Balancing, Sequencing and Determining the Number and Length of Workstations in a Mixed Model Assembly Line

by

Fatemeh Mohebalizadehgashti

A Thesis

presented to

The University of Guelph

In partial fulfilment of requirements

for the degree of

Master of Applied Science

in

Engineering

Guelph, Ontario, Canada

ABSTRACT

BALANCING, SEQUENCING AND DETERMINING THE NUMBER AND LENGTH OF WORKSTATIONS IN A MIXED MODEL ASSEMBLY LINE

Fatemeh Mohebalizadehgashti **Advisor:**

University of Guelph, 2016 **Professor F.M. Defersha**

The single model assembly line is a traditional type of assembly line, which assembles only one product in a large quantity. On the other hand, the mixed model assembly line assembles different models of a product simultaneously. Therefore, it gives a chance to companies to retain the market by satisfying various demands of customers. Because of this advantage, companies are motivated to change their assembly line from the single model to the mixed model. Balancing and sequencing problems are two important challenges in the mixed model assembly line. There are a large number of studies that have focused on balancing and sequencing problems separately. However, in this thesis, we study balancing and sequencing problems of the mixed model assembly line simultaneously. A mixed integer linear programming model is proposed to solve these problems simultaneously when the assembly line has the continuous motion and when common tasks between different models of a product can be assigned to different workstations. Objectives in this thesis are minimizing the length of workstations, minimizing the stations cost, and minimizing the tasks duplication cost. A branch and bound algorithm is exploited to solve the model. Following that, the proposed model is extended to show that it can satisfy the assembly line with the synchronous configuration. At the next step, a hybrid genetic algorithm, which is a combination of the genetic algorithm and linear programming algorithm, is employed to solve the proposed model for large size problems. Finally, numerical examples are presented to show how

the proposed hybrid genetic algorithm solves the proposed model effectively.

Keywords: *Balancing; Sequencing; Mixed Model Assembly Line; Hybrid Genetic Algorithm*

Dedicated to my parents and my husband

ACKNOWLEDGEMENTS

I am deeply thankful to my supervisor, Dr. Fantahun M. Defersha, who guided me in a right direction with his comprehensive knowledge. I would never have been able to finish my thesis without his massive support, and critical advices. I feel a profound gratitude for his patience, valuable guidance, motivation and enthusiasm. I am also grateful to Dr. Soha Eid Moussa for the time she spent for my thesis.

I must express my very heartfelt gratitude to my brother, Hadi, for his strong support during my master period. I would not be able to achieve my master degree without his tremendous help.

Furthermore, I am grateful to my husband, Mehdi Shahini, for his emotional support, great encouragement, and enthusiasm.

My sincere thank is for my lovely parents for their endless love and support, and also continuous encouragement throughout my study. I could not live and study in Canada without having their emotional support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$a_{t,k}$	Binary variable which equal to 1 if task t of any model is assigned to workstation k
B	A large positive number
c	Index for a chromosome
$D_{m,t}$	Assembly time for task t of model m
d_m	Demand of model m in the MPS
F	Feasible
g	Generator counter
H	Number of iterations which populations are generated successfully without any improvement in the best fitness function value so far obtained
$H_{max}1$	Maximum number of H that leads to enter to the second phase if the previous Phase was equal to 1
$H_{max}2$	Maximum number of H in the second phase that leads to enter to the third phase if the previous Phase was equal to 2
$H_{max}3$	Maximum number of H in the third phase that leads to stop the third phase
Inf	Infeasible
$j_{m,t}$	Binary data which equals to 1 if $D_{m,t} \geq 0$, 0 otherwise
K	Total number of workstations
k	Workstation index
Lr	Launching rate of each model
M	Total number of models
m	Model index

$maxg$	Maximum number of generations
p	Population size
$p_{s,k}$	Start position of operator at sequence s in workstation k
$Phase$	An indicator number that takes number 1, 2, and 3 based on three defined phases
$Pre_{m,t}$	Set of immediate precedent tasks for task t of model m
S	Total number of sequences
s	Sequence index
SC	Station cost, fixed cost associated with each workstation
T	Total number of tasks
t	Task index
TC	Task duplication cost
v	Speed of conveyor
w_k	Length of workstation k
$x_{m,t,s,k}$	Binary variable which equals to 1 if task t of model m at sequence s is assigned to workstation k, 0 otherwise
$y_{m,s}$	Binary variable which equals to 1 if sequence s is assigned to model m
z_k	Binary variable which equals to 1 if workstation k is open, 0 otherwise

LIST OF ACRONYMS

ALB	Assembly Line Balancing
ACO	Ant Colony Optimization
BB	Branch and Bound
DFA	Design For Assembly
DP	Dynamic Programming
GA	Genetic Algorithm
GALB	General Assembly Line Balancing
GALBP	General Assembly Line Balancing Problem
GP	Goal Programming
GRASP	Greedy Randomized Adaptive Search Procedure
HGA	Hybrid Genetic Algorithm
IP	Integer Programming
LP	Linear Programming
MILP	Mixed-Integer Linear Programming
MALBP	Mixed Model Assembly line Balancing Problem
MMD	Multi/Mixed Model Deterministic
MMS	Multi/Mixed Model Stochastic
MMSAL	Mixed Model Synchronous Assembly Lines
MP	Maximal-Path technique
MPS	Minimum Part Set
MSP	Mixed Model Sequencing Problem
SA	Simulated Annealing
SALB	Simple Assembly Line Balancing

SALBP	Simple Assembly Line Balancing Problem
SMD	Single Model Deterministic
SMS	Single Model Stochastic
SP	Shortest-Path technique
TS	Tabu Search
UALBP	U-Shaped Assembly Line Balancing Problem

Chapter 1

Introduction

In the past decade, assembly lines have attracted more attention with the advance of technology and the growing competition between companies for retaining customers and market. Ghosh and Gagnon (1989), Kriengkorakot and Pianthong (2007) and Sivasankaran and Shahabudeen (2014) explained assembly line system as following: an assembly line includes a set of different workstations, which usually have been fixed along the conveyor belt by allocating the specific machines or operators. A base part is launched at the beginning of the line. The launched part moves from station to station along the conveyor, where different components are added or some operations are performed by machines or operators. The total amount of work is broken into the various tasks, which are allocated to the different workstations according to their precedence relationships. Specifically, precedence constraints help to determine the sequence of operations in workstations. Each operation has a specific time. Therefore, the total time of a set of operations, which is assigned to each workstation, defines the workstation time that should not be longer than the cycle time, which is the time interval between two released final products at the end of the assembly line. The cycle time is also known as the maximum available time for each workstation. The significant

problem of designing an assembly line is how to assign resources to different workstations in order to satisfy product requirements with minimum cost (Graves and Redfield, 1988). On the other hand, increasing efficiency is the main objective of assembly line design (Rekiek *et al.*, 2002b; Yaman, 2008). In this thesis, we develop a model and a solution procedure for designing an assembly line with the aim of solving balancing and sequencing problems simultaneously. In the following sections, some basic assembly line concepts, design considerations, and general information about assembly line balancing are discussed.

1.1. Basic assembly line concepts

Some fundamental concepts of an assembly line, which are based on the definition of an assembly line, are explained in this section. These concepts, which give a good overview about the assembly line, are as follows:

Work piece

A work piece is an unfinished product, which is made up of different components and also needs to be processed with various operations in order to form a final product (Torenli, 2009).

Sub-assembly

Sub-assembly is a complex part that needs to be assembled with different components before being added to the main work piece in the assembly line (Torenli, 2009).

Operation

Operations are various tasks that are performed on the work piece in order to produce a final product.

Operator

Operators are responsible for performing different operations on the work piece in workstations along the conveyor belt.

Workstation

Workstations are established along a conveyor belt with the aim of performing different operations on the work piece in the assembly line. They are equipped with materials, machines, and operators.

Operation time

Operation time is the time that an operation needs to be completed in a workstation (Scholl, 1999).

Workstation time

Workstation time is the total time of all operations that are performed in that workstation.

Cycle time

Cycle time is the time between two final outputs of the assembly line, which has inverse relationship with the number of workstations. Therefore, if cycle time increases, the number of workstations decreases and vice versa (Sivasankaran and Shahabudeen, 2014).

Idle time

Idle time is the time that work piece undergoes a waiting time until it moves to the next workstation. In this time, other operations are being completed on other work pieces in some workstations.

1.2. Design for assembly line

Design is an important basic step, which should be considered before starting production or product assembly in order to prevent extra costs in future steps like material selection, manufacturing and equipment selection (Abdullah *et al.*, 2003). The design of an assembly line has different objectives, which help companies to better exploit their resources, like human, machine, space, and money, to satisfy the customer demand. These objectives can be minimizing the number of work stations or minimizing the cycle time (Sivasankaran and Shahabudeen, 2014). The basic data which are required for design of the assembly line are as follows (Sivasankaran and Shahabudeen, 2014):

- Precedence relationships between operations, which show the sequence and priority of operations
- Task time
- Cycle time or number of workstations

Designers deal with some problems at the beginning of the design process (Chow, 1990): the first problem is lack of enough information. Specifically, there are a large number of unknowns at first. The second problem comes from uncertainty. This means that available information can change during the design. For example, the product demand is dependent on the market. Therefore, it is impossible to forecast an exact demand at the beginning of the design. The third

problem is variation. A large number of the assembly line parameters are in their average values, like mean process time or mean repair time. Hence, there is no deterministic system. Finally, the last problem is complexity, which is related to the different characteristics of the line and also communication between members of a design team. Specifically, there are a large number of design alternatives for making decisions about various factors, like material handling, inventory, work pieces, labour, and information. Making a proper decision to select a proper combination of these factors is very important. What is more, communication between people in a design team, who come from different departments like production, quality control, and marketing, is another problem of the design, which makes it complicated. Abdullah *et al.* (2003) divided design for assembly (DFA) into two groups: qualitative and quantitative methods. Qualitative methods provide guidelines with examples for designers (Andreasen *et al.*, 1983). These methods are general and they do not provide a specific method for designers (Abdullah *et al.*, 2003). On the other hand, quantitative methods focus on some specifications of operation like time and cost (Miyakawa and Ohashi, 1986; Poli and Fenoglio, 1987). In addition, physical disassembly of the product is considered in quantitative methods in order to improve the structure of a product for easier assembly (Abdullah *et al.*, 2003).

