repeats itself until a pre-specified number of iterations reached. All the operators used in the proposed ACO-GA algorithm are introduced in the following sub-sections in detailed.

[*] The work presented in this chapter is published in Akpınar et al.(2013)

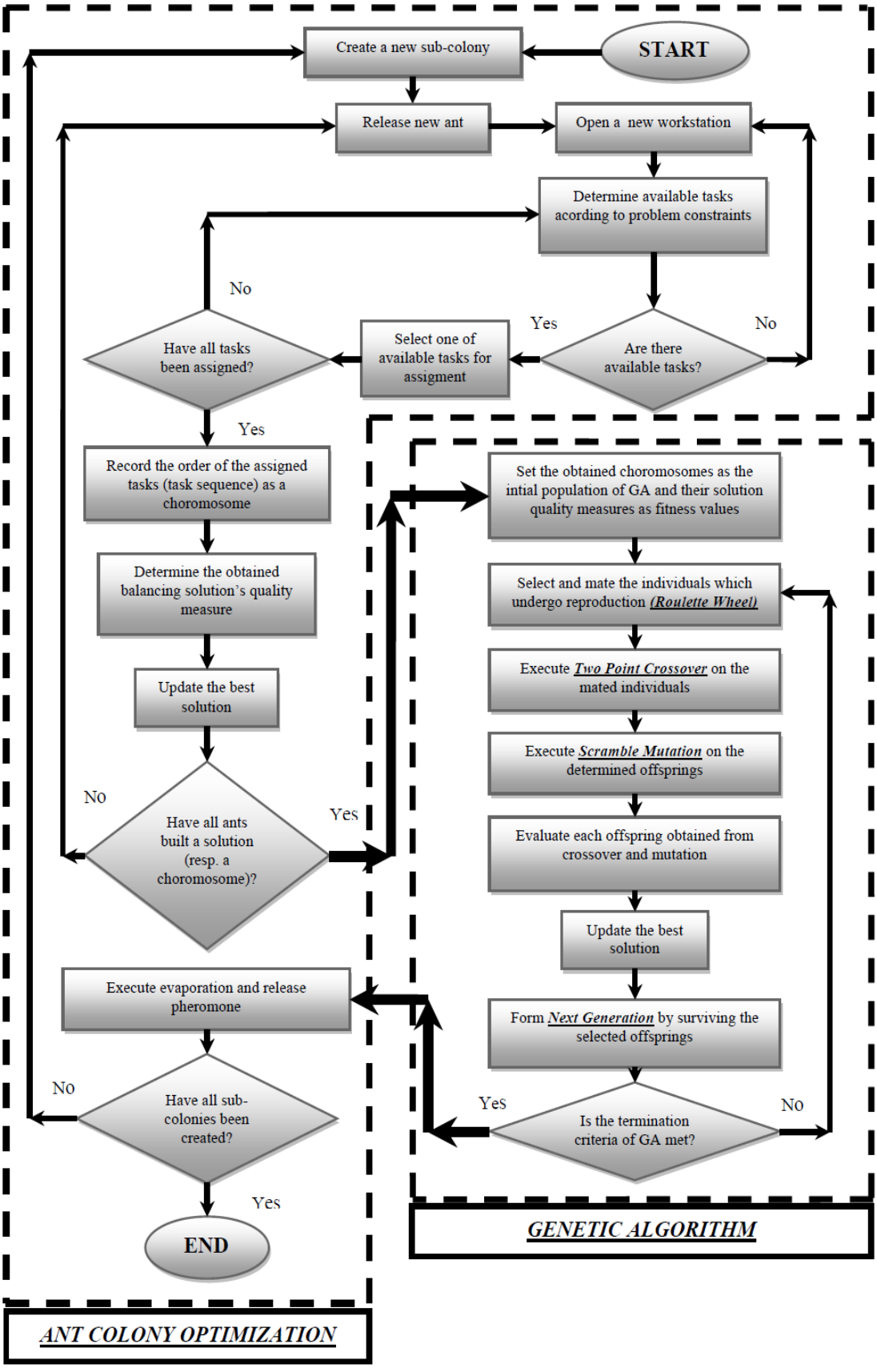


Figure 4.1 Flow diagram of the proposed hybrid ACO-GA

4.2.1 Task Selection Strategy

The probability of a task being selected, from the set of available tasks, is a function of: (1) the pheromone trail intensity between the previously selected task and each available task and (2) the information provided by the heuristic for each available task. This information is a priority rule that is assigned to each ant when the respective sub-colony is generated. The procedure uses some common static priority rules for the ALBP:

- 'Maximum positional weight',
- 'Maximum processing time' (for all models),
- 'Maximum average processing time',
- 'Maximum number of direct successors' and
- 'Maximum number of successor'.

In the current study the maximum weighted average processing time is used as the heuristic information due to the mixed-model nature of the problem.

The executed task selection rule (Vilarinho & Simaria, 2006) uses a random number *r* between 0 and 1 and three user defined parameters r_1, r_2 and r_3 such that $0 \leq r_1, r_2, r_3 \leq 1$ and $r_1 + r_2 + r_3 = 1$. The rule is given by:

$$j = \begin{cases} J_1 = \underset{j \in A_i^n}{argmax} \left\{ \left[\tau_{(i,j)}\right]^{\alpha} \times \left[\eta_j\right]^{\beta} \right\} & if\ r \leq r_1\ \ (exploi.) \\ J_2 : p_{(i,J_2)} = \dfrac{\left[\tau_{(i,J_2)}\right]^{\alpha} \times \left[\eta_{J_2}\right]^{\beta}}{\sum_{j \in A_i^n} \left(\left[\tau_{(i,j)}\right]^{\alpha} \times \left[\eta_j\right]^{\beta} \right)} & if\ r_1 < r \leq r_1 + r_2\ \ (bia.\ exp.) \\ J_3 : random\ selection\ of\ j \in A & if\ r_1 + r_2 < r \leq r_1 + r_2 + r_3 \end{cases} \quad (4.1)$$

where $\tau_{(i,j)}$is the pheromone trail intensity in the path 'selecting task *j* after selecting task *i*, η_j is the heuristic information of task *j* (e.g. the priority rule value for task *j*), A_i^n is the set of available tasks for ant *n* after the selection of task *i*, and α and β are parameters that determine the relative importance of pheromone intensity versus heuristic information.

The selection of a task from the set of available tasks is performed by one of three strategies:

- *Exploitation* selects the best task according to the values of $\left[\tau_{(i,j)}\right]^{\alpha} \times \left[\eta_j\right]^{\beta}$.
- *Biased exploration* selects a task according to a probability of $p_{(i,j)}$ as given by J_2.
- *Random selection* selects one task at random from the set of available tasks.

4.2.2 Solution Quality Measure

Finding the fittest solution after a predefined number of generations is the main purpose of the proposed hybrid ACO-GA. Therefore, the algorithm has to use an objective function which measures each solution's quality. The objective function (Vilarinho & Simaria, 2002) (Equation 4.2) is used by ACO-GA consists of three terms. The first term aims at minimizing the total number of workstations by minimizing the index of the workstation to which the last task is assigned, while, the second term balances the workload between the workstations, and the third term balances the workload within each workstation.

$$min\ Z = \sum_{k=1}^{S} k.X_{Nk} + \frac{S'}{S'-1}\sum_{m=1}^{M} q_m \sum_{k=1}^{S'}\left(\frac{s_{km}}{\sum_{l=1}^{S'} s_{lm}}\right)^2 + \frac{M}{S'(M-1)}\sum_{k=1}^{S'}\sum_{m=1}^{M}\left(\frac{q_m s_{km}}{S_k} - \frac{1}{M}\right)^2 \quad (4.2)$$

where S is the work station index that the last task is assigned, S' is the total number of workstations including replicas, X_{ik} is 1 if task i assigned to workstation k is 0 otherwise, M is the number of models assembled in the line, s_{km} is the idle time of workstation k due to model m, q_m is the overall proportion of the number of units of model m being assembled and S_k is the total proportional idle time of workstation k.

4.2.3 Genetic Algorithm

In this study, we use GA (similar to Akpınar & Bayhan, 2011) embedded into ACO as local search, by which we aim at improving the search capability of the proposed algorithm. GA is executed before pheromone release phase; hence obtained solutions release a certain amount of pheromone after GA refinement. For that reason we used an elitist strategy in GA while forming the next generations.

4.2.3.1 Roulette Wheel Selection

The proposed algorithm selects individuals for mating by using the best known selection strategy Roulette wheel (Holland, 1975), also known as fitness proportionate selection. Roulette wheel scales the fitness values of the members within the population as the total rescaled fitness values equals to 1. First, a uniform random number within the interval (0, 1) is generated (wheel is spun), and then the individual whose cumulative rescaled fitness value is greater than the generated number is selected as parent. The steps of the used roulette wheel selection strategy can be summarized in below:

I. Sum the fitness values of all the population members. Call this F_{sum}.

II. Divide the fitness values of all population members by F_{sum} in order to calculate expected values of each individual in the population.

III. Generate a uniform random number, R_s, between 0 and 1.

IV. Loop through the individuals in the population, summing the expected values, until the sum is greater than or equal to R_s. The individual whose expected value puts the sum over this limit is the one selected.

4.2.3.2 Two Point Crossover

In this study a two point crossover (Leu et al., 1994) which is particular to assembly line balancing problem is used. The classical two point crossover cuts

mated parents into three parts (head (H), middle (M) and tail (T)), by determining the cut points randomly. Offspring are created by swapping the middle parts of the parent chromosomes, i.e., parent-1, represented by $H_1M_1T_1$, recombines with parent-2, represented by $H_2M_2T_2$, in order to form the new children $H_lM_2T_1$ and $H_2M_lT_2$. In the assembly line balancing problem the situation is not so simple because of the precedence relations between tasks, which may result in feasibility problem. For that reason recombination must guarantee feasibility. The special two point crossover is applied as shown in Figure 4 which guarantees generating feasible individuals according to the precedence relations. Thus, the resulting offspring are always feasible.

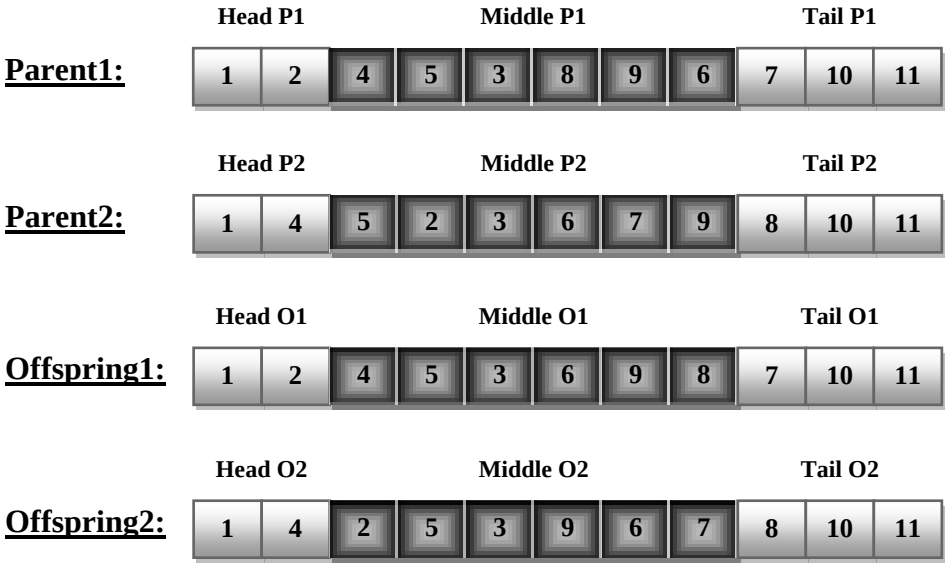


Figure 4.2 Recombination: Two point crossover

The first offspring keeps the head and the tail parts of the first parent. The middle part of the first offspring is filled in by adding the all missing tasks in the order in which they are contained in the second parent. The other offspring is built analogously based on the head and the tail parts of the second parent and its middle part is filled in by adding the missing tasks in the order in which they are contained in the first parent. Both of the generated offsprings become feasible as their middle part is also filled according to the precedence feasible order. The purpose of the two-point crossover is to conduct a neighborhood search; this is done by keeping the head

and tail of each child the same as its parent. The child should be “close” in fitness to its parents because only its middle genes have changed (Leu et al., 1994).

4.2.3.3 Scramble Mutation

Similar to recombination, mutation must also guarantee the feasibility because of the precedence relations. In this study, we also used a special mutation operator named as scramble mutation (Leu et al., 1994). First, a random point is selected for determining where the mutation will occur. After that, the head of the chosen parent is set as the head of the mutated offspring. Then, the mutation operator reconstructs the tail of the new child by using the procedure explained below. This procedure uses prohibit table (see Figure 4.3), and also guarantees the feasibility.

Task	Can Not Precede	Task	Can Not Precede
11	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	3	1, 2
10	1, 2, 3, 4, 5, 6, 7, 8, 9	2	1
6	1, 2, 3	5	4
7	1, 2, 3	1	-
8	4, 5	4	-
9	4, 5		

a. Original prohibit table

Task	Can Not Precede	Task	Can Not Precede
11	6, 7, 8, 9, 10	7	-
10	6, 7, 8, 9	8	-
6	-	9	-

b. Modified prohibit table

Task	Can Not Precede
11	10
10	-

c. Final prohibit table

Figure 4.3 Steps of the prohibit table

This procedure is performed by removing all references to head tasks in the prohibit table, and then randomly choosing a task from those in the table with no predecessor requirements. This new task is then added to the next locus in the chromosome and is removed from the prohibit table. The process continues until all tasks are assigned.

For example, consider parent (1-2-4-5-3-6-7-8-9-10-11) in Figure 4.4 and assume the mutation point is chosen to be after task 3. Then the head of the child will be 1-2-4-5-3. The rest of the child will consist of tasks 6, 7, 8, 9, 10, and 11, placed in random order, but in such a manner as not to violate precedence constraints. If the original prohibit table is as shown in Figure 4.3-a, then the prohibit table with all references to tasks in the head of the child will be as shown in Figure 4.3-b. Only tasks 6, 7, 8, and 9 can be selected to follow task 3 since they alone have no "cannot precede" tasks in the modified prohibit table; assume they are chosen randomly in the order 9, 7, 8, 6. Then the final prohibit table (with these deleted tasks) becomes as in Figure 4.3-c. Therefore, task 10 must be selected next, and 11 must be chosen last. The new child chosen with scramble mutation is 1-2-4-5-3-9-7-8-6-10-11 (see Figure 4.4).

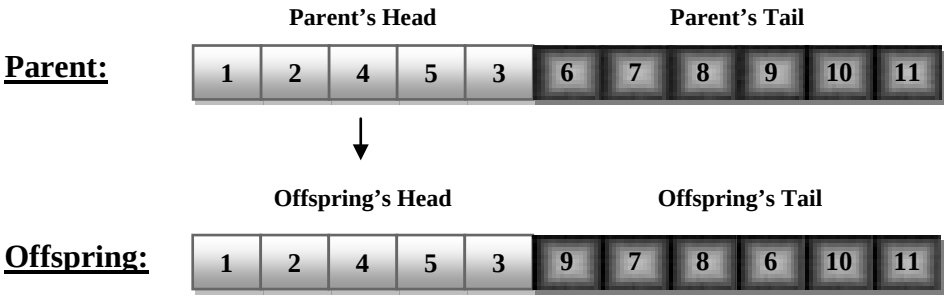


Figure 4.4 Scramble mutation

The purpose of mutation, unlike that of recombination (crossover), is to get out of a local search neighborhood and thus avoid the possibility of being trapped in a local optimum. Therefore, the goal of mutation is to change dramatically the order of the genes on the chromosome; scramble mutation does this. With scramble mutation only the head of the parent is maintained and the tail is reconstructed randomly in a manner that ensures feasibility (Leu et al., 1994).

4.2.3.4 Fitness Evaluation

The objective function as given by Equation 4.2 (see Section 4.2.2) is used also in the GA as fitness function.

4.2.3.5 New Generation

After each generation GA must decide with a replacement strategy which individuals are survived to next generation and which are not. The replacement strategy takes into account the fitness value of the individuals while selecting the survived individuals, who may be individuals from the current generation, offspring products of crossover or individuals who underwent mutation. In this study, the best fit individual is always survived to next generation and the other individuals are selected from the best fit offspring products of crossover and mutation. Until a predefined termination criterion met GA repeats itself and dispatches the last generation to ACO.