1.3. Assembly line components

An assembly line has different components that should be considered by designers in order to increase efficiency of the line and also to meet the customer demand. One division of components can be as follows:

- Operators
- Operations

- Precedence graph
- Workstations
- Equipment
- Material handling
- Buffer
- Feeder line
- Pallet
- Fixture
- Line layout
- Inspection

Operators

Operators are responsible for performing different operations on the work piece in the workstations. Therefore, designers should consider various factors when they design tasks for operators, namely, capacity of operators, skill level (Johnson, 1983), operator physical demand and fatigue (Carnahan *et al.*, 2001), and ergonomic (Chow, 1990).

Operations

Operations are assigned to the different workstations based on some rules. For example, the precedence relationships between operations should not be violated. Also, some operations, which are called compatible operations, must be assigned to the same workstation. Because they should use the same resources. However,

incompatible operations are not allowed to be assigned to the same workstation (Boysen *et al.*, 2007). Each operation has a specific time, which can be different based on the operation nature, operator skill, and machine reliability (Rekiek *et al.*, 2002a).

Precedence graph

The precedence graph shows the basic information about operations like operation names, operation times, and forward and backward path (Boysen *et al.*, 2007). Specifically, it provides information about the sequence and priority of various operations that need to be performed on the work piece to reach a final product.

workstations

All operations are performed in workstations by operators. Different factors should be considered by designers in each work station, namely, the number of operators, type of equipment, length of workstation, and workstation time.

Equipment

Equipment is one of the most important components in the design of an assembly line. Equipment is selected based on the different factors such as requirement of operations in each workstation (Becker and Scholl, 2006), and investment cost (Boysen *et al.*, 2007). The investment cost will be reduced if some operations, which need the same equipment, are assigned to the same work station. Because this matter decreases the number of installations (Boysen *et al.*, 2007).

Material handling

Material handling is to move materials, for example, from one workstation to another or from warehouse to the assembly line site, in front of the workstation

that needs to be assembled. Although material handling does not add any value to the final product, it should be considered in order to prevent some waste in the system such as delay time (Chow, 1990).

Buffer

Buffers are used between some workstations in order to store work pieces temporarily. If a buffer is necessary in the line, some factors should be considered by designers such as location and capacity of the buffer (Groover, 2007)

Feeder line

Feeder lines are used to feed the main assembly line with subassemblies, which will be added to the main work piece. Cycle time of the feeder line is the same as the cycle time of the main assembly line. Therefore, it is better to determine the cycle time of the main assembly line at the first and following that, applying that cycle time for the feeder line (Boysen *et al.*, 2007).

Pallet

Pallets are used for moving materials. Hence, they are part of the material handling subject. Using pallet in an assembly line has some advantages (Groover, 2007): they help to transfer multiple items instead of individual items. Furthermore, they prevent product damage. Also, using pallets can decrease loading and unloading times. Some factors should be considered by designers for using pallets in the assembly line like size of pallets and safety (Groover, 2007).

Fixture

Fixture is a tool that is applied to support and hold the work pieces during operations.

Line layout

Line layout is a physical arrangement of different elements of assembly line. Most of components of assembly line should be considered in the layout. An efficient layout can facilitate the production flow.

Inspection

Inspection is carried out on the work pieces in the different stages of the assembly process in order to detect problems. Then, the root of problem is determined with the aim of reducing the faults and following that extra cost.

1.4. Line balancing

Line balancing is one of the important subjects in the assembly line design. Specifically, line balancing is a method to increase the efficiency of the assembly line with the aim of reaching the highest production rate and/or shortest line. It consists of assigning tasks to the different workstations in such a way that the precedence of tasks as well as other restrictions are satisfied by using the various algorithms and methods (Ghosh and Gagnon, 1989; Erel and Sarin, 1998; Becker and Scholl, 2006; Kriengkorakot and Pianthong, 2007; Boysen *et al.*, 2008). Line balancing is performed to reach different objectives, namely, minimizing the number of workstations and minimizing the cycle time. Although there are a large number of studies, which have focused on the assembly line balancing problem, only a few companies have utilized presented methods (Ghosh and Gagnon, 1989; Boysen *et al.*, 2007, 2008). The gap between the academic studies and the real-world manufacturing environment is a result of several reasons (Ghosh and Gagnon, 1989): the first reason is that companies do not know how to use algorithms. The second reason is that there is no consistency between studies and real problems in assembly lines. Finally, the difficulty of techniques and algorithms is

another problem, which causes line balancing methods not to be used by companies. Line balancing research has focused on the simple assembly line balancing problem ($SALBP$) and the general assembly line balancing problem ($GALBP$). The simple assembly line balancing has following assumptions (Baybars, 1986; Scholl, 1999; Scholl and Becker, 2006):

- It is used for the mass-production of one product with a specific known production process.
- Precedence constraints are the main restrictions.
- Workstations are equipped with an equal level of machines and labour.
- It has a constant cycle time.
- Operation times are deterministic.
- The line has one-sided stations with the serial progression.
- It is not possible to separate a task between two or more workstations.

The precedence diagram is a key tool of the simple assembly line balancing (Boysen *et al.*, 2007). However, some of the features and assumptions of the simple assembly line balancing are removed or edited in the general assembly line balancing (Ghosh and Gagnon, 1989; Becker and Scholl, 2006; Boysen *et al.*, 2007; Scholl and Becker, 2006). This type of line balancing can help to solve more real-world problems (Becker and Scholl, 2006).

1.5. Organization of the thesis

The outline of this thesis is as follows: In Chapter 2, a review of the literature on the classification of the assembly line as well as the classification of the assembly line balancing problems are presented. In addition, Chapter 2 reviews earlier

studies on objectives in assembly line balancing as well as different techniques for solving line balancing problems. Furthermore, the mixed model assembly line is reviewed in terms of balancing and sequencing at the end of Chapter 2. In Chapter 3, a mixed integer-linear programming (MILP) model is presented, which is based on the studies conducted by Bukchin and Rabinowitch (2006) and Mosadegh *et al.* (2012a). In addition, the proposed model is extended in Chapter 3. In Chapter 4, a detailed solution procedure is depicted for the proposed model based on the hybrid genetic algorithm (HGA). Numerical examples are introduced in Chapter 5. Finally, conclusions and future research directions are presented in Chapter 6.

Chapter 2

Literature Review

2.1. Introduction

In this chapter, a classification of the assembly line is discussed based on different studies conducted in the past. Following that, a review of the literature on the classification of assembly line balancing problems, objectives in assembly line balancing, and different methodological techniques for solving assembly line balancing problems are presented. Then, the mixed model assembly line is reviewed in terms of balancing and sequencing problems.

2.2. Classification of the assembly line

Different classifications have been defined for the assembly line system, which have explained various specifications of an assembly line. The following explanations summarize the classification of the assembly line based on the studies conducted by Becker and Scholl (2006), Boysen *et al.* (2007), and Boysen *et al.* (2008). They classified the assembly line according to different factors, which are the number of models, type of line control, level of automation, task duration, and line layout.

2.2.1. Number of models

2.2.1.1 Single model assembly line

In the single model, which is known as the traditional model of the assembly line, only one type of product is produced in the assembly line. Figure 2.1 shows the single model assembly line.

Figure 2.1: Single model assembly line

2.2.1.2 Mixed model assembly line

In the mixed model, different models of a product, which have similar operations, are assembled in the same assembly line. Therefore, the set up time can be ignored between models. Mixed sequence of models is emphasized in the mixed model assembly line as illustrated in Figure 2.2.

Figure 2.2: Mixed model assembly line

2.2.1.3 Multi model assembly line

In the multi model, different products are assembled in batches. Therefore, one model or similar models of a product are assembled in each batch. The set up time is not ignored in the multi model assembly line because different models are produced with different machines or operators. Figure 2.3 shows the multi model assembly line.

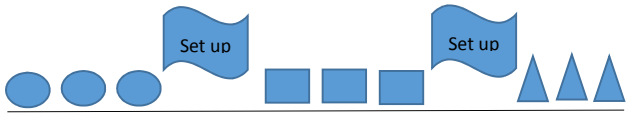


Figure 2.3: Multi model assembly line

2.2.2. Type of line control

2.2.2.1 Paced assembly line

In the paced assembly line, the same limited time is assigned to all workstations as the cycle time. Therefore, workstations have a common cycle time and their time does not exceed the defined cycle time. Hence, the paced line has a fixed production rate. In addition, the assembly line can have continuous motion or even the intermittent configuration when the line is paced.

2.2.2.2 Unpaced assembly line

In an unpaced assembly line, workstations do not have a time restriction for completing operations and they are not restricted with the same cycle time. Specifically, each workstation has its own time to complete operations on the work piece. The unpaced line is divided into two groups (Boysen *et al.*, 2007, 2008): asynchronous line and synchronous line. In the asynchronous line, work pieces can move to the next workstation whenever required operations are completed. Therefore, workstations have different times and speeds. This characteristic leads to the use of a buffer in order to minimize the waiting time for receiving work pieces from the previous work station (starvation) or to prevent blocking the next work station. In contrast, in the synchronous line, all work stations wait for the slowest work station to complete its required operations. This type of unpaced assembly line can behave as a paced line with intermittent motion if task times

are deterministic. Hence, the cycle time is the time of the slowest workstation.

2.2.3. Level of automation

2.2.3.1 Manual line

In the manual line, human operators are responsible for performing operations. This type of assembly line is used in different situations. For example, if some products are very sensitive and fragile, the manual line is the best choice to produce them. Furthermore, some operations need more accuracy. Therefore, operators can perform them better than robots (Bi and Zhang, 2001). Using a manual line has some advantages. For example, operators can support each other in adjacent workstations when they are dealing with overload in workstations. This support can easily occur in the U-shaped line (Aase *et al.*, 2004). However, the manual line has some demerits. For example, operation time can be stochastic in this type of assembly line. Therefore, workstation time can have some deviations (Tempelmeier, 2003). In addition, quality level can be reduced in the manual line if workstations have a high workload and operators work with a high speed.