4.2.4 Pheromone Release Strategy

The pheromone release strategy is based on the one used by Dorigo et al. (1996). At the end of each sub-colony's iteration, all balancing solutions provided by the ants have their objective function values computed. It is at this point that the pheromone trail intensity is updated. First, a portion of the existing pheromone value is evaporated in all paths, according to:

$$\tau_{(i,j)} \leftarrow (1-\rho)\tau_{(i,j)} \tag{4.3}$$

where ρ is the evaporation coefficient $(0 \leq \rho < 1)$. Then, each ant n releases an amount of pheromone in the paths used to build the task sequence, according to the corresponding balancing solution quality. This amount of pheromone is given by:

$$\Delta\tau_{(i,j)}^{n} = \begin{cases} {}^{1}/_{Z}, if\ in\ the\ solution\ built\ by\ ant\ n\ task\ j\ is\ performed \\ \qquad immediately\ after\ task\ i \\ 0, \quad otherwise \end{cases} \tag{4.4}$$

where Z is the objective function value for the obtained solution. The overall pheromone update effect of all ants in each path (i, j) is then:

$$\tau_{(i,j)} \leftarrow (1-\rho)\tau_{(i,j)} + \sum_{n=1}^{N} \Delta\tau_{(i,j)}^{n} \tag{4.5}$$

At the beginning of the procedure, an initial amount of pheromone (τ_0) is released in every path.

4.3 Computational Experience

The concept of sequence dependent setup times is an actual framework in assembly line balancing problems (ALBP). For that reason, there is no standard set of benchmark instances with setup times available for testing in assembly line balancing literature except the Andrés et al.'s (2008) benchmark set, which covers simple version of ALBP, however, we deal with mixed-model version of ALBP in this study. For doing the comparison we construct a set of test problems based on type-I MMALB problems used by Akpınar & Bayhan (2011). The main characteristics of the test problems are exhibited in Table 4.1 where N, M, and C denote the number of tasks of the combined precedence diagram, the number of models, and cycle time of the assembly line, respectively. As in the study of Akpınar & Bayhan (2011), we also classified the test problems as small-size (problems 1-4), medium-size (problems 5-14), and large-size (problems 15-20) according to the number of tasks they include.

Table 4.1 Main characteristics of the test problems

	Problem No	N	M	C	Problem Name		Problem No	N	M	C	Problem Name
Small-Size	1	8	2	10	Bowman	Medium-Size	11	30	2	10	Sawyer
	2	8	3	10			12	30	3	10	
	3	11	2	10	Gokcen & Erel		13	32	2	10	Lutz 1
	4	11	3	10			14	32	3	10	
Medium-Size	5	21	2	10	Mitchel	Large-Size	15	35	2	10	Gunther
	6	21	3	10			16	35	3	10	
	7	25	2	10	Vilarinho & Simaria		17	45	2	10	Kilbridge & Wester
	8	25	3	10			18	45	3	10	
	9	28	2	10	Heskiaoff		19	70	2	10	Tonge
	10	28	3	10			20	70	3	10	

N: Number of tasks; M: number of models; C: Cycle time

For each test problem, the original precedence network and operation times remained the same and the following levels of setup time variability were also considered:

- For low variability, the matrix of setup times was generated randomly according to a uniform discrete distribution $U[0, 0.25*(\min T_i)]$.
- For medium variability, the matrix of setup times was generated randomly according to a uniform discrete distribution $U[0, 0.5*(\min T_i)]$.
- For high variability, the matrix of setup times was generated randomly according to a uniform discrete distribution $U[0, 0.75*(\min T_i)]$.

As a result the constructed benchmark set contains 60 instances in three categories, low, medium and high variability of setup times, each contains 20 problems having the same number of tasks, precedence diagrams and task times.

In order to evaluate the performance of the proposed hybrid ACO-GA algorithm it is required a lower for the number of workstations with sequence dependent setup times between tasks. The fallowing sub-section explains the proposed lower bound, which is a combination of Vilarinho & Simaria's (2002) and Andrés et al.'s (2008) (see the corrigendum to this paper provided by Pastor et al.(2010)) lower bounds.

4.3.1 A Lower Bound for the Number of Workstations with Setup Times

The problem on hand has the characteristics of workstation parallelization and sequence dependent setup times between tasks. Vilarinho & Simaria (2002) proposed a lower bound in case of workstation parallelization for mixed-model lines and Andrés et al.'s (2008) proposed another lower bound for single-model lines in case of sequence dependent setup times between tasks. Both procedures aimed at finding a lower bound value for the number of workstations. Due to the mixed-model nature of the problem on hand, we modified Andrés et al.'s (2008) procedure and combined with Vilarinho & Simaria's (2002) procedure in order to handle the problem characteristics on hand.

The proposed procedure for the lower bound, LB_{pmix}, was derived by using the following set of assumptions.

- The maximum number of replicas of per workstation is set as 2.
- A workstation can be duplicated only if the task time of one of the tasks assigned to it exceeds the cycle time for at least one of the models.
- The task time of the longest task does not exceed twice the cycle time (C).
- Setup times between the tasks vary from one model to another.

The steps required to compute LB_{pmix} are described as follows.

Step-1 For each model, classify the tasks according to the corresponding task time, as shown in Table 4.2 and go to Step 2.

Step-2 For each model, compute $LB'(m)$ and go to Step 3.

$$LB'(m) = \left\lceil \left(\begin{array}{c} 2(n_A + n_B + n_C) + y(n_D - n_C) + \frac{1}{2}w(n_E - n_B) + \\ \frac{5}{3}n_F + \frac{4}{3}n_G + \frac{2}{3}n_H + \frac{1}{3}n_I \end{array} \right) \right\rceil \tag{4.6}$$

where y equals 1 if $n_D - n_C > 0$ or zero otherwise and w equals 1 if $n_E - n_B > 0$ or zero otherwise.

Table 4.2 Classification of the tasks to compute *LBpmix*

Task Type	Task Time	Task Type	Task Time
A	$\frac{5}{3}C < T_A \leq 2C$	F	$T_F = \frac{5}{3}C$
B	$\frac{4}{3}C < T_B \leq \frac{5}{3}C$	G	$T_G = \frac{4}{3}C$
C	$C < T_C \leq \frac{4}{3}C$	H	$T_H = \frac{2}{3}C$
D	$\frac{2}{3}C < T_D \leq C$	I	$T_I = \frac{1}{3}C$
E	$\frac{1}{3}C < T_E \leq \frac{2}{3}C$	J	$T_J < \frac{1}{3}C$

Step-3 For each model, compute $Z(m)$ and go to Step 4.

$$Z(m) = \left[\left[\sum_{i=J} T_i - \left(LB'(m)C - \sum_{i \neq J} T_i\right)\right]/C\right] \tag{4.7}$$

Step-4 For each model, compute $LB^0_{pmix}(m) = LB' + Z(m)$ and go to Step 5.

Step-5 **/*** *M* is the number of different models, $SUT(m)$ is the sum of the *k* lowest setup times between the *N* tasks due to model *m* ***/**

for (*m*=1 to *M***) do**

Set $kontrol(m) = 0$ and,

$k(m) = 1$,

while ($kontrol(m) == 0$**) do**

Set $LB^k_{pmix}(m) = LB^0_{pmix}(m) + \frac{SUT(m)}{C}$ and,

$q_m = N - LB^k_{pmix}(m) + 1$,

if ($k(m) \geq q(m)$**)**

Set $kontrol(m) = 1$ and,

$LB_{pmix}(m) = (ceil)LB^k_{pmix}(m)$,

else

$k(m) = k(m) + 1$

endif

endwhile

endfor. Then go to Step 6.

Step-6 Select LB_{pmix} for the problem. $LB_{pmix} = max_m\left[LB_{pmix}(m)\right]$.

Table 4.3 indicates the lower bound values for the number of workstations determined by the proposed procedure for all the test problems with low, medium and high variability of setup times between tasks. On the other hand this lower bound computation is also important for the determination of initial pheromone level τ_0. For small, medium and large sized problem classes the initial pheromone levels are calculated by using the related lower bound values.

Table 4.3 Lower bound values for the test problems

Prob. No	Lower Bound (LB_{pmix})			Prob. No	Lower Bound (LB_{pmix})		
	Low Variability	Medium Variability	High Variability		Low Variability	Medium Variability	High Variability
1	5	5	5	11	14	14	14
2	6	6	6	12	16	16	16
3	8	8	8	13	17	17	17
4	7	7	7	14	18	18	18
5	15	15	15	15	20	20	20
6	14	14	14	16	21	21	21
7	15	15	15	17	23	23	23
8	14	14	14	18	24	24	24
9	19	19	19	19	39	39	39
10	18	18	18	20	40	40	40

4.3.2 Computational Results

All the algorithms GA, hGA (Akpınar & Bayhan, 2011), ACO and hybrid ACO-GA were coded in C++, and the solutions of the test problems were obtained by running the algorithms on Intel (R) Core 2 Duo CPU T7300 (2.0 GHz). The parameter sets used for GA, hGA and ACO are given in Table 4.4 and the parameter set used for proposed hybrid ACO-GA algorithm presented in Table 4.5. These parameters were chosen experimentally for getting a satisfactory performance in an acceptable time span.

Table 4.4 Parameter sets for ACO, GA and hGA

Problem	Ant Colony Optimization									Genetic Algorithm, hGA			
	τ_0	ρ	α	β	r_1	r_2	r_3	N_{SC}	N_{AA}	P_S	R_C	R_M	N_I
1-4(Small)	7	0.15	0.25	1.25	0.6	0.3	0.1	200	20	20	0.5	0.15	200
5-14(Medium)	16	0.15	0.25	1.25	0.6	0.3	0.1	200	50	50	0.5	0.15	200
15-20(Large)	28	0.15	0.25	1.25	0.6	0.3	0.1	200	100	100	0.5	0.15	200

$\boldsymbol{\tau_0}$: Initial pheromone level; $\boldsymbol{\rho}$: evaporation coefficient; $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$: determine the relative importance of pheromone intensity versus heuristic information; $\boldsymbol{r_1}$, $\boldsymbol{r_2}$ and $\boldsymbol{r_3}$: user defined parameters for task selection strategy; $\boldsymbol{N_{SC}}$: Number of sub-colonies; $\boldsymbol{N_{AA}}$: Number of artificial ants in each sub-colony; $\boldsymbol{P_S}$: Size of the population; $\boldsymbol{R_C}$: Crossover rate; $\boldsymbol{R_M}$: Mutation Rate; $\boldsymbol{N_I}$: Number of iterations

According to Dorigo & Gambardella (1997), a rough approximation of the optimal value of the objective function is a reasonable value for τ_0. In the current study, since the average lower bound values for cycle times are approximately equal to 7, 16 and 28 for small, medium and large sized problem classes, we set the initial pheromone levels (τ_0) to 7, 16 and 28 for small, medium and large sized problems, respectively. The other parameters, except the number of ants (N_{AA}) in each sub

colony, have the same values for all test problems. N_{AA} indicates the number of different solutions which must be evaluated at each iteration by ACO, thus it have to be well proportioned to the solution space size in order to satisfy algorithm's diversification. The higher values of N_{AA} may the algorithm cause redundant computational effort while the lower values may the algorithm cause insufficient diversification. Hence, N_{AA} should have different values for the different sized problems. Since N_{AA} is a parameter related with the search space size of a problem, we set N_{AA} as 10, 25 and 50 for small sized, medium sized and large sized problems, respectively.

Table 4.5 Parameter set for hybrid ACO-GA

Problem	Ant Colony Optimization									Genetic Algorithm, hGA			
	τ_0	ρ	α	β	r_1	r_2	r_3	N_{SC}	N_{AA}	P_S	R_C	R_M	N_I
1-4(Small)	7	0.15	0.25	1.25	0.6	0.3	0.1	100	20	20	0.5	0.15	50
5-14(Medium)	16	0.15	0.25	1.25	0.6	0.3	0.1	100	50	50	0.5	0.15	50
15-20(Large)	28	0.15	0.25	1.25	0.6	0.3	0.1	100	100	100	0.5	0.15	50

The test problems were solved using GA, hGA, ACO and the proposed hybrid ACO-GA and the minimum, maximum and average values of the solutions, for each of the test problems shown in Tables 4.5 and 4.6, results from ten runs of each instance of the problem. We conduct a comparison between GA, hGA, ACO and ACO-GA in terms of determined number of workstations (N_{WS}) and computational time (CPU). As it can be seen from Tables 4.5 and 4.6, ACO outperforms both hGA and GA for medium and large sized problems in case of low, medium and high variability of setups, while hGA outperforms pure GA. On the other hand ACO-GA outperforms ACO, GA and hGA for medium and large sized problems. It must be noted that as the problem size get larger the performance of ACO increase significantly in comparison with GA and hGA. From the observations of Tables 4.5 and 4.6, it is found that the performance of hGA is superior to GA in 18.33% (11 of 60 problems), the performance of ACO superior to GA in 41.67% (25 of 60 problems) and hGA in 36.67% (22 of 60 problems), and the performance of ACO-GA superior to ACO in 21.67% (13 of 60 problems), GA in 46.67% (28 of 60 problems) and hGA in 41.67% (25 of 60 problems) of the test problems. However, it is clear that ACO has slower speed in comparison of GA, hGA and ACO-GA.

Table 4.5 Computational results of GA versus ACO

	Problem No	Low Variability of Setup Times								Medium Variability of Setup Times								High Variability of Setup Times							
		GA				ACO				GA				ACO				GA				ACO			
		N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU
		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max	
Small Size	1	5	5	5	0.12	5	5	5	0.19	5	5	5	0.12	5	5	5	0.16	5	5	5	0.13	5	5	5	0.15
	2	9	9	9	0.14	9	9	9	0.21	10	10	10	0.14	10	10	10	0.21	10	10	10	0.14	10	10	10	0.20
	3	8	8	8	0.14	8	8	8	0.22	8	8	8	0.15	8	8	8	0.25	8	8	8	0.16	8	8	8	0.23
	4	7	7	7	0.14	7	7	7	0.23	7	7	7	0.14	7	7	7	0.26	7	7	7	0.15	7	7	7	0.24
Medium Size	5	17	17	17	0.44	17	17	17	1.00	17	17	17	0.44	17	17	17	0.96	19.7	19	20	0.58	19	19	19	1.07
	6	16	16	16	0.44	16	16	16	0.90	17.8	17	18	0.24	17	17	17	0.98	18.2	18	19	0.52	18	18	18	0.99
	7	17	17	17	0.43	17	17	17	1.41	18	18	18	0.52	18	18	18	1.28	19	19	19	0.55	18.8	**18**	19	1.35
	8	16	16	16	0.56	16	16	16	1.31	16	16	16	0.55	16	16	16	1.29	19	19	19	0.55	19	19	19	1.37
	9	22.9	22	23	0.70	22	22	22	2.95	23.9	23	24	0.70	23	23	23	3.00	24.1	24	25	0.71	24	24	24	3.03
	10	21.2	21	22	0.80	20.5	**20**	21	2.96	22.4	22	23	0.83	21	**21**	21	3.07	23	23	23	0.81	21	**21**	21	3.27
	11	18	18	18	0.56	17	**17**	17	1.40	19.5	19	20	0.60	18	**18**	18	1.48	20.5	20	21	0.62	19	**19**	19	1.49
	12	21.5	21	22	0.60	21	21	21	1.42	23	23	23	0.65	22.7	**22**	23	1.42	24.3	24	25	0.65	24	24	24	1.43
	13	21	21	21	0.62	21	21	21	1.75	22	22	22	0.62	22	22	22	1.73	23	23	23	0.64	23	23	23	1.74
	14	21.4	21	22	0.65	21	21	21	1.75	23	23	23	0.66	23	23	23	1.83	26	26	26	0.77	26	26	26	1.88
Large Size	15	26.3	26	27	1.44	25.1	**25**	26	4.35	28.2	28	29	1.44	27.5	**27**	28	4.55	29.1	29	30	1.52	28	**28**	28	4.83
	16	26	26	26	1.40	26	26	26	4.59	28	28	28	1.44	27.6	**27**	28	4.73	29	29	29	1.65	28	**28**	28	4.82
	17	28.2	28	29	1.90	26	**26**	26	9.10	30.1	29	31	2.01	28	**28**	28	9.13	31.5	31	32	2.30	30	**30**	30	9.64
	18	31.2	31	32	2.16	29	**29**	29	9.42	32.9	32	33	2.17	31	**31**	31	9.98	34.2	34	35	2.22	33	**33**	33	10.12
	19	52.2	51	53	3.89	47	**47**	47	21.25	56	55	57	3.76	51.1	**51**	52	22.50	57.3	57	58	3.88	51	**51**	51	24.55
	20	52.8	52	53	3.89	48	**48**	48	21.67	56.4	56	57	3.93	52	**52**	52	22.17	58.8	58	59	4.00	56	**56**	56	23.48

N_{WS}: Number of workstations; **CPU**: Computational time

Table 4.6 Computational results of hGA versus hybrid ACO-GA

	Problem No	Low Variability of Setup Times								Medium Variability of Setup Times								High Variability of Setup Times							
		hGA				ACO-GA				hGA				ACO-GA				hGA				ACO-GA			
		N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU
		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max	
Small Size	1	5	5	5	0.32	5	5	5	0.05	5	5	5	0.33	5	5	5	0.06	5	5	5	0.33	5	5	5	0.04
	2	9	9	9	0.34	9	9	9	0.07	10	10	10	0.35	10	10	10	0.07	10	10	10	0.34	10	10	10	0.06
	3	8	8	8	0.35	8	8	8	0.08	8	8	8	0.37	8	8	8	0.08	8	8	8	0.36	8	8	8	0.08
	4	7	7	7	0.36	7	7	7	0.09	7	7	7	0.34	7	7	7	0.09	7	7	7	0.35	7	7	7	0.09
Medium Size	5	17	17	17	0.74	17	17	17	0.52	17	17	17	0.72	17	17	17	0.54	19.4	19	20	0.78	19	19	19	0.57
	6	16	16	16	0.76	16	16	16	0.54	17.2	17	18	0.74	17	17	17	0.54	18	18	18	0.72	18	18	18	0.55
	7	17	17	17	0.73	17	17	17	0.68	18	18	18	0.82	18	18	18	0.73	18.7	18	19	0.85	18.4	18	19	0.76
	8	16	16	16	0.78	16	16	16	0.75	16	16	16	0.85	16	16	16	0.81	19	19	19	0.85	18.6	**18**	19	0.81
	9	22.4	22	23	0.92	22	22	22	1.66	23.5	23	24	1.03	23	23	23	1.70	24	24	24	1.01	23.8	**23**	24	1.77
	10	21	21	21	1.02	20.1	**20**	21	1.69	22.2	22	23	1.13	21	**21**	21	1.85	22	22	22	1.14	21	**21**	21	1.79
	11	18	18	18	0.76	17	**17**	17	0.78	19	19	19	0.92	18	**18**	18	0.80	20	20	20	0.93	19	**19**	19	0.78
	12	21.2	21	22	0.80	21	21	21	0.83	23	23	23	0.95	22.1	**22**	23	0.85	24.2	24	25	0.97	24	24	24	0.86
	13	21	21	21	0.92	21	21	21	0.90	22	22	22	0.97	22	22	22	1.06	23	23	23	1.00	23	23	23	0.98
	14	21.3	21	22	0.95	21	21	21	1.00	23	23	23	0.96	23	23	23	1.10	26	26	26	1.03	26	26	26	1.07
Large Size	15	26	26	26	1.74	25	**25**	25	2.73	28	28	28	1.75	27	**27**	27	2.95	29	29	29	1.82	28	**28**	28	2.92
	16	26	26	26	1.80	25.7	**25**	26	2.94	28	28	28	1.81	27	**27**	27	3.02	29	29	29	1.88	28	**28**	28	3.02
	17	28	28	28	2.15	26	**26**	26	5.13	29.8	29	30	2.21	28	**28**	28	5.45	30.5	30	31	2.50	29.6	**29**	30	5.83
	18	31	30	32	2.27	28.6	**28**	29	5.65	32.3	32	33	2.27	30.3	**30**	31	5.64	33.2	33	34	2.52	32.3	**32**	33	5.77
	19	52	50	53	4.89	46.7	**46**	47	12.17	55	54	56	4.73	50.8	**50**	51	12.65	55.3	55	56	4.83	50.7	**50**	51	12.98
	20	52	50	53	4.91	47.7	**47**	48	12.65	55	54	56	4.92	51.5	**51**	52	13.22	57.8	57	58	4.94	55	**55**	55	13.76

N_{WS}: Number of workstations; **CPU**: Computational time

As a summary of Tables 4.5 and 4.6, we formed Table 4.7 in order to make the comparison for ACO, GA, hGA and hybrid ACO-GA more clearly in terms of gaps for the obtained minimum number of workstations. The criterion of gap is meaningful, if we consider the high cost to build and maintain an assembly line. Less workstation means less equipment to purchase, payoff and maintain. This cost affect also indicates the superior performance of ACO-GA to ACO GA and hGA.

Table 4.7 Comparison of GA, ACO, hGA and hybrid ACO-GA

GAPs		Problem No	Small Size				Medium Size										Large Size					
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	Low Variability	ACO GA	-	-	-	-	-	-	-	-	-	1	1	-	-	-	1	-	2	2	4	4
		ACO-GA GA	-	-	-	-	-	-	-	-	-	1	1	-	-	-	1	1	2	3	5	5
		ACO-GA ACO	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	1	1
		GA hGA	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	2
		ACO hGA	-	-	-	-	-	-	-	-	-	1	1	-	-	-	1	-	2	1	3	2
		ACO-GA hGA	-	-	-	-	-	-	-	-	-	1	1	-	-	-	1	1	2	2	4	3
	Medium Variability	ACO GA	-	-	-	-	-	-	-	-	-	1	1	1	-	-	1	1	1	1	4	4
		ACO-GA GA	-	-	-	-	-	-	-	-	-	1	1	1	-	-	1	1	1	2	5	5
		ACO-GA ACO	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1
		GA hGA	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	2
		ACO hGA	-	-	-	-	-	-	-	-	-	1	1	1	-	-	1	1	1	1	3	2
		ACO-GA hGA	-	-	-	-	-	-	-	-	-	1	1	1	-	-	1	1	1	2	4	3
	High Variability	ACO GA	-	-	-	-	-	-	1	-	-	2	1	-	-	-	1	1	1	1	6	2
		ACO-GA GA	-	-	-	-	-	-	1	1	1	2	1	-	-	-	1	1	2	2	7	3
		ACO-GA ACO	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	1	1	1	1
		GA hGA	-	-	-	-	-	-	1	-	-	1	-	-	-	-	-	-	1	1	2	1
		ACO hGA	-	-	-	-	-	-	-	-	-	1	1	-	-	-	1	1	-	-	4	1
		ACO-GA hGA	-	-	-	-	-	-	-	1	1	1	1	-	-	-	1	1	1	1	5	2

From the above experimental results, we can claim that ACO and ACO-GA has significantly good performance in comparison to GA and hGA. Hence, we analyzed

the results obtained from ACO and ACO-GA in more detailed in Table 4.8, which exhibits the results of the proposed hybrid ACO-GA and ACO for the 20 test problems with low, medium and high variability of setup times in terms of minimum number of workstations (N_{WS}), the weighted line efficiency (WE), percentage difference ($D\%$) between the obtained number of workstations obtained by the proposed hybrid algorithm and the LB values (LB_{pmix}). Percentage difference between the minimum number of workstations and the LB value for a problem is computed by Equation 4.8.

$$D_i(\%) = \frac{N_{ws_i} - LB_{pmix_i}}{LB_{pmix_i}} \times 100 \qquad i \in \{1,2,\dots,20\} \tag{4.8}$$

As pointed out in 6th, 9th, 12th, 15th, 18th, and 21th columns of Table 4.8,where the values of $D\%$ equal to zero for the problems 1, 3 and 4, thus, an optimal solution is found for these three problems with low, medium and high variability of setup times by both ACO and ACO. On the other hand, the worst performance is for problem 2, where the difference between the solutions obtained and the lower bound is 50%, 67% and 67% for low, medium and high variability of setup times, respectively.

The formula of the weighted line efficiency is given by Equation 4.9, which takes into account not only the task times but only the sequence dependent setup times between tasks. Hence, the weighted line efficiency is computed by considering idle times of the workstations instead of considering the tasks times. It is noted that maximizing line efficiency is equivalent to minimizing the number of workstations (N_{WS}) for a given cycle time (C).

$$WE = \sum_{m=1}^{M} \left[q_m \frac{N_{ws}C - \sum_{s=1}^{S} Idle_{sm}}{N_{ws}C} \right] \tag{4.9}$$

where q_m is the overall proportion of the number of units of model m being assembled, $Idle_{sm}$ is the idle time of workstation S due to model m, N_{WS} is the minimum value for the number of workstations obtained by the solution procedure.

Table 4.8 Performance evaluation of ACO and the hybrid ACO-GA

	Problem No	Low Variability of Setup Times							Medium Variability of Setup Times							High Variability of Setup Times						
		LB_{pmix}	ACO			ACO-GA			LB_{pmix}	ACO			ACO-GA			LB_{pmix}	ACO			ACO-GA		
			N_{WS}	WE	D %	N_{WS}	WE	D %		N_{WS}	WE	D %	N_{WS}	WE	D %		N_{WS}	WE	D %	N_{WS}	WE	D %
Small Size	1	5	5	71.9	0	5	71.9	**0**	5	5	75.9	0	5	75.9	**0**	5	5	77.8	0	5	77.8	**0**
	2	6	9	49.6	50	9	49.6	50	6	10	43.9	67	10	43.9	67	6	10	43.9	67	10	43.9	67
	3	8	8	64.2	0	8	64.2	**0**	8	8	65.2	0	8	65.2	**0**	8	8	66.0	0	8	66.0	**0**
	4	7	7	78.6	0	7	78.6	**0**	7	7	80.2	0	7	80.2	**0**	7	7	83.4	0	7	83.4	**0**
Medium Size	5	15	17	69.4	13	17	69.4	13	15	17	70.3	13	17	70.3	13	15	19	63.6	27	19	63.6	27
	6	14	16	77.0	14	16	77.0	14	14	17	73.9	21	17	73.9	21	14	18	70.3	29	18	70.3	29
	7	15	17	73.9	13	17	73.9	13	15	18	70.9	20	18	70.9	20	15	18	71.1	20	18	71.1	20
	8	14	16	81.5	14	16	81.5	14	14	16	83.8	14	16	83.8	14	14	19	69.9	36	18	74.4	29
	9	19	22	85.1	16	22	85.1	16	19	23	82.7	21	23	82.7	21	19	24	80.7	26	23	85.3	21
	10	18	20	85.1	11	20	85.1	11	18	21	83.5	17	21	83.5	17	18	21	84.7	17	21	84.7	17
	11	14	17	86.2	21	17	86.2	21	14	18	84.1	29	18	84.1	29	14	19	80.1	36	19	80.1	36
	12	16	21	81.0	31	21	81.0	31	16	22	77.2	38	22	77.2	38	16	24	71.4	50	24	71.4	50
	13	17	21	73.2	24	21	73.2	24	17	22	70.8	29	22	70.8	29	17	23	68.8	35	23	68.8	35
	14	18	21	76.8	17	21	76.8	17	18	23	71.1	28	23	71.1	28	18	26	62.7	44	26	62.7	44
Large Size	15	20	25	79.6	25	25	79.6	25	20	27	75.1	35	27	75.1	35	20	28	73.9	40	28	73.9	40
	16	21	26	81.3	24	25	84.9	19	21	27	81.2	29	27	81.2	29	21	28	78.8	33	28	78.8	33
	17	23	26	86.3	13	26	86.3	13	23	28	82.8	22	28	82.8	22	23	30	76.7	30	29	79.6	26
	18	24	29	82.1	21	28	85.1	17	24	31	79.1	29	30	82.4	25	24	33	74.3	38	32	77.4	33
	19	39	47	84.2	21	46	86.2	18	39	51	78.7	31	50	80.6	28	39	51	79.3	31	50	81.4	28
	20	40	48	82.0	20	47	83.9	18	40	52	77.6	30	51	79.2	28	40	56	70.2	40	55	73.9	38

The precedence and zoning constraints are not taken into account by the calculation procedure of the lower bound. Therefore, the obtained results from the proposed hybrid ACO-GA algorithm are fairly satisfactory. This conclusion is reinforced by the values for the weighted line efficiency shown in 8th, 15th and 22th columns of Table 4.8.

In order to understand whether the differences in the obtained results are due to the random chance or not, a paired *t*-test is executed on Minitab 15. After paired *t*-test it is figured out that outputs (average values for 10 independent replications represented in Tables 4.5 and 4.6) derived from GA, hGA, ACO and ACO-GA are meaningfully different from each other. The results of paired *t*-test are presented in Table 4.9. Based on the values from Table 4.9, it can be concluded that hybrid ACO-GA produced better results than GA, hGA and ACO, and ACO produced better results than both GA and hGA, and hGA produced better results than GA in general.

Table 4.9 p values obtained from the comparison of algorithms using paired t-test (α=0.05)

	p Value		
	Low Variability	Medium Variability	High Variability
ACO - GA	0.013	0.007	0.011
hGA - GA	0.007	0.007	0.005
ACO - hGA	0.018	0.009	0.023
ACO-GA - GA	0.010	0.004	0.006
ACO-GA - ACO	0.016	0.011	0.015
ACO-GA - hGA	0.014	0.004	0.009

Considering the average computational (CPU) times, GA algorithm is faster than hGA, ACO and hybrid ACO-GA, while the proposed hybrid ACO-GA is faster than ACO. This situation may be explained by the fact that solution generation effort of

the ACO in each iteration is much higher that GA, hGA and hybrid ACO-GA. On the other hand, it must be noted that the computational time is also related to the used parameter set.

This set of computational experiments shows that the overall performance of ACO-GA is superior to ACO, GA and hGA, the overall performance of ACO is superior to GA and hGA, and the overall performance of hGA is superior to GA.

4.4 Chapter Conclusions

In this chapter, we aimed at hybridizing ACO with GA in order to improve search ability of ACO for solving mixed-model assembly line balancing problem with sequence dependent setup times between tasks. In the proposed hybrid ACO-GA algorithm, GA embedded into ACO. In order to evaluate the real performance of the proposed hybrid ACO-GA algorithm, a set of 20 mixed-model assembly line balancing problems with parallel workstations and zoning constraints was tested. As to the scope of this study, this set of benchmark problems was differentiated by adding sequence dependent setup times between tasks with low, medium and high variability. Due to the lack of optimal solutions for the benchmark set with setups, a procedure for determining lower bounds for the problem on hand was derived, for evaluating the proposed hybrid ACO-GA algorithm in terms of number of workstations. The performance of the proposed hybrid ACO-GA algorithm was also compared with the performances of pure ACO, pure GA and hGA. Computational results indicate that ACO-GA can improve search performance and outperforms ACO, GA and hGA.

CHAPTER FIVE
A MULTIPLE COLONY HYBRID BEES ALGORITHM FOR MMALBPS-I

5.1 Chapter Introduction

The MMALBP-I is NP-hard (Bukchin & Rabinowitch, 2006), complex, and CPU time-consuming (Battaïa & Dolgui, 2012a). Thus, exhaustive search methods can not to solve MMALBP-I within polynomially bounded computation times. The reader can refer to Battaïa & Dolgui (2012b) for a recent survey on solution approaches of assembly line balancing problems.

Design issues of the meta-heuristic approaches are generally depending on nature just because offering much broader wealth of inspiration. Social insects seem to be more interesting than the other sources of inspiration in nature, since their communication systems provides developing efficient solution procedures for combinatorial optimization (Özbakır et al., 2010). In fact, their behavior is attractive not only individually but also in a population from the optimization point of view, such that, some meta-heuristics approaches based on the simulation of this group behavior. This class of meta-heuristic approaches is named as population-based or swarm-based optimization algorithms and includes Ant Colony Optimization (ACO) (Dorigo et al., 1991), Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995), Bee Colony Optimization (BCO) (Karaboga, 2005) and Bees Algorithm (BA) (Pham et al., 2006). Additionally, Genetic Algorithm (GA) (Holland, 1975) must be mentioned when population-based optimization algorithms are discussed, however, GA simulates the genetic evolutional process not the group behaviors of social insects. In this study, we are interested in BA for solving type I mixed-model assembly line balancing problem with setups (MMALBPS-I) because of the multi population structure of the honey bees; each population represents the honey bees living in a different hive.