2.2.3.2 Automated line

In an automated line, operations are performed by robots. Using this type of assembly line has some merits. For example, the operation time is deterministic. Therefore, there is no deviation in the workstation time (Becker and Scholl, 2006; Boysen *et al.*, 2008). In addition, robots can work in a dangerous environment, where operators can not work (Boysen *et al.*, 2008). However, the automated line needs a high investment cost (Boysen *et al.*, 2007, 2008).

2.2.4. Task duration

2.2.4.1 Deterministic

In this case, all operation times are constant or they have small deviations. Therefore, workstation times are known and stable (Becker and Scholl, 2006; Boysen *et al.*, 2007).

2.2.4.2 Stochastic

Task time is stochastic if the operation time deviates from what it should be. Therefore, the operation time is probabilistic (Becker and Scholl, 2006; Boysen *et al.*, 2007).

2.2.4.3 Dynamic

Dynamic task time occurs if the operation time has a dynamic variation because of individual experience of a operator, who deal with a new assembly line or a new product in the assembly line. Specifically, an operator needs time to adjust himself with these new situations, which learning effects can impact on the operation time and can reduce it (Simaria, 2006; Boysen *et al.*, 2007).

2.2.4.4 Dependent

Operation time is dependent on different factors, which can increase times. One of these factors can be the sequence of operations. For example, when two consecutive operations are performed in one workstation, extra time is needed for preparing a workstation to perform the second operation such as changing tools (Wilhelm, 1999). Hence, operation time is not fixed here.

2.2.5. Based on the line layout

2.2.5.1 Serial lines

Serial lines are the traditional type of assembly line, in which the layout of workstations is straight along the conveyor belt.

2.2.5.2 U-shaped lines

In U-shaped lines, the layout of assembly lines is like U form. Therefore, operators can have an opportunity to work in different workstations (Becker and Scholl, 2006; Boysen *et al.*, 2007).

2.2.5.3 Two-sided lines

In two-sided lines, two serial lines are arranged in parallel in such a way that two opposite workstations can perform operations simultaneously on the two different sides of the work piece (Bartholdi, 1993). These lines are proper for some large products like car.

2.2.5.4 Parallel lines

Parallel lines are installed when multiple products are assembled in the system. Specifically, each line is used for each product or for a family of similar products (Becker and Scholl, 2006).

2.2.5.5 Parallel workstations

In parallel workstations, work pieces are distributed in several workstations, which have been arranged in parallel. Then, different operators perform identical operations on work pieces (Becker and Scholl, 2006).

2.3. Classification of assembly line balancing problems

Different categories have been proposed in the past in order to classify assembly line balancing problems. For example, Kriengkorakot and Pianthong (2007) classified assembly line balancing problems based on the literature into two categories. The first category was described by Ghosh and Gagnon (1989), who classified assembly line balancing problems in literature into two main models: single model and multi/mixed model. Then, they considered two types of task time and divided each model into deterministic and stochastic time. Finally, they grouped the two task times into the simple and general assembly line. Figure 2.4 illustrates the related categorization.

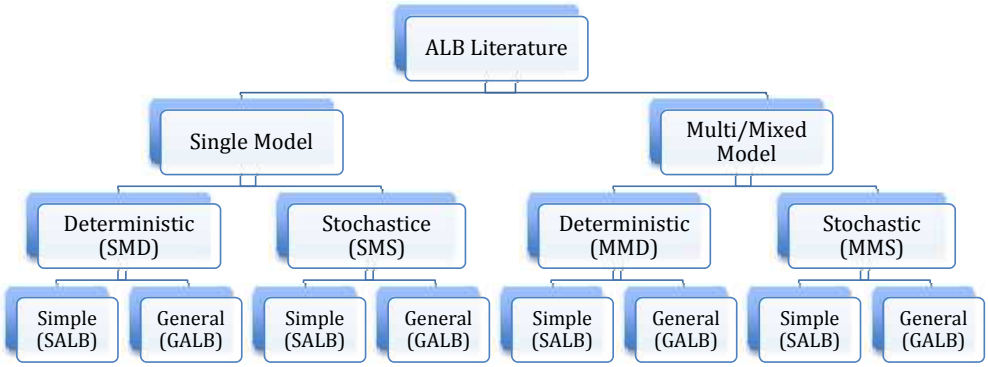


Figure 2.4: Classification of assembly line balancing based on Ghosh and Gagnon (1989)

The Single Model Deterministic (SMD) refers to a model with one product in the assembly line, which has deterministic operation time. It is the simplest type of assembly line balancing ($SALB$), which will be converted to the general

assembly line balancing ($GALB$) if some constraints are added or some assumptions change. Introducing zoning constraints or using parallel workstations in the assembly line are two examples that convert $SALB$ to $GALB$. The Single Model Stochastic (SMS) refers to a model with one product in the assembly line, which has a probabilistic operation time. This type of model is used for the manual assembly line. The Multi/Mixed Model Deterministic (MMD) refers to the deterministic task time in the multi or mixed model assembly line. Various factors are involved in the multi/mixed model such as launching rate and model sequences, which are not considered in the single model. The Multi/Mixed Model Stochastic (MMS) refers to the variable operation time in the multi/mixed model assembly line. All of subjects which are associated with the SMS model are present in the MMS model, but they are more complicated compared with the SMS model.

The second category was described by Scholl and Becker (2006) and Becker and Scholl (2006). They proposed different classification compared with Ghosh and Gagnon (1989) and classified assembly line balancing problems into two main groups: the simple assembly line balancing problem ($SALBP$) and the general assembly line balancing problem ($GALBP$). Then, they considered various objectives and extended their divisions to $SALBP-1$, $SALBP-2$, $SALBP-E$, and $SALBP-F$ for the simple assembly line balancing problem, and $MALBP/MSP$ and $UALBP$ for the general assembly line balancing problem. Figure 2.5, demonstrates the related classification. The Simple Assembly Line Balancing ($SALB$) is appropriate for a single product in a serial line with a straight conveyor belt. Precedence constraint is the only restriction in the $SALB$. The objective of $SALBP-1$ (Type 1) is minimizing the number of workstations in the assembly line when the cycle time is constant while the objective of $SALBP-2$ (Type 2) is minimizing the cycle time when the number of workstations is known and given. On the other hand, $SALBP-E$ (Type E) focuses on maximizing the line efficiency, which leads to minimizing the cycle time and the number of workstations

simultaneously, and $SALBP-F$ (Type F) focuses on finding a feasible solution for the line balancing problem while the number of workstations and cycle time are known and given.

In the general assembly line balancing ($GALB$), some features and assumptions of the simple assembly line balancing ($SALB$) are removed or edited. In this category, the Mixed Model Assembly Line Balancing Problem ($MALBP$), focuses on finding the number of workstations, cycle time, and line balancing techniques to optimize the cost and capacity (Scholl, 1999). On the other hand, the Mixed Model Sequencing Problem (MSP) focuses on detecting a sequence for different models of a product in order to reach different objectives such as minimizing line inefficiency (Scholl *et al.*, 1998). The U-shaped Assembly Line Balancing Problem ($UALBP$) is appropriate for a single model in the U form assembly line. In this case, operators can work in several workstations, which are located on both sides of the U. Therefore, precedence constraints can be modified. Different problems are identified in the U-shaped assembly line balancing problem in comparison with the simple assembly line balancing problem (Miltenburg and Wijngaard, 1994; Erel *et al.*, 2001).

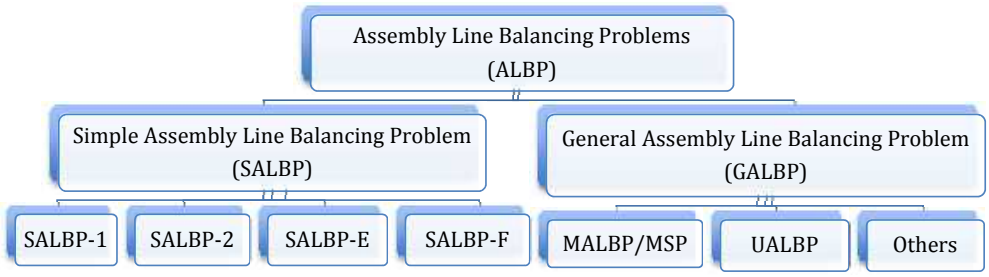


Figure 2.5: Classification of assembly line balancing based on Scholl and Becker (2006) and Becker and Scholl (2006)

However, Sivasankaran and Shahabudeen (2014) defined another classification for the assembly line balancing, which was similar to Ghosh and Gagnon (1989) classification. Figure 2.6 shows their proposed categorization. The big difference between the last classification and Ghosh and Gagnon (1989) classification is the layout of workstations, which has not been considered in Ghosh and Gagnon (1989) classification. This layout is divided into two groups in Sivasankaran and Shahabudeen (2014) classification: the straight line and the U-shaped line. In the straight line, workstations are arranged beside the straight conveyor belt, where different components are added to the launched part step by step. In contrast, in the U-shaped line, the layout of workstations is U form. This type of assembly line allows operators to work in more than one workstation.

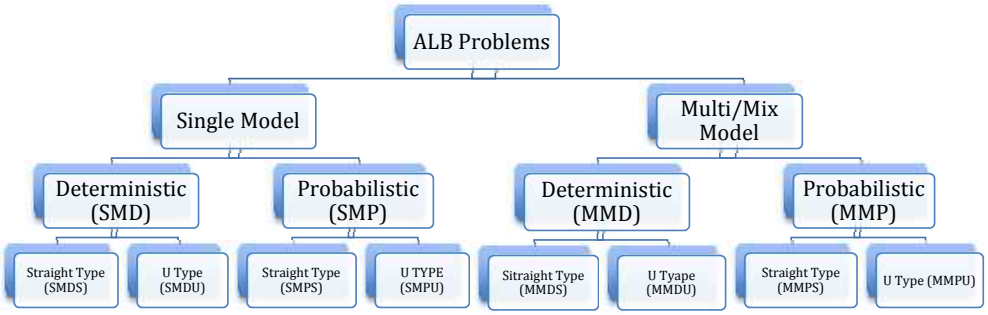


Figure 2.6: Classification of assembly line balancing based on Sivasankaran and Shahabudeen (2014)

2.4. Objectives in assembly line balancing

Objectives in the assembly line balancing are divided into a single objective and multiple objectives. In the single objective group, one objective is determined to optimize the assembly line and also to balance it. However, when the goal is to optimize several objectives at the same time, the optimization is dealing with

multiple objectives in the assembly line (Malakooti and Kumar, 1996). Kriengkorakot and Pianthong (2007) divided the objective criteria in the literature into two groups based on studies conducted by Ghosh and Gagnon (1989) and Scholl (1999).