Baykasoglu et al. (2007) surveyed the application areas and algorithms on behavioral characteristics of the honey bees and the authors presented a

classification, while, Ozbakır & Tapkan (2011) gave a brief literature review about the “foraging behavior” based optimization algorithms. Moreover, another survey of the algorithms based on the intelligence in bee swarms and applications has been presented by Karaboga & Akay (2009). For more detailed information about the applications of bee swarm intelligence, the readers can refer to one of these three papers, especially to (Baykasoglu et al., 2007) or (Karaboga & Akay, 2009). In this study, we only deal with the application of bee swarm intelligence to the assembly line balancing problems. The literature about the applications of bee swarm intelligence to assembly lines is scarce. To the best of our knowledge, there are only two papers which dealt with ALBPs. Ozbakır & Tapkan (2011) and Tapkan & Ozbakır (2012) used BA for solving Two-sided Assembly Line Balancing Problem (TSALBP). Both of the papers were adopted BA to TSALBP by generating random solutions according to different heuristic rules by using shift and swap movements as neighborhood generators.

The basic version of the BA is a combination of random search and neighborhood search, which may have different structures according to the features of the problem on hand. Most of the existing literature about the applications of BA to combinatorial optimization tries to evolve only a single population, with an exception of Akbari & Ziarati’s (2011) work as they developed a cooperative bee swarm optimization algorithm for functional optimization. Furthermore, the implementations of BA for the assembly line balancing problems (ALBPs) could be classified into improvement type of search algorithms because of the employed neighborhood structures (shift and swap movements), however, the constructive type of search algorithms like ACO are much more effective if the problem has an inherently network structure (Baykasoglu et al., 2006) as ALBPs. The existing literature (McMullen & Tarasewich, 2003; Simaria & Vilarinho, 2009; Vilarinho & Simaria, 2006; Yagmahan, 2011) addressing the solution of mixed-model ALBPs using ACO also introduced the encouraging performance of ACO.

In the present study, we have proposed a multiple colony hybrid Bees algorithm (MCHBA) for MMALBPS-I. Our proposed approach is based on the multiple

colonies (similar to Özbakır et al.(2011)); each colony is formed according to a different heuristic information, with the purpose of improving the diversification of the algorithm. Diversification generally refers to the ability to visit many and different regions of the search space (Lozano & García-Martínez, 2010). Moreover, we used a new neighborhood structure which ensures the algorithm to be a constructive type. This neighborhood structure also enables the proposed approach to utilize the positive feedback mechanism as ACO does. Due to the multiple colonies, the proposed algorithm needs a communication strategy to be realized by the new neighborhood structure for sharing the information. Information sharing is an essential issue from the optimization point of view. However, we should mention here that there is not much information about the information sharing mechanisms between different colonies in real honey bees. We have adopted a mechanism which is similar to Ozbakir *et al*.'s (2011) study.

The remainder of this chapter is organized as follows. The proposed Multiple Colony Hybrid Bees Algorithm is defined in Section 5.2. Comparative computational study is given in Section 5.3. Finally, the conclusions are presented in Section 5.4.

5.2 Multiple Colony Hybrid Bees Algorithm

In this section the proposed MCHBA is presented in detail. The pseudo code of MCHBA is given in Figure 5.1 and the notations are given in Table 5.1.

Table 5.1 Notations for the pseudo code and their corresponding definitions

Notation	Definition	Notation	Definition
M	Number of colonies (m=1,…,M)	*MaxIter*	Iteration number (Stopping criteria)
S	Number of scout bees (s=1,…,S)	σ^{best}	Best solution
P	Number of employed bees (p=1,…,P)	${}_{mp}\sigma^{best}$	Best solution of p^{th} population of the m^{th} colony
e	Number of best employed bees	${}_{m}\sigma^{best}$	Best solution of the the m^{th} colony
nep	Number of onlooker bees for each e employed bees	${}_{m}\sigma^{s}$	Solution of the s^{th} scout bee of the m^{th} colony
nsp	Number of onlooker bees for each P–e employed bees ($nsp < nep$)	$f({}_{m}\sigma^{s})$	Fitness function value of the s^{th} scout bee of the m^{th} colony

```
1-  Determine M different heuristic rules for M different colonies
2-  Parameter initialization and forming global pheromone matrix
3-  Initialize each colony (consist of S scout bees) by using its own heuristic rule
4-  for (m=1 to M) do
        for (s=1 to S) do
            Evaluate scout bee's fitness function (f(mσ^s))
        endfor
    endfor
5-  k=0
    while (k<MaxIter)
        for (m=1 to M) do
            Sort scout bees (mσ^s) according to their fitness (f(mσ^s)) in increasing order
            Determine best P solutions as employed bees and select best e employed bees from P
            Release a certain amount of pheromone for each employed bee
            Form each employed bee's initial pheromone matrix.
            Assign nep onlooker bees to each best employed bees in order to form e different
            populations each one has nep individuals
            Assign nsp onlooker bees to each remaining P-e employed bees in order to form P-e
            different populations each one has nsp individuals
                for (p=1 to P) do
                    Apply neighborhood structure to population p
                    Record p^th population's best bee (mpσ^best)
                    Update m^th colony's best bee (mσ^best)
                        if f(mpσ^best) ≤f(mσ^best)
                            mσ^best=mpσ^best
                        endif
                endfor
        endfor
            Update best solution
                for (m=1 to M) do
                    if f(mσ^best)≤f(σ^best)
                        σ^best= σ^p
                    endif
                endfor
            Release a certain amount of pheromone for best bee and update global pheromone
            matrix
        k=k+1
            if (k< MaxIter)
                for (m=1 to M) do
                    Initialize S-P scout bees with heuristic rule
                        for (s=1 to S-P) do
                            Evaluate scout bee's fitness function (f(mσ^s))
                        endfor
                endfor
            endif
    endwhile
```

Figure 5.1 Pseudo code of the proposed multiple colony hybrid bees algorithm

The proposed MCHBA starts by determining the heuristic rules and matching them to exactly one colony. After that, the algorithm continues by initializing the parameters and forming the initial global pheromone matrix, which is used for information sharing between colonies. The information sharing mechanism (Section 5.2.1) is an important issue in the way of the behaviors of the bees in their own colonies. Initial colonies are generated by their own heuristic rules which are explained in Section 5.2.2. For each colony, the following steps are executed until a

predefined number of iteration is reached. *P* number of fittest bees within the set of generated solutions is determined as employed bees. From this set of employed bees *e* numbers of bees are selected as the best bees. *nep* numbers of onlooker bees are assigned to these best bees and *nsp* number of onlooker bees are assigned to the remaining *P–e* bees in order to generate different populations. The neighborhood mechanism is applied to each population for improving the algorithm's intensification. The best onlooker bee of each population is compared with the original employed bee and, if it is better, the employed bee is substituted for the best onlooker bee. Moreover, each best onlooker bee is compared with the best bee, if it is better than the previous best bee, the best bee is updated. For global search, *S–P* number of scout bees is generated by using heuristic rules.

Real honey bees use a mechanism named as waggle dance for sharing the information. Waggle dance is performed by the employed forager bees in order to share with the other bees of the colony information about the direction and distance of the food sources. If an unemployed recruit bee decides to start searching, the bee attends to a waggle dance done by some other bee for getting information and uses this information throughout its search. These behavioral properties of the real honey bees are so similar to the pheromone laying and following behaviors of ants. This similarity constitutes the basis of our proposed multiple colony hybrid bees algorithm, whose steps are explained in details in the following sub-sections.

5.2.1 Behaviors of the Bees in their own Colonies

The proposed MCHBA is trying to discipline each bee to approach the best bee of its own colony and the best bee of all colonies. In order to attain such behavior we need to realize an information sharing mechanism between the colonies, however, as we have mentioned previously there is not much information about this issue in the literature on real honey bees. Furthermore, such an information sharing mechanism is a vital issue from the optimization point of view. The global pheromone matrix (Özbakır et al., 2011), explained in Section 5.2.4, provides us to share the information between colonies.

The behaviors of the bees in their own colonies that we envisaged before constructing the algorithm can be seen in Figure 5.2. The bees of the initial colonies are scattered due to the effect of randomness, as can be seen in Figure 5.2-a. After a predetermined number of iterations, we aim at making the algorithm to localize the bees near to their own colony's best bee. We also envisaged that the bees are localized by the algorithm at the closer side of their colony to the global best bee, as it can be seen in Figure 5.2-b. So, each bee wants to fallow not only its own colony's best bee, but also the global best bee. Additionally, we want to test if the bees have these two behavioral properties or not, however, we are not able to do this test on the assembly line balancing problem because of its complex structure. Such a test can be done on a function in continuous domain more easily. For that reason we select one of the well known non-convex functions named as Rastrigin (RF) and this test carried out on Matlab 7.9.0. The alignments of the bees in initial colonies (Fig. 5.3-a) and in final colonies (Fig. 5.3-b) verify the behaviors of the bees in Figure 5.2 on RF.

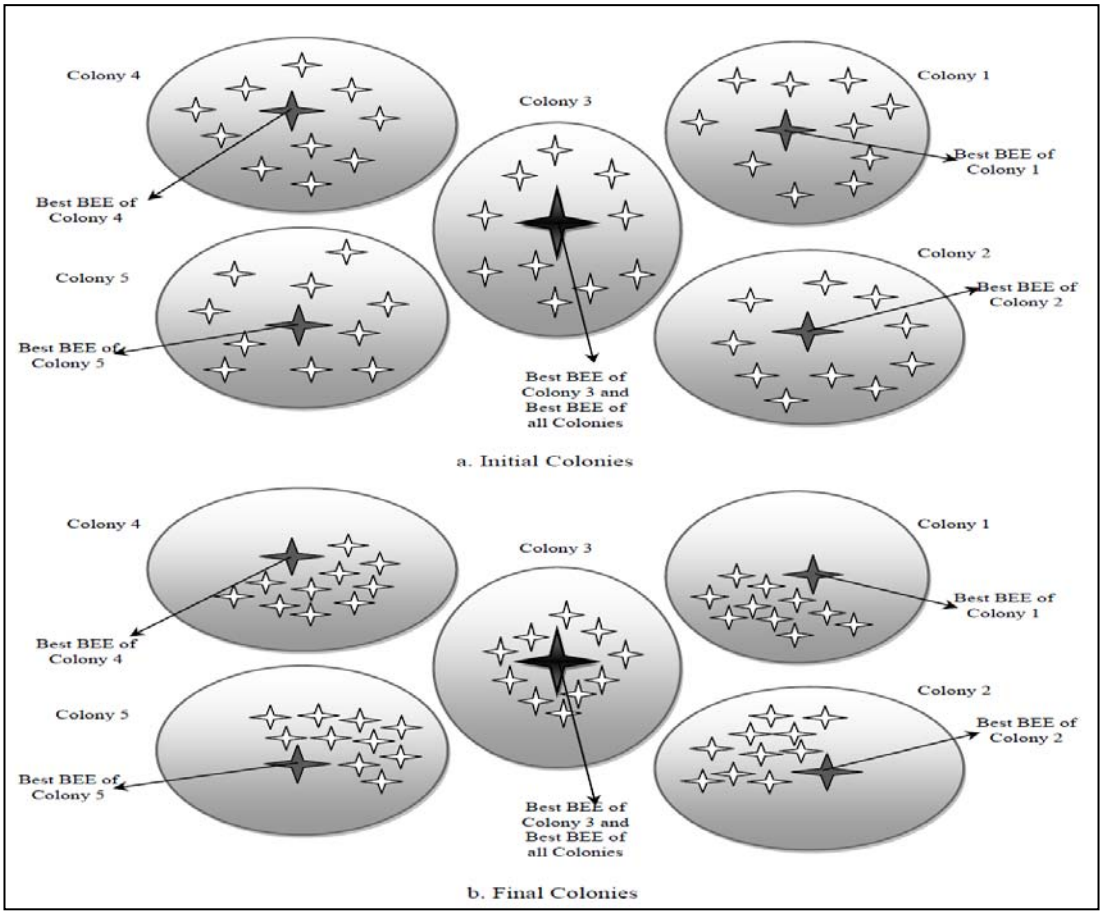
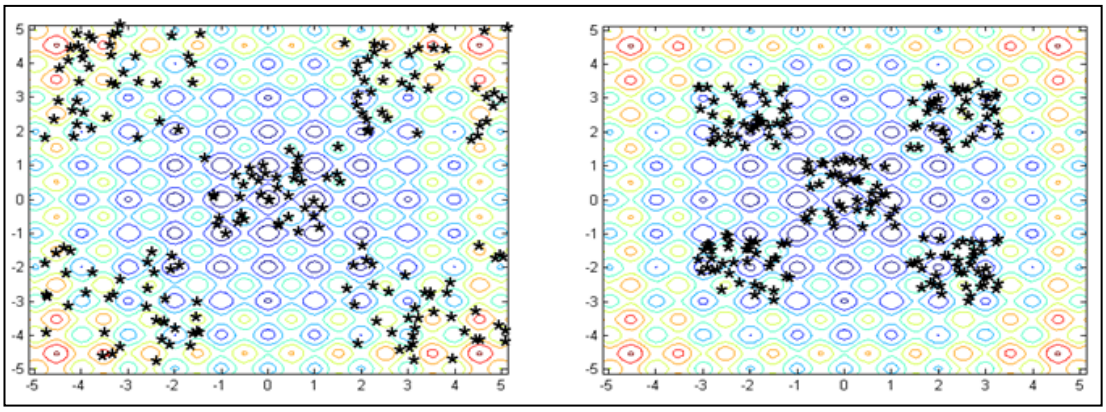


Figure 5.2 Envisaged behaviors of BEEs in their own colonies



a. Initial colonies b. Final colonies

Figure 5.3 Behaviors of BEEs in their own colonies on RF

Above mentioned two behavioral properties of the bees are important because of the effectiveness of the algorithm. Following colony's best bee provides the algorithm convergence in each colony while following the best bee provides the algorithm convergence between colonies. Hence, the algorithm achieves the ability of approaching to the global optimum rapidly.

5.2.2 Initial Colonies

In the proposed MCHBA two types of bees exist, one represents a coding of a solution when the other constructs step by step a feasible balancing solution. The first type of bees is used to generate initial colonies when the other used by the proposed neighborhood structure. For coding, we used task based representation (Leu et al., 1994; Sabuncuoglu et al., 2000), which is the most appropriate representation scheme for type-1 balancing problems of assembly lines. The length of the representation scheme is defined by the number of tasks and each value of this scheme represents a task. In this study, the phenomenon of the multiple colonies and randomly generated solutions are considered. That is to say, each colony used a different heuristic rule and each colony consists of a predefined number of solutions generated by selecting randomly a task at a time according to the heuristic rule. Heuristic rules based procedures developed and examined for different types of balancing problems by (Andrés et al., 2008; Martino & Pastor, 2010; Baykasoglu, 2006; Bautista & Pereira, 2002; Wilhelm, 1999).

The heuristic rules assigned to colonies are: *(i)-Maximum Processing Time of all Models (MPTM):* Selects the task having maximum processing time for all models. *(ii)-Maximum Average Processing Time (MAPT):* Selects the task having maximum average processing time. The average processing time of a task is the sum of the processing times of that task for each model weighted by the respective production share. *(iii)-Maximum Average Ranked Positional Weight (MARPW):* Selects the task having maximum average ranked positional weight. In a mixed-model assembly line, the positional weight of a task is the cumulative average task processing time associated with itself and its successors. *(iv)-Maximum Number of Direct Successors (MNDS):* Selects the task having maximum total number of direct followers according to combined precedence diagram. *(v)-Maximum Total Number of Successors (MTNS):* Selects the task having maximum total number of followers according to combined precedence diagram.

Work assignment within the proposed approach is made by the following procedure. Tasks are assigned to the workstations according to the task sequence in the representation scheme, as long as the predetermined cycle time is not exceeded. Once the cycle time is exceeded at least for a model or the zoning constraints are not satisfied, a new workstation is opened for assignment, and the procedure is repeated. Figure 5.4 illustrates assignment of tasks to workstations according to a representation scheme.