Ghosh and Gagnon (1989) divided the objective criteria of ALB in the literature into two main categories, which were Technical and Economic objectives. Table 2.1 shows the related classification. Each number inside the table shows the total number of articles which used a specific objective for each assembly line balancing problem. They defined technical objectives with seven restrictions which were correlated to the throughput or operational efficiency. Minimizing the number of workstations was the most common objective, which was widely used in different types of assembly line. They also described economic objectives with the six criteria, which were associated with the assembly line operating cost or profitability measures. As the Table 2.1 highlights, minimizing the labour cost or the labour idleness cost was the main focus of a large number of studies. However, few studies focused on minimizing other costs such as product incompletions (Kottas and Lau, 1973), penalty costs (Dar-El and Cucuy, 1977), and inventory and set up cost (Caruso, 1965).

On the other hand, Scholl (1999) considered two objective criteria as principle categories; capacity oriented goals and cost oriented goals. Maximizing the capacity utilization of the assembly line is one of the main goals of the assembly line balancing. This objective is measured by the line efficiency, which relies on the cycle time and the number of workstations, if the line deals with the production of the single model that has the constant operation time. Therefore, minimizing the number of workstations for a given cycle time, minimizing the cycle time for a given number of workstations, minimizing the balance delay time, and the balance delay over all workstations are the most important objectives,

which are considered to reach the maximum line efficiency. In addition, minimizing the total cost is one of the important goals of the assembly line balancing, which deals with a large number of costs in a short time and also in a long time such as costs of machinery and tools, wage costs, material costs, operation costs, set up cost, and investment cost. The number of workstations and the cycle time are two factors that affect the total cost.

Table 2.1: Assembly line balancing objective criteria based on Ghosh and Gagnon (1989)

	Frequency of use				
Type of objective	SMD	SMS	MMD	MMS	Total
Technical					
Minimize the no. of workstations	16	2	2	1	21
Minimize the cycle time	13	1	2	0	16
Minimize the total idle time	9	0	3	0	12
Minimize the balance delay	2	0	1	0	3
Minimize the overall facility or line length	0	0	2	0	2
Minimize the throughput time	0	0	1	0	1
Minimize the probability that one or more work stations will exceed the cycle time	0	1	1	1	3
Total	40	4	12	2	58
Economic					
Minimize the combined cost of labour, workstations and product incompleteness	0	3	0	1	4
Minimize the labour cost	3	1	0	0	4
Minimize the total penalty cost for a number of inefficiencies	0	0	2	0	2
Minimize the inventory, set up and idle time cost	0	0	1	0	1
Minimize the total in-process inventory cost	0	1	0	0	1
Maximize the net profit	1	0	0	1	1
Total	4	5	3	2	13

However, Boysen *et al.* (2007), defined following objectives:

- Minimizing the number of workstations for a given cycle time.
- Minimizing the cycle time or maximizing the production rate for a given number of stations.

- Maximizing the line efficiency.
- Minimizing the cost of the assembly line.
- Maximizing the profit, which is difference between the revenue and the cost.
- Smoothing the workstation times within a workstation and between workstations.
- Minimizing or maximizing some composite score related to the one or more features describing bottleneck aspects or further measures of efficiency.
- Finding a feasible solution

As is obvious, some of the above objectives are similar to defined goals by Scholl (1999), and also Ghosh and Gagnon (1989).

2.5. Methodological techniques for solving line balancing problems

Different solution methodologies have been explored in order to reach the optimal solutions. Ghosh and Gagnon (1989), organized these methods into two groups, which were exact and inexact methods, and analysed their use in the different models of the assembly line balancing. Table 2.2 demonstrates the related methods. Each number inside the table shows the total number of articles, which used a specific method for each assembly line balancing problem. Table 2.2 shows that researchers focused on the SMD model and used various methods to reach different objectives. In addition, Branch and Bound method (BB) was the most popular method in exact methods group, while priority-ranking method was the most common method in the second group.

However, Sivasankaran and Shahabudeen (2014), organized methods with consideration of different factors, which are various types of assembly line balancing (type 1 and type 2), different assembly line balancing problems ($SMD-S$, $SMD-U$, $SMP-S$, $SMP-U$, $MMD-S$, $MMD-U$, $MMP-S$, and $MMP-U$), various objectives, computational effort levels (high, medium, and low), and also different types of solution optimality (optimal, near optimal, very near optimal, and far less from optimality). A summary of their research is demonstrated in Table 2.3. Each number inside the table shows the total number of articles, which have used a specific method for each assembly line balancing problem. As it is clear from Table 2.3, the $SMDS$ has the maximum number of articles, which apply different methods to solve the problem. Also, few articles have focused on the U-shaped assembly line balancing problem compared with the straight type assembly line balancing. In addition, it is obvious that the genetic algorithm has the maximum number of articles compared with other methods. This means that researchers tended to utilize the genetic algorithm more than other methods to solve problems. Following that, heuristic methods, mathematical models, and optimization algorithms were more popular in the research. However, memetic algorithm, bee algorithms, critical path method, and particle swarm optimization algorithm were the least common methods in the research with smallest number of articles.

Table 2.2: ALB methodological techniques based on Ghosh and Gagnon (1989)

	Frequency of use				
ALB Techniques	*SMD*	*SMS*	*MMD*	*MMS*	Total
Exact Methods					
Linear Programming (*LP*)	1	0	0	0	1
Integer Programming (*IP*)	7	0	1	0	8
Dynamic Programming (*DP*)	4	2	0	0	6
Goal Programming (*GP*)	1	0	0	0	1
Shortest-path technique (*SP*)	2	0	2	0	4
Maximal-path technique (*MP*)	1	0	0	0	1
Branch and bound (*BB*)	11	1	0	1	13
Total					34
Inexact Methods					
Priority ranking and assignment	10	5	7	2	24
Tree search (heuristic *BB*)	8	1	0	0	9
Trade and transfer	1	2	1	0	4
Random sampling	3	1	0	0	4
Other heuristic methods consist of task grouping, approximation technique	5	3	1	2	11
Total					52

Table 2.3: ALB methodological techniques based on Sivasankaran and Shahabudeen (2014)

	Frequency of use								
Methodological Techniques	*SMD S*	*SMD U*	*SMP S*	*SMP U*	*MMD S*	*MMD U*	*MMP S*	*MMP U*	Total
Mathematical models	7	2	3	0	4	0	0	1	17
Petri net algorithms	2	0	0	0	0	0	0	0	2
Heuristics	13	2	3	1	0	2	0	0	21
Genetic algorithms	14	0	4	1	6	2	1	1	29
Simulated annealing	5	1	2	0	2	0	1	0	11
Tabu search algorithms	4	0	0	1	1	0	0	0	6
Ant colony	7	3	0	0	2	0	0	1	14
Shortest path	1	1	1	0	0	0	0	0	3
Memetic algorithm	1	0	0	0	0	0	0	0	1
Bee algorithms	1	0	0	0	0	0	0	0	1
Critical path method	0	1	0	0	0	0	0	0	1
Imperialistic competitive algorithm	0	1	0	1	0	0	0	0	2
Particle swarm optimization algorithm	0	0	1	0	0	0	0	0	1
Total	55	11	14	4	15	4	3	3	

2.6. Mixed model assembly line

The mixed model assembly line is widely used in companies which have customers with different demands because this type of assembly line gives a chance to companies for producing different models of one product in an assembly line simultaneously. This means that workstations in the mixed model are flexible enough to perform operations on different models. Differences of models in the mixed model come from various factors such as size and colour diversity, applied materials or even equipment. Therefore, various operations, operation times and/or precedence relations are required to produce them (Becker and Scholl, 2006). Also, each model has a specific precedence diagram that can be integrated to reach a single precedence diagram. The combined precedence diagram was used first time for balancing the mixed model assembly line by Thomopoulos (1967). This version of precedence diagram has some advantages. For example, every repetition

of an operation is performed by the identical workstation. This matter leads to diminish training costs (Van Zante-de Fokkert and de Kok, 1997). Cycle time is another subject, which is very important in the mixed model assembly line. Each model can have its own cycle time in the assembly line but it is usually calculated based on the average cycle time for all models (Boysen *et al.*, 2009b). The precedence diagram and the cycle time are two main restrictions that must be satisfied in balancing of the mixed model assembly line. Several subjects should be considered for design of the mixed model assembly line, namely, line balancing (Manavizadeh *et al.*, 2012), layout design (Ho, 2005), and model sequencing (Boysen *et al.*, 2009b). However, the main problems for the planners of the mixed model assembly line are as follows (McMullen and Frazier, 2000):

- How to balance the assembly line?
- How to determine the optimum launch sequence?

These two problems are discussed in the following sections.

2.6.1. Mixed model assembly line balancing

The mixed model assembly line balancing includes assigning tasks to different workstations with consideration of various constraints in order to minimize cost and to satisfy demands of the products (Simaria and Vilarinho, 2004). The objective of the mixed model assembly line balancing is similar to the single model assembly line balancing, which is assigning tasks to workstations as evenly as possible (Groover, 2007). Different constraints should be considered when tasks are assigned to workstations. Three simple but important constraints are explained as follows: the precedence constraint, the cycle time constraint, and restriction of assigning tasks to one workstation. The first constraint is precedence relationships. Specifically, precedence constraints with defining predecessors and successors for

each task can limit alternative assignments of tasks to the workstations. The second constraint is the cycle time, which can determine the total task times that are allocated to each workstation. Therefore, this leads to limit the number of tasks that are assigned to each workstation. Finally, the last constraint is to assign each task to only one workstation. Simaria (2006) proposed different constraints for assignment of tasks to workstations as follows: zoning constraints, workstation constraints, position constraints, and operator constraints. The first group is related to assign the compatible (incompatible) tasks to the same (different) workstations. Compatible tasks are those, which can use the same equipment or are performed in the same conditions like temperature. However, incompatible tasks are not assigned to the same workstation because of several reasons such as safety requirements or capacity restriction of equipment. The second defined group by Simaria (2006) is workstation restrictions. Specifically, some workstations have special equipment, which are not available in other workstations. Therefore, tasks that need that equipment are assigned to those workstations. The third group is position constraints that limit assignment of tasks to the different positions. These constraints make easy to produce large products that have a fixed position by grouping tasks based on their positions in a two-sided assembly line. The last group is operator constraints. Some complicated tasks require operators with a high skill level. Therefore, qualified operators are selected in order to perform complex tasks. However, Boysen *et al.* (2007) proposed different constraints for assignment of tasks to workstations. These constraints are as follows:

- Zoning constraints
- Cumulative restrictions of task-station-assignment, which consider the cumulative value of different factors for task assignment such as space at a workstation and work content.

- Fixed task assignment to a particular workstation because of different restrictions, namely, using a special equipment that can not be moved to other workstations
- Assigning task to a certain type of workstation such as those tasks that need to be performed by a subset of machines (Boysen *et al.*, 2008), and vice versa
- Minimum and maximum restriction between tasks.

Assignment of tasks to workstations is characterized by two types of variability in the mixed model assembly line (Bukchin, 1998): model variability and station variability. The first variability is related to a certain model with the variability of the assembly times on the different stations, while the second one is related to the different models with variability of the assembly times at a specific station. These types of variability can cause blockage and starvation and following that, high idle time within stations, which totally lead to have the low throughput. Equal division of total assembly time of each model between workstations as well as having equal assembly times for different models can reduce these types of variability. Mixed model assembly line balancing problems are divided into two categories (Scholl, 1999): $MALBP-I$, which focuses on minimizing the number of workstations for a given cycle time, and $MALBP-II$, which focuses on minimizing the cycle time for a given number of workstations. $MALBP-I$ is frequently used when demand is foreseeable and the goal is to design a new assembly line, while $MALBP-II$ focuses on maximizing the production rate of an existing assembly line. Both of $MALBP-I$ and $MALBP-II$ are NP- hard (Bukchin and Rabinowitch, 2006). The mixed model assembly line balancing problem has the following specific characteristics (Simaria, 2006):

- Different models, which have some similarities, of a product are assembled in the assembly line at the same time.

- The demand of each model is known in a planning horizon.
- The cycle time of the line is known.
- Each model has a specific precedence diagram, which can be combined in order to have one precedence diagram.
- Each task has a specific time that may vary in different models.
- Workstations are flexible to perform various tasks of different models.

Different studies have been conducted in the past about mixed model assembly line balancing. For example, Simaria and Vilarinho (2004) presented a mathematical model to solve the mixed model assembly line balancing problem by using a genetic algorithm. Their objectives were minimizing the cycle time and also balancing the workload within each workstation. Bukchin and Rabinowitch (2006) used a branch and bound algorithm to solve the mixed model assembly line balancing problem with the aim of minimizing workstation costs and task cost. They allowed common tasks of different models to be assigned to different workstations. Akpinar and Bayhan (2011) proposed a hybrid algorithm containing of genetic algorithm, kilbridge and wester heuristic, Phase-I of Moodie and Young method, and ranked positional weight technique to solve a mixed model assembly line balancing problem. They focused on three objectives in their study: minimizing the number of workstations, maximizing the workload smoothness between workstations, and maximizing the workload smoothness within workstations. Manavizadeh *et al.* (2012) introduced a multi objective genetic algorithm to solve a mixed model assembly line balancing. The objectives were minimizing the cycle time and minimizing the number of workstations. Tiacci (2015) focused on solving the mixed model assembly line balancing problem and also buffer allocation problem simultaneously in such a way that task times are stochastic and using parallel workstations are allowed. They applied simulation techniques

with a genetic algorithm to solve these problems. Ramezanian and Ezzatpanah (2015) focused on solving multi-objective mixed model assembly line balancing problem as well as worker assignment problem. The objectives were minimizing the cycle time as well as minimizing total operating costs simultaneously. A goal programming approach was used to solve these problems.

2.6.2. Mixed model assembly line sequencing

A large number of studies have been conducted in the past about the mixed model assembly line sequencing, which show the importance of this subject. For example, Yano and Rachamadugu (1991) minimized the total utility work by solving models sequence problem in the mixed model assembly line. Kim and Jeong (2007) minimized unfinished work within workstations by solving a sequencing problem in the mixed model assembly line with a sequence-dependent set-up. Fattahi and Salehi (2009) proposed a heuristic model to minimize the total utility work and idle times. They solved sequencing problem of the mixed model assembly line with a variable launching interval between products. Salehi *et al.* (2013) proposed a meta-heuristic model to solve the sequencing problem in the mixed model assembly line. Three objectives were considered in their study: minimising the total idle cost, minimizing the total production rate variation cost, and minimizing the total set-up cost. Different objectives are exploited by researchers for solving sequencing problem in the mixed model assembly line, which can be explained namely through the following classification (Akgunduz and Tunali, 2011):

- Keeping a constant rate of the part usage, which has a direct relationship with the demand rate of actual production.
- Minimizing variation of production rates, which helps to have a constant rate of part usage (Mansouri, 2005).

- Minimizing work overload, which leads to minimize the operation time or even workstation border. The work overload can be compensated by additional utility workers and it can be avoided by determining a sequence of models (Boysen *et al.*, 2009b).
- Minimizing the set-up cost/time.
- Minimizing line length, which leads to minimize the investment cost (Boysen *et al.*, 2009b).
- Levelling workloads, which leads to have a balanced workload in each work station (Ding *et al.*, 2006).
- Minimizing throughput time, which is the time interval between the launching of the first work piece and the finishing of the last work piece.
- Minimizing the duration of line stoppages, which refers to the time when a line is stopped. Therefore, no work pieces can be completed during this time. Hence, this objective helps to minimize the cost for lost sales (Boysen *et al.*, 2009b).

One of the important concerns for solving the sequencing problem in the mixed model assembly line is launching discipline of models, which is defined as a time interval of launching the base parts into the beginning of the assembly line (Groover, 2007; Boysen *et al.*, 2009b). This time interval is constant and equal to the cycle time in a single model assembly line. However, that is complicated in the mixed model assembly line because of having different models of a product in the same assembly line (Groover, 2007). The launching discipline was divided into two groups (Wester and Kilbridge, 1964): fixed rate launching and variable rate launching. Base parts are launched in the constant time interval, which is equal to the cycle time, when the assembly line is dealt with the fixed rate launching (Wester and Kilbridge, 1964). This time interval relies on the product mix and

production rates of the models. Selecting the sequence of models is an important subject in the fixed rate launching which can avoid station congestion and/or idle time in the assembly line (Groover, 2007). However, the time interval is based on the cycle time of the current base part in the variable rate launching. Sequencing of models is not important in the variable rate launching. Therefore, models can be launched in any sequence desired without causing idle time or congestion at workstations (Groover, 2007).

Different methods have been used for solving the sequencing problem in the mixed model assembly line, such as heuristic (McMullen and Frazier, 2000; Mansouri, 2005), branch and bound algorithm (Bard *et al.*, 1994; Drexl *et al.*, 2006), integer programming (Dar-El and Cucuy, 1977; Drexl and Kimms, 2001), dynamic programming (Yano and Rachamadugu, 1991), tabu search (McMullen, 1998), simulated annealing (McMullen and Frazier, 2000; Boysen *et al.*, 2009a), and genetic algorithm (McMullen *et al.*, 2000; Kim *et al.*, 2000; Mansouri, 2005; Kim *et al.*, 2006; Akgunduz and Tunali, 2010). The genetic algorithm was the most common method, which has attracted more researchers for solving the sequencing problem.

Chapter 3

Mathematical Model

3.1. Problem definition

In the mixed model assembly line, some differences are considered between several products, which have a common basis and are assembled in one assembly line. Two important problems of the mixed model assembly line are line balancing and models sequencing. Line balancing is how to assign tasks to different workstations in the line while models sequencing is how to select the sequence of different models of a product. These two problems have been studied together in a hierarchical manner or simultaneously. The hierarchical manner focuses on balancing the assembly line first. Following that, the sequencing problem is solved to determine the sequence of models based on the obtained results from line balancing (Mosadegh *et al.*, 2012a). Specifically, the optimal model sequencing depends on the obtained results from line balancing (Hwang and Katayama, 2010).

A large number of studies have been conducted on the balancing and sequencing problems. For example, Hwang and Katayama (2010) solved balancing and sequencing problems in a hierarchical manner to minimize the number of workstations and also variance of workload. Mosadegh *et al.* (2012b) solved balancing and sequencing problems simultaneously in the mixed model assembly

line with station-dependent assembly times. They proposed a mixed-integer linear programming model to minimize the total utility work. Manavizadeh *et al.* (2015) solved balancing and sequencing problems simultaneously to minimize the cycle time, the wastages in each station, and also the work overload by proposing a heuristic approach. In this thesis, we solve balancing and sequencing problems simultaneously with the aim of minimizing the workstation length, the workstation cost, and task duplication cost in the mixed model assembly line.