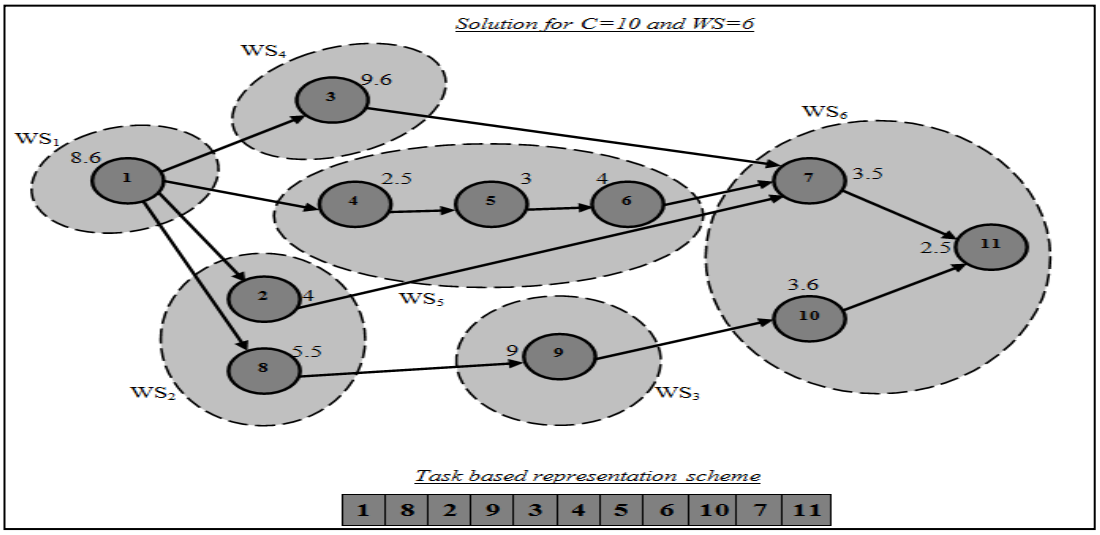


Figure 5.4 Assignment procedure according to a representation scheme (Akpınar & Bayhan, 2011)

In the proposed MCHBA, it is required to use for each bee in each colony a solution encoding mechanism, as shown in Figure 5.5, for building feasible balancing solutions belonging to initial colonies. Each bee begins by determining the available tasks for forming a feasible task assignment sequence according to the precedence constraints. Then, each bee selects randomly one task at a time among the set of available tasks according to heuristic rule assigned to its colony. After a sequence containing all tasks was generated, the artificial bee starts the procedure (see Fig. 5.4) for turning the sequence into a feasible balancing solution.

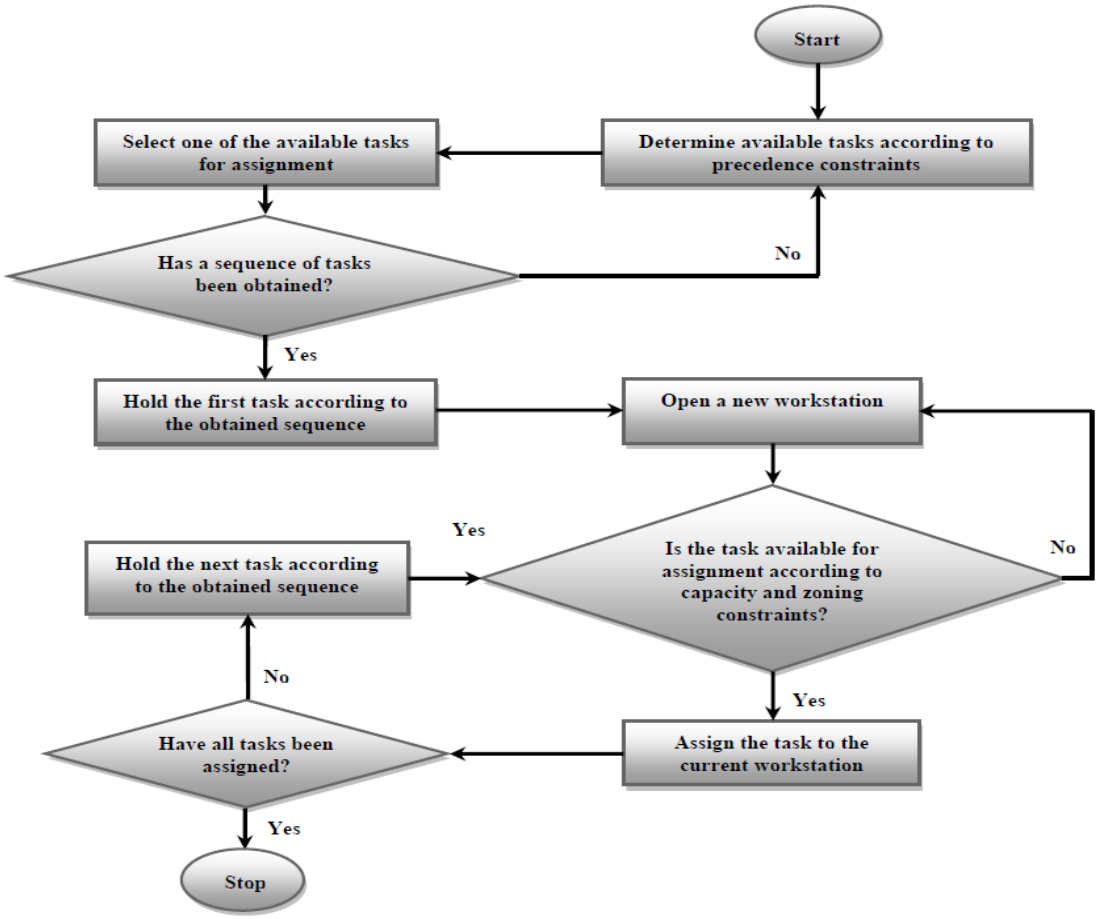


Figure 5.5 Solution encoding mechanism used while generating initial colonies

5.2.3 Fitness Evaluation

As it can be seen from the literature, the main purpose of type-I problems of assembly line balancing is minimizing the number of workstations according to a

predetermined cycle time. Besides workstation minimization, it is required additional goals to be optimized due to the complex nature of the mixed-model assembly lines. The objective function used in the current study for evaluating the solutions is given by Equation 4.2 (Vilarinho & Simaria, 2002) (see Section 4.2.2). The first term in the objective function minimizes the index of the workstation to which the last task is assigned, thus minimizing the number of workstations. The second term balances the workload between the workstations. The third term balances the workload within each workstation. Workload balancing provides the sense of equity among workers, and, contributes to increasing the output (Kim et al., 1998).

5.2.4 Neighborhood Structure

The neighborhood structure used in this study is based on the task selection strategy which was used by Vilarinho & Simaria (2006) and makes the algorithm to be a constructive type. Each bee generates a solution by selecting one task for assignment at a time instead of trying to improve an existing solution by using the proposed neighborhood structure. In the original task selection strategy, the probability of a task being selected, from the set of available tasks, is a function of: *(i)* the pheromone trail intensity between the previously selected task and each available task and *(ii)* the information provided by the heuristic rule for each available task. This information is a priority rule (one of the rules mentioned in Section 5.2.2) that is assigned to each task when the respective solution is generated.

Due to multiple colonies the proposed algorithm uses two types of pheromone matrixes, local and global pheromone matrixes. Local matrixes keep the information for each colony, while the global matrix keeps the information for all colonies and is only updated by the best solution. Moreover, the global pheromone matrix provides the communication between colonies. It is to say, global pheromone matrix is used as the information sharing mechanism.

The proposed neighborhood structure is based on the information sharing mechanism. As the original task selection strategy does, the proposed neighborhood

structure uses a random number *r* between 0 and 1 and three user defined parameters r_1, r_2 and r_3 such that $0 \leq r_1, r_2, r_3 \leq 1$ and $r_1 + r_2 + r_3 = 1$. The rule is given by:

$$j = \begin{cases} J_1 = \underset{j \in A_i^n}{argmax} \left\{ \left[\varphi . \tau_{(i,j)}^{g} + \gamma . \tau_{(i,j)}^{l} \right]^{\alpha} . \left[\eta_j \right]^{\beta} \right\} & if\ r \leq r_1 \\ J_2 : p_{(i,J_2)} = \dfrac{\left[\varphi . \tau_{(i,j_2)}^{g} + \gamma . \tau_{(i,j_2)}^{l} \right]^{\alpha} . \left[\eta_{J_2} \right]^{\beta}}{\sum_{j \in A_i^n} \left(\left[\varphi . \tau_{(i,j)}^{g} + \gamma . \tau_{(i,j)}^{l} \right]^{\alpha} . \left[\eta_j \right]^{\beta} \right)} & if\ r_1 < r \leq r_1 + r_2 \\ J_3 : random\ selection\ of\ j \in A & if\ r_1 + r_2 < r \leq r_1 + r_2 + r_3 \end{cases} \quad (5.1)$$

where $\tau_{(i,j)}^{g}$ and $\tau_{(i,j)}^{l}$ are the pheromone trail intensities kept by global and local pheromone matrixes respectively in the path 'selecting task *j* after selecting task *i*', η_j is the heuristic information of task *j* (e.g. the priority rule value for task *j*), A_i^n is the set of available tasks for bee *n* after the selection of task *i*, φ and γ are parameters that determine the relative importance of global pheromone intensity versus local pheromone intensity, and α and β are parameters that determine the relative importance of pheromone intensity versus heuristic information.

Figure 5.6 shows the solution encoding mechanism used by the proposed neighborhood structure for building a feasible balancing solution. Each artificial bee starts the mechanism by determining the available tasks for assignment to the current workstation, according to the precedence, zoning and capacity constraints. After that the bee selects one task from the set of available tasks and assigns it to the current workstation. The artificial bee opens a new workstation, if there is no task available for assignment to the current workstation. Until all tasks have been assigned to a workstation the bee repeats the procedure.

As can be seen from Equation 5.1, each bee uses two types of pheromone intensities by generating a balancing solution. The local pheromone intensity provides bees approaching to its colony's best bee while the global pheromone intensity provides bees approaching to the best bee of all colonies. This feature ensures that the bees have the behavioral properties as mentioned in Section 5.2.1.

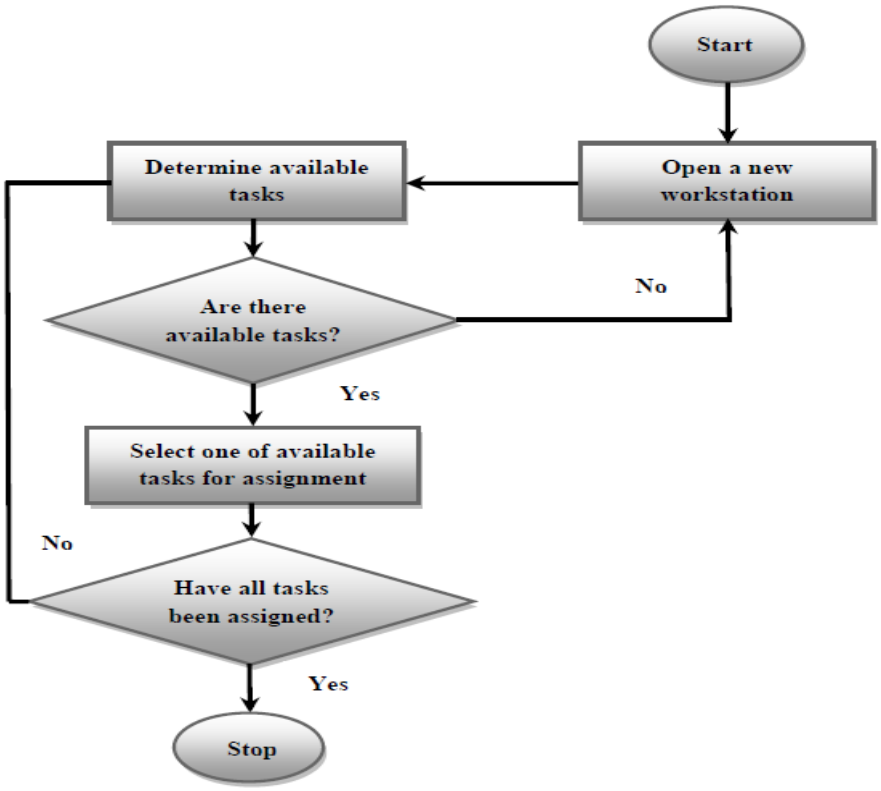


Figure 5.6 Solution encoding mechanism used by the neighborhood structure

5.3 Computational Experience

The concept of sequence dependent setup times is an actual framework in assembly line balancing problems (ALBP). For that reason, there is no standard set of benchmark instances with setup times available for testing in assembly line balancing literature except the Andrés et al.'s (2008) benchmark set, which covers simple version of ALBP, however, we deal with mixed-model version of ALBP in this study. For doing the comparison we construct a set of test problems based on type-I MMALB problems, some them (problems1-18 and 23-24) were used by Akpınar & Bayhan (2011) and some them (problems 19-22 and 25-36) were generated randomly. The main characteristics of the test problems are exhibited in Table 5.2 where N, M, and C denote the number of tasks of the combined precedence diagram, the number of models, and cycle time of the assembly line, respectively. We also classified the test problems as small-size (problems 1-4), medium-size (problems 5-22), and large-size (problems 23-36) according to the number of tasks they include.

Table 5.2 Main characteristics of the test problems

	Problem No	N	M	C	Problem Name		Problem No	N	M	C	Problem Name
Small-Size	1	8	2	10	Bowman	Medium-Size	19	53	2	10	Hahn
	2	8	3	10	Bowman		20	53	3	10	Hahn
	3	11	2	10	Gokcen & Erel		21	58	2	10	Warnecke
	4	11	3	10	Gokcen & Erel		22	58	3	10	Warnecke
Medium-Size	5	21	2	10	Mitchel	Large-Size	23	70	2	10	Tonge
	6	21	3	10	Mitchel		24	70	3	10	Tonge
	7	25	2	10	Vilarinho & Simaria		25	75	2	10	Wee-Mag
	8	25	3	10	Vilarinho & Simaria		26	75	3	10	Wee-Mag
	9	28	2	10	Heskiaoff		27	83	2	10	Arcus 1
	10	28	3	10	Heskiaoff		28	83	3	10	Arcus 1
	11	30	2	10	Sawyer		29	89	2	10	Lutz 2
	12	30	3	10	Sawyer		30	89	3	10	Lutz 2
	13	32	2	10	Lutz 1		31	94	2	10	Mukherje
	14	32	3	10	Lutz 1		32	94	3	10	Mukherje
	15	35	2	10	Gunther		33	111	2	10	Arcus 2
	16	35	3	10	Gunther		34	111	3	10	Arcus 2
	17	45	2	10	Kilbridge & Wester		35	148	2	10	Barthold
	18	45	3	10	Kilbridge & Wester		36	148	3	10	Barthold

N: Number of tasks; M: number of models; C: Cycle time

For the problems 1-18 and 23-24 the original precedence networks and operation times remained the same and for the problems 19-22 and 25-36 the precedence networks taken from http://alb.mansci.de/ and operation times were generated randomly. While generating setup times two types of setups (forward and backward setups) were considered as mentioned by Scholl et al. (2011). Thus, we defined two types of setup matrixes (forward and backward setup matrixes) as result of forward and backward setups between the tasks. All the setup times were generated according to the levels of setup time variability (Andrés et al., 2008) as mentioned in Section 4.3, and so as to fulfill the triangle inequality (Scholl et al., 2011) given by Equation 3.31(see Sec. 3.4). Moreover, considering any task may be a single element of a workstation, the pre-condition given by Equation 3.32 (see Sec. 3.3) have to be satisfied for all tasks.

As a result the constructed a benchmark set contains 108 instances in three categories, low, medium and high variability of setup times, each category contains 36 problems having the same number of tasks, precedence diagrams, and task times with different setup times.

5.3.1 Computational Results

In this study the performance of the proposed MCHBA is compared with single colony bees algorithms. Each single colony algorithm is based on one of the heuristic rules as mentioned in Section 5.2.2. We aim at determining the advantages of using multiple colonies instead of just using a single colony through such a comparison. All of the algorithms were coded in C++, and the solutions of the test problems were obtained by running the algorithms on Intel (R) Core 2 Duo CPU T7300 (2.0 GHz). The parameter sets used for single colony algorithms are given in Table 5.3 and the parameter set used for proposed MCHBA presented in Table 5.4. These parameters were chosen experimentally for getting a satisfactory performance in an acceptable time span.

Table 5.3 Parameter set for single colony algorithms

Problem	τ_0	ρ	α	β	φ	γ	r_1	r_2	r_3	S	P	e	nep	nsp	N_I
1-4 (Small Sized)	7	0.15	0.25	1.25	0.4	0.6	0.6	0.3	0.1	25	10	5	3	2	200
5-22 (Medium Sized)	20	0.15	0.25	1.25	0.4	0.6	0.6	0.3	0.1	50	25	10	5	3	200
23-36 (Large Sized)	47	0.15	0.25	1.25	0.4	0.6	0.6	0.3	0.1	100	50	20	10	5	200

τ_0: Initial pheromone level; ρ: evaporation coefficient; α and β: determine the relative importance of pheromone intensity versus heuristic information; φ and γ: determine the relative importance of global pheromone intensity versus local pheromone intensity; r_1, r_2 and r_3: user defined parameters for task selection strategy; S: Number of scout bees; P: number of employed bees; e: number of best employed bees; nep: The number of onlooker bees for each e employed bees; nsp: The number of onlooker bees for each $P - e$ employed bees (nsp<nep); N_I: Number of iterations ($MaxIter$)

According to Dorigo & Gambardella (1997), a rough approximation of the optimal value of the objective function is a reasonable value for τ_0. In the current study, since the average lower bound values (LB_{pmix}) (Akpınar et al., 2013) for cycle times are approximately equal to 7, 20 and 47 for small, medium and large sized problem classes, we set the initial pheromone levels (τ_0) to 7, 20 and 47 for small, medium and large sized problems, respectively. The other parameters, except the number of scout bees (S) the number of employed bees (P) and the number of best employed bees (e), have the same values for all test problems. S indicates the number of different solutions which must be evaluated in each iteration, thus it has to be well proportioned to the solution space size in order to satisfy algorithm's diversification. The higher values of S may the algorithm lead to redundant computational effort

while the lower values may the algorithm cause insufficient diversification. Hence, *S* is a parameter related with the search space size of a problem and should have different values for the different sized problems. Since *P* and *e* are parameters related to *S*, they should also have different values for the different sized problems.

Table 5.4 Parameter set for MCHBA

Problem	τ_0	ρ	α	β	φ	γ	r_1	r_2	r_3	S	P	e	nep	nsp	N_I
1-4 (Small Sized)	7	0.15	0.25	1.25	0.4	0.6	0.6	0.3	0.1	5	2	1	3	2	200
5-22 (Medium Sized)	20	0.15	0.25	1.25	0.4	0.6	0.6	0.3	0.1	10	5	2	5	3	200
23-36 (Large Sized)	47	0.15	0.25	1.25	0.4	0.6	0.6	0.3	0.1	20	10	4	10	5	200

As pointed out in Tables 5.3 and 5.4, the parameters *S*, *P* and *e* have different values for the proposed MCHBA and single colony algorithms. Since MCHBA has five different colonies and each colony has a number of scout bees, MCHBA evaluates all the scout bees for all colonies in each iteration. So, a single colony algorithm must be evaluating the same number of scout bees in each iteration. In such a way, it is possible to do a consistent comparison between the proposed MCHBA and the other single colony algorithms. As can be seen from Tables 5.3 and 5.4, the parameters *S*, *P* and *e* used for a single colony algorithm are five times greater than used for MCHBA.

The test problems were solved by using proposed MCHBA and other 5 heuristics (*MPTM, MAPT, MARPW, MNDS, and MTNS*) based single colony algorithms. The minimum, maximum and average values of the solutions, for each of the test problems shown in Tables 5.5, 5.6, and 5.7 results from ten runs of each instance of the problem set. Tables 5.5, 5.6, and 5.7 cover the results for the categories of low, medium, and high variability of setup times, respectively. The third and fourth columns of the Tables 5.5, 5.6, and 5.7 contain the values of LB_{pmix} and optimal solutions for the test problems respectively. The values of LB_{pmix} were provided by the procedure proposed by Akpınar et al. (2013), while the optimal solutions were provided by the MILP model developed within the scope of this study and explained in Chapter 3.

Table 5.5 Computational results for problems with low variability of setup times

	Problem No	LB_{pmix}	Optimal Solution	MPTM				MAPT				MARPW				MNDS				MTNS				MCHBA			
				N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU
				Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max	
Small Size	1	5	5	5	5	5	0.11	5	5	5	0.09	5	5	5	0.12	5	5	5	0.11	5	5	5	0.13	5	5	5	0.10
	2	6	9	9	9	9	0.18	9	9	9	0.16	9	9	9	0.19	9	9	9	0.17	9	9	9	0.18	9	9	9	0.16
	3	8	8	8	8	8	0.27	8	8	8	0.25	8	8	8	0.26	8	8	8	0.24	8	8	8	0.25	8	8	8	0.23
	4	7	7	7	7	7	0.26	7	7	7	0.28	7	7	7	0.30	7	7	7	0.26	7	7	7	0.29	7	7	7	0.27
Medium Size	5	15	-	17	17	17	1.93	17	17	17	1.81	17	17	17	1.97	17	17	17	1.88	17	17	17	1.96	17	17	17	1.76
	6	14	-	16	16	16	2.02	16	16	16	1.87	16	16	16	1.89	16	16	16	2.13	16	16	16	2.04	16	16	16	1.79
	7	15	-	17	17	17	2.56	17	17	17	2.42	17	17	17	2.65	17	17	17	2.59	17	17	17	2.65	17	17	17	2.48
	8	14	-	16.3	16	17	2.87	16.3	16	17	2.78	16	16	16	2.90	16.2	16	17	3.06	16	16	16	3.13	15	**15**	15	2.68
	9	19	-	22	22	22	6.99	22	22	22	6.35	22	22	22	7.11	22	22	22	7.30	22	22	22	7.61	22	22	22	6.69
	10	18	-	20	20	20	7.05	20	20	20	6.73	20	20	20	7.27	20	20	20	7.53	20	20	20	8.02	20	20	20	6.87
	11	15	-	17	17	17	2.87	17	17	17	2.79	17	17	17	3.04	17	17	17	2.96	17	17	17	3.27	17	17	17	2.87
	12	18	-	21.5	21	22	3.17	21.4	21	22	3.10	21.2	21	22	3.43	21	21	21	3.39	21	21	21	3.44	20	**20**	20	3.20
	13	17	-	21	21	21	3.46	21	21	21	3.45	21	21	21	3.59	21	21	21	3.39	21	21	21	3.75	21	21	21	3.37
	14	18	-	21	21	21	3.72	21	21	21	3.70	21	21	21	3.87	21	21	21	3.58	21	21	21	4.04	21	21	21	3.40
	15	20	-	25	25	25	5.87	25	25	25	5.77	25	25	25	5.33	26	26	26	6.96	25	25	25	6.45	25	25	25	5.64
	16	21	-	26.1	26	27	6.49	26.1	26	27	5.25	26.2	26	27	5.47	27	27	27	5.40	26.4	26	27	6.49	25	**25**	25	5.88
	17	23	-	27	27	27	12.77	27	27	27	11.77	27.4	27	28	13.36	27	27	27	13.91	28	28	28	13.93	26	**26**	26	12.12
	18	24	-	29	29	29	14.40	28.7	28	29	13.30	29.8	28	29	14.28	29.7	29	30	14.13	29	29	29	15.41	28	28	28	12.16
	19	26	-	34	34	34	14.07	34	34	34	14.17	33.1	33	34	14.17	34	34	34	14.11	33.9	33	34	14.17	33.1	33	34	13.46
	20	25	-	36	36	36	15.32	36	36	36	14.15	36	36	36	15.14	36	36	36	14.14	36	36	36	14.18	36	36	36	13.53
	21	31	-	34.6	34	35	16.31	34.9	34	35	15.37	34	34	34	15.47	34.4	34	35	14.48	34	34	34	14.49	34	34	34	14.64
	22	28	-	35.3	35	36	16.54	35.2	35	36	16.42	34.8	34	35	15.48	35.7	35	36	15.48	34.5	34	35	15.47	34.5	34	35	15.08
Large Size	23	39	-	47	47	47	84.13	46	46	46	81.39	46	46	46	89.15	46	46	46	96.93	46	46	46	94.30	46	46	46	84.55
	24	40	-	48.3	48	49	89.11	48.1	48	49	83.88	48	48	48	92.69	49	49	49	99.44	48.2	48	49	94.52	47	**47**	47	87.55
	25	39	-	43.1	43	44	109.63	43	43	43	107.94	42.8	42	43	107.05	43	43	43	108.50	42.9	42	43	109.27	42.3	42	43	108.73
	26	38	-	45	45	45	110.69	45.1	45	46	109.17	45	45	45	110.01	45	45	45	110.62	45.9	45	46	109.85	44.3	**44**	45	111.32
	27	44	-	54.3	54	55	113.25	54.9	54	55	114.64	53.1	53	54	113.98	55.7	55	56	113.66	53	53	53	113.11	52.6	**52**	53	112.68
	28	39	-	57.7	57	58	114.37	56.2	56	57	113.98	53	53	53	112.98	60	60	60	113.70	53	53	53	112.56	52.6	**52**	53	115.53
	29	48	-	61	61	61	128.73	60.7	60	61	129.17	59.6	59	60	127.98	60.5	60	61	128.14	59.8	59	60	128.28	59.5	59	60	126.53
	30	48	-	67.2	67	68	131.72	66.6	66	67	130.44	65	65	65	131.47	67.3	67	68	130.74	65.1	65	66	130.55	65	65	65	130.71
	31	54	-	62	62	62	148.04	62	62	62	149.03	62	62	62	147.40	63	63	63	147.75	62	62	62	147.61	61.6	**61**	62	148.28
	32	43	-	53	53	53	155.81	53.1	53	54	154.19	53	53	53	153.70	53.9	53	54	153.76	53.7	53	54	154.20	52.4	**52**	53	153.88
	33	49	-	56.4	56	57	242.59	54.5	54	55	241.74	53.2	53	54	240.63	56.3	56	57	240.41	53.6	53	54	241.03	53.2	53	54	240.34
	34	51	-	61.2	61	62	243.74	60.1	60	61	243.33	57.7	57	58	242.98	62.1	62	63	241.94	58.6	58	59	242.99	57.6	57	58	243.53
	35	69	-	79.8	79	80	306.06	79.1	79	80	305.61	78.3	78	79	305.95	79.6	79	80	306.90	78.3	78	79	309.92	78	78	78	306.14
	36	64	-	83.1	82	84	307.76	81.3	81	82	306.83	78	78	78	307.72	80.2	80	81	308.18	79.2	79	80	309.54	77.6	**77**	78	309.10

N_{WS}: Number of workstations; **CPU**: Computational time

Table 5.6 Computational results for problems with medium variability of setup times

Problem No		LB_{pmix}	Optimal Solution	MPTM				MAPT				MARPW				MNDS				MTNS				MCHBA			
				N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU
				Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max	
Small Size	1	5	5	5	5	5	0.12	5	5	5	0.13	5	5	5	0.11	5	5	5	0.11	5	5	5	0.10	5	5	5	0.12
	2	6	9	9	9	9	0.16	9	9	9	0.15	9	9	9	0.16	9	9	9	0.17	9	9	9	0.18	9	9	9	0.21
	3	8	8	8	8	8	0.25	8	8	8	0.24	8	8	8	0.26	8	8	8	0.26	8	8	8	0.27	8	8	8	0.22
	4	7	7	7	7	7	0.27	7	7	7	0.28	7	7	7	0.27	7	7	7	0.29	7	7	7	0.25	7	7	7	0.26
Medium Size	5	15	-	17	17	17	1.98	17	17	17	1.97	17	17	17	2.00	17	17	17	1.85	17	17	17	1.97	17	17	17	1.84
	6	14	-	17	17	17	2.10	17	17	17	1.97	17	17	17	1.87	17	17	17	1.94	17	17	17	1.94	17	17	17	1.78
	7	15	-	18	18	18	2.57	18	18	18	2.51	18	18	18	2.76	18	18	18	2.75	18	18	18	3.08	18	18	18	2.58
	8	14	-	16	16	16	3.02	16	16	16	2.70	16	16	16	2.89	16	16	16	3.34	16	16	16	3. 24	16	16	16	2.69
	9	19	-	22	22	22	7.29	22	22	22	6.49	22	22	22	7.60	22	22	22	7.26	22	22	22	7.90	22	22	22	6.93
	10	18	-	21	21	21	7.43	21	21	21	6.62	21	21	21	7.75	21	21	21	7.43	21	21	21	8.07	21	21	21	7.06
	11	15	-	18	18	18	3.32	18	18	18	2.89	18	18	18	3.06	18	18	18	3.02	18	18	18	3.29	18	18	18	3.14
	12	18	-	23	23	23	3.13	22.6	22	23	3.01	22.2	22	23	3.57	23	23	23	3.52	23	23	23	3.65	21	**21**	21	4.51
	13	17	-	21	21	21	3.64	21	21	21	3.29	21	21	21	3.56	21	21	21	3.40	21	21	21	3.74	21	21	21	3.68
	14	18	-	23	23	23	4.03	23	23	23	3.50	23	23	23	3.93	23	23	23	3.90	23	23	23	3.89	23	23	23	3.79
	15	20	-	26	26	26	6.52	26	26	26	5.80	26	26	26	6.48	27	27	27	6.20	26	26	26	7.06	26	26	26	6.67
	16	21	-	26	26	26	6.34	26	26	26	5.15	26	26	26	7.08	27	27	27	6.69	26	26	26	7.35	26	26	26	6.46
	17	23	-	27	27	27	14.05	27	27	27	12.85	27	27	27	14.62	27	27	27	13.88	27	27	27	14.62	26.4	**26**	27	13.10
	18	24	-	29	29	29	14.54	29	29	29	12.69	29	29	29	14.45	29	29	29	13.50	29	29	29	14.65	29	29	29	13.64
	19	26	-	36	36	36	13.14	36	36	36	13.17	36	36	36	14.23	37	37	37	14.13	35.8	35	36	14.19	35.5	35	36	14.78
	20	25	-	37	37	37	16.15	37	37	37	15.11	37	37	37	15.17	37	37	37	16.11	37	37	37	16.17	37	37	37	15.65
	21	31	-	35.8	35	36	17.37	36	36	36	16.33	35	35	35	16.53	35	35	35	17.40	35	35	35	17.53	35	35	35	17.20
	22	28	-	36.3	36	37	18.43	36.8	36	37	17.41	36.1	36	37	17.51	37.9	37	38	18.46	36.6	36	37	17.49	36	36	36	17.86
Large Size	23	39	-	48.2	48	49	93.64	48	48	48	84.22	48.3	48	49	90.43	48.1	48	49	97.14	48.3	48	49	98.35	47.5	**47**	48	89.20
	24	40	-	51.4	51	52	94.48	51	51	51	85.38	51	51	51	94.80	51	51	51	96.08	51	51	51	99.73	50	**50**	50	96.21
	25	39	-	44	44	44	109.62	44	44	44	109.46	44	44	44	110.54	44	44	44	110.92	44	44	44	111.42	43.4	**43**	44	110.21
	26	38	-	45.7	45	46	112.98	45.8	45	46	113.61	46	46	46	112.56	46	46	46	112.88	46.1	46	47	113.73	45.4	45	46	114.13
	27	44	-	55.1	55	56	117.49	55.7	55	56	118.65	54.7	54	55	117.12	56.7	56	57	117.36	54.9	54	55	117.26	54.6	54	55	117.41
	28	39	-	58.8	58	60	120.65	57	57	57	119.53	54	54	54	119.87	61	61	61	119.89	54.2	54	55	119.78	53.5	**53**	54	119.80
	29	48	-	62.9	62	63	130.89	63	63	63	130.60	61.3	61	62	130.35	62	62	62	129.32	61	61	61	130.40	60.4	**60**	61	129.18
	30	48	-	68.9	68	69	132.51	67.2	67	68	131.53	67	67	67	131.60	68.8	68	69	131.29	67.2	67	68	131.72	66.6	**66**	67	133.29
	31	54	-	64	64	64	149.42	63.8	63	64	150.82	64.1	64	65	150.78	65	65	65	150.45	65	65	65	151.45	63.6	63	64	151.13
	32	43	-	55	55	55	157.46	55	55	55	154.25	55	55	55	155.64	55.8	55	56	156.01	55	55	55	156.08	54.7	**54**	55	155.39
	33	49	-	57.7	57	58	239.03	56	56	56	238.38	54.8	54	55	237.18	58.1	58	59	237.73	54.6	54	55	237.99	54.4	54	55	238.14
	34	51	-	63	63	63	247.03	62.8	62	63	248.64	60	60	60	249.11	64.4	64	65	248.84	60.8	60	61	248.42	59.4	**59**	60	253.84
	35	69	-	83	83	83	311.49	82.2	82	83	312.83	80.6	80	81	312.52	81.5	81	82	311.03	80.9	80	81	312.48	80.3	80	81	313.24
	36	64	-	85.4	85	86	314.71	85.1	84	86	313.71	81	81	81	313.70	83	83	83	312.94	81.6	81	82	313.51	80.7	**80**	81	313.87