Several aspects should be considered in the design of the mixed model assembly line. One aspect is satisfying the demand of each model of a product in a planning horizon. This demand is broken into h cycles in order to use a cyclic production strategy; h is the greatest common divisor of demand values. For example, if the demand of product m during the entire planning horizon is shown with De_m (where $m = 1, \ldots, M$ is the number of models), then the vector $de = de_1, \ldots, de_m$, where $de_m = \frac{De_m}{h}$, represents the product mix, a set of models which is called Minimum Part Set (MPS) and is manufactured in each repetitive cycle. Specifically, repetition of producing the MPS products for h times can satisfy the total demand in the planning horizon (Hyun *et al.*, 1998; Mosadegh *et al.*, 2012a). To explain the MPS vector, an example is presented here. Imagine that three models of a product, like A, B and C are produced in the same assembly line. The demand of each model during the entire planning horizon is 12, 8, and 4 respectively. Therefore, $h = 4$ and the MPS vector is $de = (3, 2, 1)$, which needs to be repeated 4 times in order to meet the total demand of each product in the planning horizon. In this thesis, demand of each model of a product is based on the MPS strategy.

Another aspect that should be considered in the mixed model assembly line is selecting the operator schedule. There are two types of operator schedule: early start and late start. In the early start schedule, all operations are started at their earliest start while in the late start schedule, operations are started in their

latest time which does not increase the duration of the entire project. The first schedule is more common and helps an assembly line to have the shortest length (Hyun *et al.*, 1998). Therefore, in this thesis the early start schedule is used. A third aspect is selecting the type of workstation. Two types of stations are used in the assembly line: closed and opened stations. In a closed station, operators can not cross boundaries to work while in an opened station, operators can cross boundaries (Hyun *et al.*, 1998). In this thesis, closed type stations are used.

The objective of this thesis is to balance and sequence the mixed model assembly line simultaneously based on the studies conducted by Bukchin and Rabinowitch (2006) and Mosadegh *et al.* (2012a). Bukchin and Rabinowitch (2006) focused on solving only the line-balancing problem in the mixed model assembly line. The objective of their study was to develop an integer linear programming model to minimize the total cost, which is the sum of the stations cost and tasks cost. In their study, common tasks were permitted to be assigned to different workstations with respect to the precedence constraints. This is called task duplication. On the other hand, Mosadegh *et al.* (2012a) focused on solving balancing and sequencing problems simultaneously in the mixed model assembly line. The aim of their study was to develop a mixed-integer linear programming ($MILP$) model to minimize the total utility work, which is the total amount of work that is not completed within the given length of workstation. In their study, the uncompleted tasks were passed to the utility workers. Also, they assumed that the cost of assigning tasks to workstations is minimal. In addition, they assumed that the common tasks are allowed to be assigned to different workstations with respect to the precedence constraints. Unlike the first study, the second study did not consider task cost, which includes task duplication cost. Specifically, the authors of the first study divided the total tasks cost into two parts: a fixed part and a variable part. The fixed part is associated with the cost of performing each task in a single station without duplication while the variable part is dependent

on task duplication, namely, duplication cost of machinery and tools.

This thesis combines different conditions of the above two studies to simultaneously balancing and sequencing problems in the mixed model assembly line. One of the main contributions of this thesis is to develop a mixed-integer linear programming ($MILP$) model to minimize the workstation length, workstation cost, and task duplication cost. Therefore, we deal with a multi-objective function in our mathematical model, but we need a single solution point to minimize these objectives simultaneously. Hence, the multi-objective function is transferred to a single-objective function by adding weight to each objective. These weights show the importance of each objective.

The number of opened workstations in the assembly line as well as length of each workstation are unknown in our mathemathical model. We focus on the mathematical model of the study conducted by Mosadegh *et al.* (2012a), and we call this mathematical model the reference mathematical model in this thesis. Modifications are made in the reference mathematical model to minimize the length of workstation, workstation cost as well as task duplication cost. The first modification is that the utility work is ignored in this thesis for two reasons: first, our mathematical model will be complicated if the utility work is considered; second, if the utility work goes outside the workstation boundary, as it occured in the reference mathematical model, we cannot identify the exact length of the assembly line. The second modification is that the station cost and task cost are considered in our mathematical model based on the study conducted by Bukchin and Rabinowitch (2006). However, this task cost includes only task duplication costs associated with assigning common tasks to different workstations, because we believe that performing each task in a single workstation has unavoidable task cost that cannot be optimized. Therefore, our mathematical model will focus only on minimizing the task duplication cost. Details of the proposed mathematical model will be discussed in the next section. The following assumptions are made

to deal with the problem:

- There is a conveyor which moves along workstations with a constant speed.
- Fixed rate launching is used to launch different models of a product into the conveyor.
- All workstations are closed type and there is no buffer between the workstations.
- The number of workstations is unknown in the assembly line and the gap is not permitted between the two consecutive workstations.
- The early start schedule with a reference point for each work station is used for all workstations. Therefore, operators are not allowed to work beyond the reference point.
- The utility work is ignored in all workstations.
- The common tasks can be assigned to different workstations with consideration of related precedence constraints; also, there is no conflicting precedence relationships among tasks of different models of a product.
- Operation times are deterministic and known for all tasks while they might differ in various models of a product.
- Demand of each model of a product must be satisfied in a planning horizon with respect to the cyclic production strategy in the assembly line.
- The time of moving workers along a workstation from the previous product to the next product is ignored.

In addition, each model is specified by its order in the MPS because tasks' assignment might be different in models. Therefore, the order of models is shown by $A_1B_1C_1A_2B_2A_3$ instead of $ABCABA$.

3.2. Problem formulation

In this section, a mathematical model is proposed based on the study conducted by Mosadegh *et al.* (2012a). A mixed-integer linear programming ($MILP$) model is introduced in this section to minimize the workstation length, workstation cost, and task duplication cost. Therefore, the length of each workstation as well as the number of workstations are unknown in the $MILP$ model. A branch and bound ($B\&B$) algorithm is used to solve the model for small size problems. The indices, parameters and variables of the $MILP$ model are as follows:

Indices:

m	Index for model
t	Index for task
h	Indice for tasks
s	Index for sequence
k	Index for station

Parameters:

M	Total number of models where models are indexed by $m = 1, \ \ldots, \ M$
T	Total number of tasks where tasks are indexed by $t = 1, \ \ldots, \ T$
S	Total number of sequences where sequences are indexed by $s = 1, \ \ldots, \ S$
K	Total number of workstations where workstations are indexed by $k = 1, \ \ldots, \ K$
v	Speed of conveyor
Lr	Launching rate of each model
$Pre_{m,t}$	Set of immediate precedent tasks for task t of model m

$D_{m,t}$ Assembly time for task t of model m

d_m Demand of model m in the MPS

SC Station cost, fixed cost associated with each workstation

TC Task duplication cost

B A large positive number

$j_{m,t}$ Binary data which equals to 1 if $D_{m,t} \geq 0$, 0 otherwise

Variables:

Continuous Variables:

w_k Length of workstation k

$p_{s,k}$ Start position of operator at sequence s in workstation k

Binary Variables:

$x_{m,t,s,k}$ Binary variable which equals to 1 if task t of model m at sequence s is assigned to workstation k, 0 otherwise

$a_{t,k}$ Binary variable which equal to 1 if task t of any model is assigned to workstation k

z_k Binary variable which equals to 1 if workstation k is open, 0 otherwise

$y_{m,s}$ Binary variable which equals to 1 if sequence s is assigned to model m

MILP Model

Minimize:

$$Objective = f_1 \cdot \sum_k w_k + f_2 \cdot \sum_k SC \cdot z_k + f_3 \cdot \sum_t TC \cdot ((\sum_k a_{t,k}) - 1) \quad (3.1)$$

Subject to:

$$p_{s,k} + (\sum_{m} \sum_{t} D_{m,t} \cdot x_{m,t,s,k}) \cdot v \leq w_k \ ; \quad \forall (s,k) \tag{3.2}$$

$$p_{s,k} + (\sum_{m} \sum_{t} D_{m,t} \cdot x_{m,t,s,k}) \cdot v - Lr \cdot v \leq p_{s+1,k} \ ; \quad \forall (s,k) | (s < S) \tag{3.3}$$

$$p_{S,k} + (\sum_{m} \sum_{t} D_{m,t} \cdot x_{m,t,s,k}) \cdot v - Lr \cdot v \leq p_{1,K} \ ; \quad \forall (k) \tag{3.4}$$

$$\sum_{k} x_{m,t,s,k} = y_{m,s} \cdot j_{m,t} \ ; \quad \forall (m,t,s) \tag{3.5}$$

$$\sum_{s} y_{m,s} = d_m \ ; \quad \forall (m) \tag{3.6}$$

$$\sum_{m} y_{m,s} = 1 \ ; \quad \forall (s) \tag{3.7}$$

$$\sum_{k} k \cdot x_{m,h,s,k} \leq \sum_{k} k \cdot x_{m,t,s,k} \ ; \quad \forall (m,t,s,h) | h \in Pre_{m,t} \tag{3.8}$$

$$w_k \leq B \cdot z_k \ ; \quad \forall (k) \tag{3.9}$$

$$x_{m,t,s,k} \leq z_k \ ; \quad \forall (m,t,s,k) \tag{3.10}$$

$$z_k \geq z_{k+1} \ ; \quad \forall (k) \tag{3.11}$$

$$a_{t,k} \geq x_{m,t,s,k} \ ; \quad \forall (m,t,s,k) \tag{3.12}$$

$$x_{m,t,s,k},\ y_{m,s},\ a_{t,k} \text{ and } z_k \text{ are binary} \tag{3.13}$$

$$p_{s,k} \text{ and } w_k \text{ are greater than equal zero} \tag{3.14}$$

The objective function of the model in Eq. (3.1) minimizes the length of workstations and also the total cost, which is the sum of workstations cost and tasks duplication cost. The constraint set in Eq. (3.2), declares that all operations should be performed within the length of the workstation. The constraint given in Eq. (3.3) determines the starting position of the operator in the workstation after finishing each task of each model in each sequence (except for the last sequence). The constraint given in Eq. (3.4) determines the start position of operator in the last sequence of each cycle. The constraint in Eq. (3.5) states that each task of each model is assigned to a particular sequence, if its model is assigned to that sequence before. Eq. (3.6) emphasises that the demand for each model in the MPS must be satisfied. Eq. (3.7) guarantees that each model is assigned to a specific sequence. Therefore, all tasks of a special model are completed in the same sequence. Precedence constraints are satisfied in the constraint set of Eq. (3.8). The constraint set in Eq. (3.9) declares that the workstation length will be zero if that workstation is not open in the assembly line. Eqs. (3.10) and (3.11) and Eq. (3.12) impose the logical constrains on the binary variables. Specifically, the constraint set in Eq. (3.10) states that a task is assigned to a particular workstation if that workstation is open. Eq. (3.11) prevents gap between two consecutive opened workstations in the assembly line. Constraint set of Eq. (3.12) declares that similar tasks of different models can be assigned to different workstations. Eqs. (3.13) and (3.14) shows binary variables and variables that are greater than equal zero respectively.