N_{WS}: Number of workstations; **CPU**: Computational time

Table 5.7 Computational results for problems with high variability of setup times

	Problem No	LB_{pmix}	Optimal Solution	MPTM				MAPT				MARPW				MNDS				MTNS				MCHBA			
				N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU	N_{WS}			CPU
				Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max		Avg	Min	Max	
Small Size	1	5	5	5	5	5	0.11	5	5	5	0.12	5	5	5	0.10	5	5	5	0.11	5	5	5	0.09	5	5	5	0.11
	2	6	10	10	10	10	0.22	10	10	10	0.21	10	10	10	0.20	10	10	10	0.22	10	10	10	0.19	10	10	10	0.17
	3	8	8	8	8	8	0.29	8	8	8	0.30	8	8	8	0.27	8	8	8	0.30	8	8	8	0.23	8	8	8	0.22
	4	7	7	7	7	7	0.30	7	7	7	0.31	7	7	7	0.32	7	7	7	0.31	7	7	7	0.38	7	7	7	0.25
Medium Size	5	15	-	18	18	18	2.29	18	18	18	2.04	18	18	18	2.08	18	18	18	1.97	18	18	18	2.03	18	18	18	2.09
	6	14	-	18	18	18	2.12	18	18	18	1.97	18	18	18	2.03	18	18	18	2.01	18	18	18	2.11	18	18	18	1.97
	7	15	-	18	18	18	3.00	18	18	18	2.71	19	19	19	2.92	18.8	18	19	2.80	19	19	19	3.00	18	18	18	2.73
	8	14	-	18.2	18	19	3.23	18.3	18	19	3.02	18	18	18	3.17	18	18	18	3.34	18.1	18	19	3.31	17	**17**	17	2.84
	9	19	-	23.4	23	24	6.72	23.2	23	24	6.32	23	23	23	7.57	23	23	23	7.31	23.6	23	24	7.78	22	**22**	22	6.77
	10	18	-	21	21	21	7.02	21	21	21	6.76	21	21	21	7.71	21	21	21	7.57	21	21	21	7.95	21	21	21	7.06
	11	15	-	18	18	18	3.22	18	18	18	3.01	18	18	18	3.35	18	18	18	3.06	18	18	18	3.37	18	18	18	2.96
	12	18	-	23	23	23	3.46	23	23	23	3.04	23	23	23	3.77	23	23	23	3.56	23	23	23	3.55	23	23	23	3.36
	13	17	-	22	22	22	3.70	22	22	22	3.29	22	22	22	3.51	22	22	22	3.56	22	22	22	3.84	22	22	22	3.51
	14	18	-	25	25	25	4.09	25	25	25	3.73	25	25	25	4.12	25	25	25	3.81	25	25	25	4.22	25	25	25	3.67
	15	20	-	27	27	27	6.93	27	27	27	5.00	27	27	27	6.97	27	27	27	6.94	27	27	27	7.01	27	27	27	5.67
	16	21	-	27	27	27	6.08	27	27	27	5.25	27	27	27	7.38	27.4	27	28	5.79	27	27	27	6.42	26	**26**	26	5.77
	17	23	-	28	28	28	13.34	28	28	28	11.80	28	28	28	13.98	28	28	28	14.27	28	28	28	13.89	27	**27**	27	12.19
	18	24	-	30	30	30	14.77	30	30	30	12.99	30	30	30	14.77	30	30	30	15.26	30	30	30	14.77	30	30	30	13.88
	19	26	-	36.3	36	37	16.17	36	36	36	15.14	36.1	36	37	16.21	36.4	36	37	15.15	36.9	36	37	16.23	35.8	**35**	36	15.17
	20	25	-	37.3	37	38	15.21	37	37	37	16.12	37	37	37	16.16	38	38	38	15.15	37.2	37	38	16.19	37	37	37	16.35
	21	31	-	37.8	37	38	19.40	37	37	37	18.36	36	36	36	18.49	36.9	36	37	18.47	36.8	36	37	19.51	36	36	36	18.11
	22	28	-	37	37	37	20.34	36.9	36	37	19.37	36.7	36	37	19.49	38.1	38	39	19.54	36	36	36	20.52	36	36	36	19.34
Large Size	23	39	-	49	49	49	95.01	49.3	49	50	85.83	49	49	49	92.71	49	49	49	95.80	49.5	49	50	96.12	48	**48**	48	89.98
	24	40	-	52.2	52	53	99.39	52.1	52	53	87.61	52	52	52	97.23	52	52	52	99.18	52	52	52	98.63	51.5	**51**	52	97.97
	25	39	-	44.4	44	45	107.61	44	44	44	107.56	44.3	44	45	108.22	44.4	44	45	108.21	44.5	44	45	107.03	44	44	44	108.62
	26	38	-	46.1	46	47	113.78	46.2	46	47	112.77	46	46	46	113.18	47	47	47	112.15	46.9	46	47	113.09	45.7	**45**	46	112.36
	27	44	-	57.4	56	58	116.35	57.6	57	58	115.57	55.8	55	56	115.13	58.5	58	59	114.98	55.8	55	56	114.49	55.5	55	56	116.76
	28	39	-	59.5	59	60	118.68	58.1	58	59	117.64	54.3	54	55	118.06	61.5	61	62	118.14	54.7	54	55	117.32	54.1	54	55	118.88
	29	48	-	63.6	63	64	119.82	64	64	64	118.03	62.7	62	63	116.50	63.9	63	64	117.07	62.7	62	63	116.90	62.5	62	63	113.76
	30	48	-	72.8	72	73	120.74	71.2	71	72	118.76	69.1	69	70	117.64	71.5	71	72	116.68	69	69	69	116.47	68.5	**68**	69	115.42
	31	54	-	64.8	64	65	150.62	64.8	64	65	151.08	64.9	64	65	152.54	65.7	65	66	151.93	65	65	65	153.48	64.5	64	65	153.53
	32	43	-	55.7	55	56	155.11	55.5	55	56	155.30	55.5	55	56	154.88	57	57	57	155.46	55.8	55	56	154.26	54.6	**54**	55	156.01
	33	49	-	58.5	58	59	242.09	56.9	56	57	240.06	56	56	56	240.91	58.9	58	60	242.68	56.8	56	57	240.55	55.7	**55**	56	241.02
	34	51	-	64.6	64	65	258.43	63.3	63	64	257.55	61	61	61	256.95	64.9	64	65	256.86	61.6	61	62	256.88	60.6	**60**	61	257.52
	35	69	-	85	85	85	314.56	84	84	84	315.15	81	81	81	314.23	82.7	82	83	314.29	81.8	81	82	313.54	80.6	**80**	81	314.40
	36	64	-	87.6	87	88	315.43	86.4	85	87	314.31	81.8	81	82	316.09	84	83	85	315.44	82.9	82	83	315.98	81.6	81	82	316.17

N_{WS}: Number of workstations; **CPU**: Computational time

As can be seen from the Tables 5.5, 5.6, and 5.7 the mixed integer linear programming model (see Chapter 3) cannot solve optimality as the problem size increased due to its complex nature. On the other hand, the proposed multiple colony hybrid bees algorithm is able to provide satisfactory solutions in reasonable running times.

We conduct comparisons between the proposed multiple colony hybrid bees algorithm and the other single colony algorithms into three categories, low variability of setup times, medium variability of setup times, and high variability of setup times, in terms of determined number of workstations (N_{WS}) and computational time (CPU). As can be seen from Tables 5.5, 5.6, and 5.7 multiple colony hybrid bees algorithm outperforms single colony algorithms for medium and large sized problems in case of low, medium and high variability of setups. It must be noted that as the problem size get larger the performance of multiple colony hybrid bees algorithm increase significantly in comparison with single colony algorithms. In order to make the comparisons more clearly in terms of gaps for the obtained minimum number of workstations we formed Table 5.8 as a summary of the Tables 5.5, 5.6, and 5.7.

From the observation of Table 5.8, it is found that the performance of multiple colony hybrid bees algorithm is superior to MPTM, MAPT, MARPW, MNDS, and MTNS in 51.85% (56 of 108 problems), 49.07% (53 of 108 problems), 36.11% (39 of 108 problems), 56.48% (61 of 108 problems), and 38.89% (42 of 108 problems) of the test problems, respectively.

In order to understand whether the differences in the obtained results are due to the random chance or not, a paired *t*-test is executed on Excel 2007. After paired *t*-test it is figured out that outputs (average values for 10 independent replications represented in Tables 5.5, 5.6, and 5.7) derived from multiple colony hybrid bees algorithm are meaningfully different from single colony algorithms. The results of paired *t*-test are presented in Table 5.9. Based on the values from Table 5.9, it can be concluded that multiple colony hybrid bees algorithm produced better results than single colony algorithms in general.

Table 5.8 Comparisons MCHBA versus single colony algorithms

	Problem No	Small Size				Medium Size																		Large Size													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
GAPs Low Variability	MCHBA																																				
	MPTM	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	1	1	1	1	-	-	1	1	1	1	1	2	5	2	2	1	1	3	4	1	5
	MCHBA																																				
	MAPT	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	1	1	-	1	-	-	1	-	1	1	1	2	4	1	1	1	1	1	3	1	4
	MCHBA																																				
	MARPW	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	1	1	-	-	-	-	-	-	1	-	1	1	1	-	-	1	1	-	-	-	1
	MCHBA																																				
	MNDS	-	-	-	-	-	-	-	1	-	-	-	1	-	-	1	2	1	1	1	-	-	1	-	2	1	1	3	8	1	2	2	1	3	5	1	3
	MCHBA																																				
	MTNS	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	1	2	1	-	-	-	-	-	1	-	1	1	1	-	-	1	1	-	1	-	2
GAPs Medium Variability	MCHBA																																				
	MPTM	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	1	-	1	-	-	-	1	1	1	-	1	5	2	2	1	1	3	4	3	5
	MCHBA																																				
	MAPT	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	1	-	1	-	1	1	1	-	1	4	3	1	-	1	2	3	2	4
	MCHBA																																				
	MARPW	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	1	-	-	-	1	1	1	1	-	1	1	1	1	1	-	1	-	1
	MCHBA																																				
	MNDS	-	-	-	-	-	-	-	-	-	-	-	2	-	-	1	1	1	-	2	-	-	1	1	1	1	1	2	8	2	2	2	1	4	5	1	3
	MCHBA																																				
	MTNS	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	1	-	-	-	-	-	1	1	1	1	-	1	1	1	2	1	-	1	-	1
High Variability	MCHBA																																				
	MPTM	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	1	1	-	1	-	1	1	1	1	-	1	1	5	1	4	-	1	3	4	5	6
	MCHBA																																				
	MAPT	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	1	1	-	1	-	1	-	1	1	-	1	2	4	2	3	-	1	1	3	4	4
	MCHBA																																				
	MARPW	-	-	-	-	-	-	1	1	1	-	-	-	-	-	-	1	1	-	1	-	-	-	1	1	-	1	-	-	-	1	-	1	1	1	1	-
	MCHBA																																				
	MNDS	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	1	1	-	1	1	-	2	1	1	-	2	3	7	1	3	1	3	3	4	2	2
	MCHBA																																				
	MTNS	-	-	-	-	-	-	1	1	1	-	-	-	-	-	-	1	1	-	1	-	-	-	1	1	-	1	-	-	-	1	1	1	1	1	1	1

Table 5.9 *p* values for the comparison of MCHBA and single colony algorithms (α=0.05)

	p Value		
	Low Variability	Medium Variability	High Variability
MCHBA MPTM	0.000032	0.000263	0.000079
MCHBA MAPT	0.000009	0.000196	0.000052
MCHBA MARPW	0.000116	0.000018	0.000003
MCHBA MNDS	0.000029	0.000128	0.000021
MCHBA MTNS	0.000017	0.000031	0.0000004

Considering the average computational times (*CPUs*), all the algorithms solve the problems in almost equally amount of times. This situation may be explained by the used parameter sets, hence, it must be noted that the computational time is related to the used parameter set. This set of computational experiments shows that the overall performance of MCHBA is superior to single colony algorithms.

5.4 Chapter Conclusions

In this chapter, we aimed at developing a new multiple colonies Bees Algorithm in order to improve the search ability of the basic Bees Algorithm for solving the mixed-model assembly line balancing problem with sequence dependent setup times between tasks. In the proposed MCHBA, a new neighborhood structure, which was based on the task selection strategy of Ant Colony Optimization, was used. In this manner, this new neighborhood structure ensures our developed algorithm to be a constructive type. In order to evaluate the real performance of the proposed MCHBA, a set of 36 mixed-model assembly line balancing problems with parallel workstations and zoning constraints was tested. As to the scope of this study, this set of benchmark problems was differentiated by adding sequence dependent setup times between tasks with low, medium and high variability. The performance of the proposed MCHBA was also compared with the performances of single colony algorithms. Computational results indicate that the new neighborhood structure and multiple colonies can improve search performance of the basic Bees Algorithm.

CHAPTER SIX
CONCLUSIONS

6.1 Summary

This dissertation dealt with the balancing problem of assembly lines by taking into consideration the sequence dependent setup times, which is an actual framework in assembly line balancing problems. Most of the studies on assembly line balancing problems with sequence dependent setup times have focused extensively on single model lines; however, single model assembly lines are not able to respond the demand for higher product variability anymore because of the current consumer-centric market conditions. Thus, mixed-model assembly lines substitute for single model assembly lines. That is to say, high-mix/low-volume manufacturing strategies substitute for low-mix/high-volume manufacturing strategies.

Since the existing literature on setups only covers the single model assembly lines, the lack of studies dealing with the consideration of setups for mixed-model assembly lines stands out. Under this conditions, the main goal of this dissertation was to introduce the type-I mixed-model assembly line balancing problem with setups (MMALBPS-I), which is an extension of classical MMALBP-I and takes into consideration the sequence dependent setup times between tasks.

Within this context, firstly, we have developed a mixed integer linear programming (MILP) model by considering the phenomena of sequence dependent setup times for mixed-model assembly, in order to formally describe the problem. However, due to the NP-Hard nature of the problem the proposed MILP model was not able to solve large scale problems. Therefore, we have developed meta-heuristics based hybrid algorithms in order to tackle the problem. Among the meta-heuristics, we have selected genetic algorithm, ant colony optimization, and bess algorithm and we have developed effective hybrid algorithms based on these three meta-heuristics. Computational experiments were carried out in order to determine the capability of the developed MILP and the performances of the proposed hybrid algorithms.

6.2 Contributions of the Dissertation

In Chapter 3, we have developed a MILP model for MMALBPS-1. The MILP provides us to formally formulate the MMALBPS-I. Moreover, our MILP can solve the problem with and without sequence dependent setup times, parallel workstation assignments, and zoning constraints. Since, the SALBP-I is a special case of MMALBP-I, our MILP is able to solve SALBP-I with and without the aforementioned characteristics. Thus, we can conclude that our MILP is a general model for some of the assembly line balancing problems.