3.3. Model extension

In this section, we extend the mathematical model, which was presented in section 3.2. In section 3.1, we assumed that the conveyor moves along workstations with a constant speed. Therefore, it moves work pieces steadily from station to station. Hence, it has a continuous motion, and we can have two or more assemblies in one station at the same time. Also, we assumed that fixed rate launching is used to launch different models of a product into the conveyor. This means that we have a constant cycle time, which is equal for all models of a product in the assembly line. Therefore, the assembly line is paced. Now, we will change configuration of the assembly line in order to know how it affects our mathematical model. Specifically, we assume that we have a paced line with a synchronous configuration, which means that the conveyor has an intermittent motion. Therefore, we have a mixed model synchronous assembly lines ($MMSAL$) in the model extension. In this situation, the conveyor moves work pieces between stations periodically. This means that work pieces stay in workstations for a fixed time period, which is at least equal to the maximum operation time, and then they move to the next station at the same time. Therefore, only one work piece is available in each station that needs to be assembled. The mixed model synchronous assembly line ($MMSAL$) are more common for assembling large products such as automotive, household appliances, aircrafts, and ships (Salehi *et al.*, 2013). There are a few studies in the past which focused on $MMSAL$. Kouvelis and Karabati (1999) minimized the cycle time by introducing an integer programming model to solve the scheduling problem in the $MMSAL$. Salehi *et al.* (2013) solved the sequencing problem of $MMSAL$ by exploiting simulated annealing algorithm. Then, they compared their results with Lingo 9 software. They focused on three objectives for solving this problem: minimizing the total idle cost, minimizing the total production rate variation cost, and minimizing the total set up cost. Faccio

et al. (2015) solved balancing and sequencing problems of $MMSAL$ hierarchically with using a supplementary flexible operator, which is called jolly operator. Objectives of their study are minimizing the number of jolly operators as well as minimizing work-overloads .

The proposed mathematical model in this section focuses on solving balancing and sequencing problems of the mixed model synchronous assembly lines ($MMSAL$) simultaneously. The following assumptions are made to deal with the problem:

- There is a conveyor which moves along workstations with an intermittent motion. Therefore, the synchronous configuration is used in the paced assembly line.
- Fixed rate launching is used to launch different models of a product into the conveyor. Therefore, models have a constant and equal cycle time.
- All workstations are closed type and there is no buffer between them.
- The number of workstations is unknown in the assembly line and the gap is not permitted between two consecutive workstations.
- The early start schedule with the zero reference point is used for all work stations. Therefore, all operations are started at the beginning of workstations.
- The utility work and also ideal time are ignored in all workstations.
- The common tasks can be assigned to different workstations with consideration of precedence constraints; also, there is no conflicting precedence relationships among tasks of different models of a product.
- Operation times are deterministic and known for all tasks while they might vary in different models of a product.

- Demand of each model of a product must be satisfied in a planning horizon with respect to the cyclic production strategy in the assembly line.
- The time of moving workers along a workstation from the previous product to the next product is ignored.

Now, two important outcomes, which are resulted from above assumptions, are discussed. First, workstations will have an equal length if the line configuration changes from the continuous motion to the synchronous motion because the length of workstation will be equal to the launching rate value multiplied by the conveyor speed value in this situation. Both launching rate and conveyor speed values are constant here. Therefore, we have the same length for all stations. Hence, we do not have the continuous variable, w_k, which was introduced in section 3.2, when the assembly line deals with the synchronous motion. With this outcome, objectives of the proposed mathematical model in section 3.2 will change to reach the new objectives in the proposed extended model in this section. Specifically, the extended model deals with minimizing the number of stations as well as minimizing the total cost, which is sum of stations cost and tasks duplication cost. Second, $p_{s,k}$ is equal to zero because we assumed that workstations have the zero reference point, which means that operations are performed on work pieces at the beginning of stations. Therefore, we do not have the continuous variable $p_{s,k}$, which was introduced in section 3.2, when the assembly line deals with the synchronous motion. The proposed extended model is presented as follows:

MILP Model

Minimize:

$$Objective = \sum_{k} SC \cdot z_k + \sum_{t} TC \cdot ((\sum_{k} a_{t,k}) - 1) \tag{3.15}$$

Subject to:

$$(\sum_{m} \sum_{t} D_{m,t} \cdot x_{m,t,s,k}) \leq Lr \cdot z_k \ ; \quad \forall (s,k) \tag{3.16}$$

$$\sum_{k} x_{m,t,s,k} = y_{m,s} \cdot j_{m,t} \ ; \quad \forall (m,t,s) \tag{3.17}$$

$$\sum_{s} y_{m,s} = d_m \ ; \quad \forall (m) \tag{3.18}$$

$$\sum_{m} y_{m,s} = 1 \ ; \quad \forall (s) \tag{3.19}$$

$$\sum_{k} k \cdot x_{m,h,s,k} \leq \sum_{k} k \cdot x_{m,t,s,k} \ ; \quad \forall (m,t,s,h) | h \in Pre_{m,t} \tag{3.20}$$

$$x_{m,t,s,k} \leq z_k \ ; \quad \forall (m,t,s,k) \tag{3.21}$$

$$z_k \geq z_{k+1} \ ; \quad \forall (k) \tag{3.22}$$

$$a_{t,k} \geq x_{m,t,s,k} \ ; \quad \forall (m,t,s,k) \tag{3.23}$$

$$x_{m,t,s,k},\ y_{m,s},\ a_{t,k} \text{ and } z_k \text{ are binary} \tag{3.24}$$

The objective function of the model in Eq. (3.15) minimizes the total cost, which is the sum of the workstations cost and tasks duplication cost. like the constraint set in Eq. (3.2) of section 3.2, Eq. (3.16) declares that all operations should be performed within the length of the workstation. However, the continuous variable of $p_{s,k}$ has been removed in Eq. (3.16) based on the second outcome of changing conveyor's motion which was discussed before. In addition, the workstation length is constant and known in Eq. (3.16) based on the first outcome of changing conveyor's motion which was discussed before. Constraints in Eq. (3.3) and Eq. (3.4) of section 3.2 have been removed in the above model because $p_{s,k}$ is equal to zero. Eq. (3.5) to Eq. (3.8) are remained the same from section 3.2 and inserted into the above model as constraints from Eq. (3.17) to Eq. (3.20) respectively. The constraint given in Eq. (3.9) is removed in the above model because of having the constant workstation length that affects Eq. (3.9) when the workstation length is equal to the launching rate multiplied by the speed of the conveyor. Constraints from Eq. (3.10) to Eq. (3.13) are kept the same from section 3.2 and inserted into the above model as constraints from Eq. (3.21) to Eq. (3.24) respectively. The constraint in Eq. (3.14) is not considered in the above model because of the first and second outcomes of changing the conveyor's motion which were discussed before introducing the above model.

Chapter 4

The Proposed Algorithm

Balancing and sequencing problems are individually NP-hard in the mixed model assembly lines (Mosadegh *et al.*, 2012b). This comes up with this result that our proposed model, which includes balancing and sequencing problems simultaneously, is also NP-hard. Therefore, a metaheuristic method is used to solve the proposed model in a short amount of time. Specifically, metaheuristic methods are approximate algorithms, which explore search spaces to find near-optimal solutions (Blum and Roli, 2003). These methods, which are utilized for optimization of various large-sized problems, can solve problems in a reasonable time (Talbi, 2009). Different classes of metaheuristic methods are as follows (Battaia and Dolgui, 2013): neighbourhood methods, evolutionary approaches, and swarm intelligence based metaheuristics. Neighbourhood methods include various optimization techniques such as Tabu search (TS) (Ozcan *et al.*, 2009; Kalayci and Gupta, 2014), Kangaroo method (Minzu and Henrioud, 1998), Greedy randomized adaptive search procedure ($GRASP$) (Guschinskaya *et al.*, 2011; Essafi *et al.*, 2012), and Simulated annealing (SA) (Cercioglu *et al.*, 2009; Ozcan, 2010). Evolutionary approaches are divided into different methods, namely, differential evolution methods (Nourmohammadi and Zandieh, 2011; Mozdgir *et al.*, 2013), Imperialist competitive algorithms (Bagher *et al.*, 2011), and Genetic algorithms

(GA) (Akgunduz and Tunali, 2010; Tiacci, 2015). Swarm intelligence based metaheuristics also include several methods such as Particle swarm optimization algorithms (Nearchou, 2011; Chutima and Chimklai, 2012), Bees algorithms (Tapkan *et al.*, 2012), and Ant colony optimization (ACO) (Simaria and Vilarinho, 2009; Yagmahan, 2011). The most common techniques of metaheuristic methods are tabu search, simulated annealing, ant colony optimization, and genetic algorithm. In this thesis, a combination of Genetic algorithm (GA) and Linear programming (LP) is utilized to create a hybrid genetic algorithm, which solves the proposed mathematical model in section 3.2 effectively. In the following section, different steps of the hybrid genetic algorithm, which is used in this thesis, are discussed. Then, these steps are illustrated in a flowchart.