In Chapter 4, we have proposed a hybrid ACO-GA algorithm. In the proposed hybrid ACO-GA algorithm, GA was embedded into ACO. The proposed ACO-GA algorithm enhanced the performance of ACO by incorporating GA as a local search strategy for MMALBPS-I. In the proposed hybrid algorithm ACO was conducted to provide diversification, while GA was conducted to provide intensification. The rationale why we attempted to hybridize ACO with GA was to exploit the complementary character of different optimization strategies. Viz, our proposed hybrid algorithm integrated the positive feedback mechanism and the satisfactory performance of ACO with the faster speed of GA. Thus, the proposed hybrid ACO-GA algorithm attempted to overcome the slower speed of ACO and the poor searching capability of GA, especially for large sized problems, by embedding GA into ACO as a local search. Furthermore, ACO-GA utilized the synergy of GA as an improvement procedure and ACO as a constructive procedure.

In Chapter 5, we have proposed a multiple colony hybrid Bees algorithm for MMALBPS-I. Our proposed approach was based on the multiple colonies; however, most of the existing literature about the applications of BA to combinatorial optimization tries to evolve only a single population. The phenomena of multiple colonies were used with the purpose of improving the diversification of the algorithm, which refers to the ability to visit many and different regions of the search space. Moreover, we used a new neighbourhood structure which ensures the algorithm to be a constructive type, since the constructive type of search algorithms

like ACO are much more effective for the assembly line balancing problems. This neighbourhood structure also enables the proposed approach to utilize the positive feedback mechanism as ACO does. Due to the multiple colonies, we have adopted a mechanism provides the communication between different colonies in the proposed algorithm in order to share the information, since information sharing is an essential issue from the optimization point of view.

6.3 Future Research Directions

During this dissertation some research areas have become clear that can influence the further research directions. Within the scope of this dissertation we can classify future researches into three groups as follows.

In Chapter 3, we have developed a general MILP model for some of the assembly line balancing problems with regard to some characteristics. In further researches, we might extend the proposed MILP so as to solve different assembly line balancing problems with different line configurations.

In Chapters 4 and 5, we have developed two different hybrid algorithms; hybrid ACO-GA algorithm and multiple colony hybrid bees algorithm. Future researches will focus on applying both algorithms to different types of assembly line balancing problems. Moreover, both the algorithms may be arranged so as to solve different combinatorial optimization problems or so as to implement to continuous domains.

Due to the multiple colony structure of the proposed hybrid bees algorithm, future researches will focus on the parallel/distributed applications of MCHBA to different combinatorial optimization problems.

REFERENCES

Akbari, R, & Ziarati, K. (2011). A cooperative approach to bee swarm optimization. *Journal of Information Science and Engineering, 27*, 799–818.

Akpınar, S, & Bayhan, G. M. (2011). A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints. *Engineering Applications of Artificial Intelligence, 24*(3), 449–457.

Akpınar, S, Bayhan, G. M., & Baykasoglu, A. (2013). Hybridizing ant colony optimization via genetic algorithm for mixedmodel assembly line balancing problem with sequence dependent setup times between tasks. *Applied Soft Computing, 13*(1), 574-589.

Allahverdi,A., Gupta, J. N. D., & Aldowaisan,T. (1999). A review of scheduling research involving setup considerations. *OMEGA the International Journal of Management Sciences, 27*, 219–239.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, *187*(3), 985–1032.

Andrés, C., Miralles, C., & Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, *187*(3), 1212–1223.

Askin, R. G., & Zhou, M. (1997). A parallel station heuristic for the mixed-model production line balancing problem. *International Journal of Production Research, 35*, 3095-3106.

Battaïa, O., & Dolgui, A. (2012a). Reduction approaches for a generalized line balancing problem. *Computers and Operations Research*, *39*, 2337–2345.

Battaïa, O., & Dolgui, A. (2012b). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, http://dx.doi.org/10.1016/j.ijpe.2012.10.020.

Bautista, J., & Pereira, J. (2002). Ant algorithms for assembly line balancing. *Lecture Notes in Computer Science, 24*(63), 65–75.

Baykasoğlu, A., Ozbakır, L., & Tapkan, P. (2007). Artificial bee colony algorithm and its application to generalized assignment problem. In F. T. S., Chan, & M.K. Tiwari (Ed.). *Swarm intelligence: Focus on ant and particle swarm optimization* (113-144). Vienna–Austria: I–Tech Education and Publishing.

Baykasoglu, A., Dereli, T., & Sabuncu, I. (2006). An ant colony based algorithm for solving budget constrained and unconstrained dynamic facility layout problems. *Omega: International Journal of Management Science, 34*, 385–396.

Baykasoğlu, A. (2006). Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. *Journal of Intelligent Manufacturing, 17*, 217-232.

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

Blum,C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing, 11*(6), 4135-4151.

Bock, S. (2006). Using distributed search methods for balancing mixed-model assembly lines in the automotive industry. *OR Spectrum, 30*(3), 551-578.

Bock, S. (2008). Supporting offshoring and nearshoring decisions for mass customization manufacturing processes. *European Journal of Operational Research, 184,* 490-508.

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

Boysen, N., Fliedner, M. & Scholl, A. (2008). Assembly line balancing: Which model to use when? *International Journal of Production Economics, 111,* 509-528.

Bukchin, Y., & Rabinowitch, I. (2006). A branch-and-bound based solution approach for the mixed-model assembly line balancing problem for minimizing stations and task duplication costs. *European Journal of Operational Research, 174,* 492-508.

Bukchin, J., Dar-El, E. M., & Rubinovitz, J. (2002). Mixed model assembly line design in a make-to-order environment. *Computers and Industrial Engineering, 41,* 405-421.

Crainic, T. G., & Toulouse, M. (2003). Parallel strategies for meta-heuristics. In F. Glover, & G. Kochenberger, (Ed.). *Handbook of Metaheuristics* (475-513). Kluwer Academic Publishers, Dordrecht.

Dorigo, M. A., & Gambardella, L. (1996). *Ant colonies for the travelling salesman problem*. TR/IRIDIA/1996-3, Université Libre de Bruxelles, Belgium.

Dorigo, M. A., & Gambardella, L. (1997). *Ant colony system: a cooperative learning approach to the traveling salesman problem*. TR/IRIDIA/1996-5, Université Libre de Bruxelles, Belgium.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991). *Positive feedback as a search strategy*. Technical Report No: 91-016, Politecnico di Milano, Italy.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, *26*(1), 1-13.

Dorigo, M., Di Caro, G., & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, *5*, 137-172.

Erel, E. & Gokcen, H. (1999). Shortest-route formulation of mixed-model assembly line balancing problem. *Europen Journal of Operational Research*, *116*, 194-204.

Erel, E & Sarin S. C. (1998). A survey of the assembly line balancing procedures. *Production Planning and Control*, *9*(5), 414-434.

Gokcen, H. & Erel, E. (1997). A goal programming approach to mixed-model assembly line balancing problem. *International Journal of Production Economics*, *48*(2), 177-185.

Gökçen, H. & Erel, E. (1998) Binary integer formulation for mixed-model assembly line balancing problem. *Computers and Industrial Engineering*, *23*, 451-461.

Goldberg, D.E. (1989). *GAs in search, optimization and machine learning*. Reading, Massachusetts: Addison-Wesley.

Hamta, N., Fatemi Ghomi, S. M. T., Jolai, F., & Shirazi, M. A. (2012). A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. *International Journal of Production Economics*, *141*(1), 99-111.

Hamzadayi,A., & Yildiz, G. (2012). A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints. *Computers and Industrial Engineering*, *62*(1), 206-215

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan.

Hop, N. V. (2006). A heuristic solution for fuzzy mixed-model line balancing problem. *European Journal of Operational Research*, *168*, 798-810.

Hwang, R. K., & Katamaya, H. (2009). A multi-decision genetic approach for workload balancing of mixed-model U-shaped assembly line systems. *International Journal of Production Research*, *47*(14), 3797-3822.

Hwang, R. K., & Katamaya, H. (2010). Integrated procedure of balancing and sequencing for mixed-model assembly lines: a multi-objective evolutionary approach. *International Journal of Production Research*, *48*(21), 6417-6441.

Kara, Y., Özcan, U., & Peker, A. (2007). An approach for balancing and sequencing mixed-model JIT U-lines. *International Journal of Advanced Manufacturing Technology*, *32*, 1218-1231.

Kara,Y., Özgüven, C. Seçme, N. Y., & Chang, C. T. (2011). Multi-objective approaches to balance mixed-model assembly lines for model mixes having precedence conflicts and duplicate common tasks. *International Journal of Advanced Manufacturing Technology*, *52*, 725-737.

Karaboga, D., & Akay, B. (2009). A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, *31*, 61–85.

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*. Technical Report TR06, Computer Engineering Department, Engineering Faculty, Erciyes University, Turkey.

Kazemi, S. M., Ghodsi, R., Rabbani, M., & Moghaddam, R. T. (2011). A novel two-stage genetic algorithm for a mixed-model U-line balancing problem with duplicated tasks. *International Journal of Advanced Manufacturing Technology*, *55*, 1111–1122.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *In IEEE international conference on neural networks*, Piscataway–NJ, 1942–1948.

Kim, Y. J., Kim, Y. K. & Cho, Y. (1998) A heuristic-based genetic algorithm for workload smoothing in assembly lines. *Computers and Operations Research*, *25*, 99-111.

Lee, Z. J., Su, S. F., Chuang, C. C., & Liu, K. H. (2008). Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment. *Applied Soft Computing*, *8*, 55-78.

Leu, Y. Y., Matheson, L. A., & Rees, L. P. (1994). Assembly line balancing using genetic algorithms with heuristic generated initial populations and multiple criteria. *Decision Sciences*, *15*, 581-606.

Lozano, M., & García-Martínez, C. (2010). Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers and Operations Research*, *37*, 481–497.

Martino, L., & Pastor, R. (2010). Heuristic procedures for solving the general assembly line balancing problem with setups. *International Journal of Production Research*, *48*(6), 1787–1804.

McMullen, P. R., & Frazier, G. V. (1998). Using simulated annealing to solve a multiobjective line balancing problem with parallel workstations. *International Journal of Production Research, 36*, 2717–2741.

McMullen, P. R., & Frazier, G. V. (1997). A heuristic for solving mixed-model line balancing problems with stochastic task durations and parallel stations. *International Journal of Production Economics, 51,* 177-190.

McMullen, P. R. & Tarasewich, P. (2003). Using ant techniques to solve the assembly line balancing problem. *IIE Transactions: Design and Manufacturing, 35*(7), 605-617.

Mendes, A. R., Ramos, A. L., Simaria, A. S., & Vilarinho, P. M. (2005). Combining heuristic procedures and simulation models for balancing a PC camera assembly line. *Computers and Industrial Engineering, 49*, 413-431.

Merengo, C., Nava, F., & Pozzetti, A. (1999). Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research, 37*, 2835-2860.

Miltenburg, J. (2002). Balancing and scheduling mixed-model U-shaped production lines. *International Journal of Flexible Manufacturing Systems, 14*, 119-151.

Noorul Haq, A., Jayaprakash, J., & Rengarajan, K. (2006). A hybrid genetic algorithm approach to mixed-model assembly line balancing. *International Journal of Advanced Manufacturing Technology, 28*, 337-341.

Ozbakır, L., & Tapkan, P. (2011). Bee colony intelligence in zone constrained two–sided assembly line balancing problem. *Expert System with Applications, 38*, 11947–11957.

Ozbakir, L., Baykasoglu, A., Gorkemli, B., & Gorkemli, L. (2011). Multiple–colony ant algorithm for parallel assembly line balancing problem. *Applied Soft Computing, 11*, 3186–3198.

Özbakır, L., Baykasoğlu, A., & Tapkan, P. (2010). Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation, 215*(11), 3782-3795.

Özcan, U., & Toklu, B. (2009). Balancing of mixed-model two-sided assembly lines. *Computers and Industrial Engineering, 57*(1), 217-227.

Ozcan, U., & Toklu, B. (2010). Balancing two–sided assembly lines with sequence–dependent setup times. *International Journal of Production Research, 48*(18), 5363–5383.

Özcan, U., Çerçioğlu, H., Gökçen, H., & Toklu, B. (2010). Balancing and sequencing of parallel mixed-model assembly lines. *International Journal of Production Research, 48*(17), 5089-5113.

Pastor, R., Andrés, C., & Miralles, C. (2010). Corrigendum to 'Balancing and scheduling tasks in assembly lines with sequence-dependent setup' [European Journal of Operational Research 2008; 187: 1212–1223]. *European Journal of Operational Research, 201*(1), 336.

Pham, D. T., Koc, E., Ghanbarzadeh, A., Otri, S., Rahim, S., & Zaidi, M. (2006). The bees algorithm – A novel tool for complex optimisation problems. *In Proceedings of IPROMS 2006 conference*, 454–461.

Ponnambalam, S. G., Aravindan, P. & Naidu G. M. (1999). A comparative evaluation of assembly line balancing heuristics. *International Journal of Advanced Manufacturing Technology, 15*, 577-586.

Preux, P., & Talbi, E. G. (1999). Towards hybrid evolutionary algorithms. *International Transactions in Operational Research*, *6*(6), 557-570.

Raidl, G. R. (2006). A unified view on hybrid metaheuristics. In Francisco Almeida et al., editors, *Lecture Notes in Computer Science, 4030*, 1-12.

Sabuncuoglu, I., Erel, E., & Tanyer, M. (2000). Assembly line balancing using genetic algorithms. *Journal of Intelligent Manufacturing*, *11*(3), 295-310.

Salveson, M. E., 1955. The assembly line balancing problem. *Journal of Industrial Engineering*, *6*, 18-25.

Scholl, A., Boysen, N., & Fliedner, M. (2008). The sequence–dependent assembly line balancing problem. *OR Spectrum*, *30*(3), 579–609.

Scholl, A., Boysen, N., & Fliedner, M. (2011). The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics. *OR Spectrum*, doi 10.1007/s00291-011-0265-0.

Scholl, A. (1999). *Balancing and Sequencing of Assembly Lines*. Physica-Verlag, Heidelberg.

Seyed-Alagheband, S. A., Fatemi Ghomi, S. M. T., & Zandieh, M. (2011). A simulated annealing algorithm for balancing the assembly line type II problem with sequence–dependent setup times between tasks. *International Journal of Production Research*, *49*(3), 805–825.

Simaria, A. S., & Vilarinho, P. M. (2009). 2–ANTBAL: an ant colony optimisation algorithm for balancing two–sided assembly lines. *Computers and Industrial Engineering*, *56*, 489–506.

Sparling, D., & Miltenburg, J. (1998). The mixed-model U-line balancing problem. *International Journal of Production Research, 36*(2), 485–501.

Stützle, T., & Dorigo, M. (1999). *ACO algorithms for the travelling salesman problem*. T.R./IRIDIA/99-3, Université Libre de Bruxelles, Belgium.

Talbi, E. G. (2002). A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics, 8,* 541–564.

Tapkan, P. , Ozbakır, L., & Baykasoğlu, A. (2012). Bees algorithm for constrained fuzzy multi–objective two–sided assembly line balancing problem. *Optimization Letters, 6*, 1039-1049.

Thomopoulos, N. T. (1967). Line balancing–sequencing for mixed–model assembly. *Management Science, 14*, B59–B75.

Thomopoulos, N. T. (1970). Mixed-model line balancing with smoothed station assignments. *Management Science, 16*, 593-603.

Vilarinho, P. M. & Simaria S. A. (2002). A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research, 40*(6), 1405–1420.

Vilarinho, P. M. & Simaria, A. S. (2006). ANTBAL: an ant colony optimization approach for balancing mixed model assembly lines with parallel workstations. *International Journal of Production Research, 44*, 291-303.

Wilhelm, W. (1999). A column-generation approach for the assembly system design problem with tool changes. *International Journal of Flexible Manufacturing Systems, 11*(2), 177–205.

Yagmahan, B. (2011). Mixed-model assembly line balancing using a multi-objective ant colony optimization approach. *Expert Systems with Applications, 38,* 12453-12461.

Yolmeh, A., & Kianfar, F. (2012). An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times. *Computers and Industrial Engineering, 62,* 936-945.

Zhao, X., Ohno, K. & Lau, H. S. (2004). A balancing problem for mixed-model assembly lines with a paced moving conveyor. *Naval Research Logistics, 51,* 446-464.