4.1. Hybrid genetic algorithm

Currently, researchers are more interested to use hybrid metaheuristics algorithms instead of applying a single metaheuristic method, because hybrid algorithms enable them to solve big real-world problems more effectively. A combination of a metaheuristic method with other metaheuristic methods or with other techniques outside metaheuristic methods is called a hybrid metaheuristic (Gendreau and Potvin, 2010). Different classifications have been defined for hybrid metaheuristics. One classification, which is based on the study conducted by Raidl (2006) is presented here. This classification is categorized with respect to four factors: type of algorithms, level of combination, order of execution, and control strategy. Type of algorithms determine what kind of algorithms are combined. For example, different metaheuristic methods can be combined or one metaheuristic method can be combined with an exact technique such as branch and bound (BB) or linear programming (LP). In this thesis, a genetic algorithm, which is one type of metaheuristic methods, is combined with LP. Level of combination

is divided into two levels: high level versus low level. In the high-level combination, identities of algorithms are maintained and there is no strong or direct internal relationship between algorithms. This means that algorithms work independently. In contrast, in the low-level combination, algorithms depend on each other in terms of different factors like function or individual components. Order of execution, which determines the implementation sequence of algorithms, is divided into three groups: batch execution, parallel execution, and interleaved execution. For example, in the batch execution, algorithms are performed back to back and results of each algorithm are used as the input for the next algorithm. Finally, control strategies are divided into two groups: integrative and collaborative. Integrative strategy uses one algorithm as a subordinate algorithm, which is embedded in a main algorithm. However, in the collaborative strategy, there is no relation between algorithms but they can exchange information. With consideration of the above explanations, an integrative strategy is employed in this thesis. Specifically, a linear programming, which is a subordinate algorithm, is embedded in the genetic algorithm as the main algorithm. Therefore, a hybrid genetic algorithm (HGA) is used in this thesis.

The HGA has attracted more attention in the recent decade for solving balancing and sequencing problems of an assembly line. Haq *et al.* (2006) utilized a HGA, which was combination of a genetic algorithm and the modified rank position weight method to solve a mixed model assembly line balancing problem with the aim of minimizing the number of workstations. Wang *et al.* (2008a,b) exploited a HGA, which was a combination of a genetic algorithm and a simulated annealing for solving the sequencing problem with limited intermediate buffers in the mixed model assembly line. Wang *et al.* (2008a) focused on minimizing the total production rate variation, the total set-up, and the total assembly cost while Wang *et al.* (2008b) focused on minimizing the variation in parts usage and minimizing the makespan. Akpinar and Bayhan (2011) used

a *HGA* to solve a mixed model assembly line balancing problem with parallel workstations. They exploited three well known heuristics, kilbridge and wester heuristic, phase-I of moodie and young method, and ranked positional weight technique, with a genetic algorithm. Objectives in their study were: minimizing the number of workstations, maximizing the workload smoothness between workstations, and maximizing the workload smoothness within workstations. Kalayci *et al.* (2014) employed a *HGA* that combines a genetic algorithm with a variable neighbourhood search method to solve a sequence dependent disassembly line balancing problem. However, in this thesis, a *HGA* is employed to solve balancing and sequencing problems simultaneously to minimize the workstation length, workstation cost, and also task duplication cost in the mixed model assembly line.

4.1.1. Genetic algorithm

A genetic algorithm is a stochastic search method, which was introduced by John Holland in the 1970s. This method, which is classified as one of the metaheuristic methods, has two main advantages: first, it focuses on the population instead of a single point to search for solutions. Thus, many solutions are investigated with this way to reach a near-optimal solution. Second, multiple fitness function in different forms can be applied in this algorithm (Tasan and Tunali, 2008). However, this algorithm has some disadvantages. For example, it has a slow and premature convergence. In addition, it is not able to do local search. A genetic algorithm usually starts with an initial solution space, which is known as an initial population. This population is generated randomly. Candidate solutions (individuals) in the population are encoded to start the search process (Tasan and Tunali, 2008). In the following subsections, different steps of *HGA*, which have been used in this thesis, are discussed in detail.

4.1.2. Solution representation

Generating a good solution representation is the first step to implement the genetic algorithm. The solution representation, which is encoded in this subsection, includes three phases to address balancing and sequencing problems simultaneously in the mixed model assembly line. In the first phase, each task is assigned to only one workstation. This means that common tasks between different models are not allowed to be performed in different workstations. In the second phase, common tasks are permitted to be assigned to different workstations but common tasks in different occurrences of one model are assigned to only one workstation. In the third phase, all common tasks are allowed to be assigned to different workstations, even common tasks in different occurrences of one model. In this way, we can achieve the target solution in a short amount of time, because we have limited the search spaces to the above three phases. In this thesis, the initial solution is only generated for phase one. Therefore, phase two and phase three use the generated initial solution from phase one. The initialization of each segment in phase one should generate a valid chromosome. However, there are some situations that the initialization leads to having an infeasible solution such as having a small number of opened workstations or assigning several tasks to only one workstation. In these situations, the infeasible solution will be ignored and the next initialization will be started.

Decoding the defined solution representation in this subsection gives us the values of binary variables, which are $x_{m,t,s,k}$, $a_{t,k}$, z_k, and $y_{m,s}$. Hence, constraint sets from Eq. (3.5) to Eq. (3.8) and constraint sets from Eq. (3.10) to Eq. (3.12) in section 3.2 are satisfied by this solution representation in different segments. More explanations will be provided in the following subsections to show how these constraints are satisfied by decoding the solution representation. The rest of the constraints, which are related to continuous variables, are satisfied by solving the LP-subproblem. The proposed solution representation is illustrated in Figure

4.1. As is clear in Figure 4.1, this representation includes different segments, which clarify various steps in the feasible tasks' assignment to each workstation and also feasible models' assignment to each sequence based on the defined three phases. These segments are explained in the following sub-subsections.

Segment-1

This segment, which is used in all three phases, provides information about a feasible sequence of models in a production cycle. Specifically, the defined chromosome in this segment determines the assignment of various models to different sequences. Therefore, the length of the chromosome is equal to the sum of demands of different models in the MPS. For example, if the MPS is $de = (2, 1, 2)$ for three models, A, B and C, the length of the chromosome should be equal to five, which means that we have five sequences. Each gene in the chromosome shows the name of the model which has been assigned to a particular sequence with respect to the MPS vector. For instance, we have defined 2 demands for model A, 1 demand for model B, and 2 demands for model C in the above example. Therefore, model A and model C are repeated twice in the chromosome, but there is only one gene for model B in the chromosome. Hence, the sequence of $(A_1B_1C_1C_2A_2)$ can be an alternative feasible solution for segment-1. To sum up, α_S is presented by the index of the model, which has been assigned to the sequence S. This index is used to decode the binary variable of $y_{m,s}$ using Eq. (4.1). Since α_S takes only a single value m, the decoded values of $y_{m,s}$ satisfy the constraint set in Eq. (3.7) in section 3.2. In addition, since the length of the chromosome is equal to the sum of demands of different models in the MPS, the constraint set in Eq. (3.6) in section 3.2 will also be satisfied with this chromosome.

$$y_{m,s} = \begin{cases} 1 & ; \text{if } \alpha_S = m \\ \\ 0 & ; \text{otherwise} \end{cases} \tag{4.1}$$

Segment-2

As with the previous segment, this segment is used in all three phases. The chromosome in this segment shows the total number of workstations which are opened in the assembly line. This number should be less than the given maximum number of workstations. The binary variable z_k is decoded by this segment. Therefore, the constraint given in Eq. (3.11) is satisfied by decoding this chromosome.

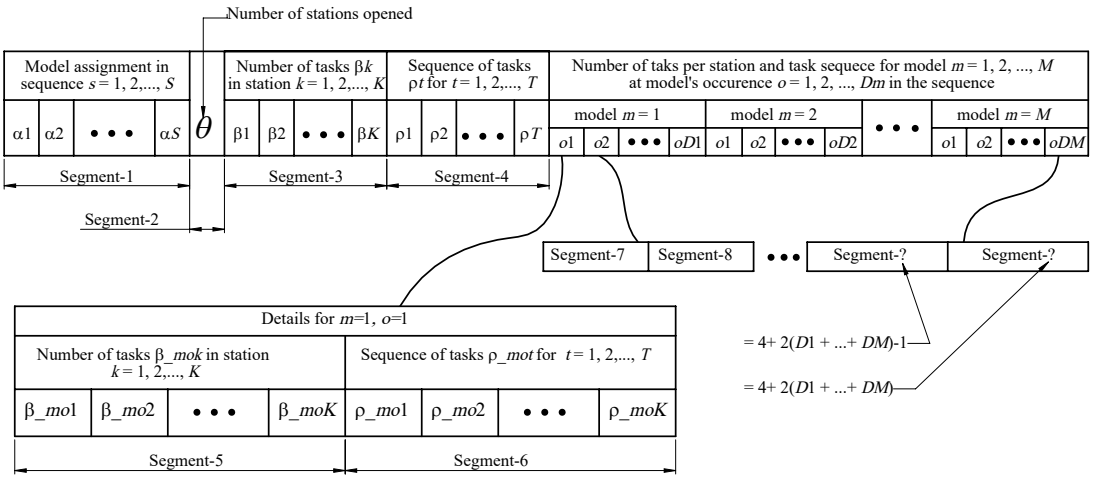


Figure 4.1: Solution Representation

Segment-3 and Segment-4

These two segments are used only in the first phase of the algorithm. Therefore, each task is assigned to only one workstation, which means that common tasks between different models are not allowed to be performed in different workstations in these segments. Segment-3 provides information about the total number of

tasks in each workstation. Specifically, it determines how many tasks are assigned to each workstation. Therefore, the length of the chromosome is equal to the maximum number of workstations, and the value of each gene in the chromosome shows the number of tasks which have been assigned to each workstation. A combined precedence diagram is used in this segment to determine the number of tasks in workstations. In addition, the encoded solution in this segment should satisfy the following equations:

$$\beta_1 + \beta_2 + ... + \beta_\theta = T \tag{4.2}$$

$$\beta_{\theta+1} + \beta_{\theta+2} + ... + \beta_K = 0 \tag{4.3}$$

Eq. (4.2) ensures that the sum of the number of tasks which have been assigned to different workstations are equal to the total number of tasks, T. Eq. (4.3) states that the remaining workstations should be equal to zero if they are greater than θ. A small example is presented to show how the chromosomes of segment-3 and segment-4 are obtained. Imagine that there are three models with the precedence diagrams, which are illustrated in Figure 4.2. The combined precedence diagram of these three models is shown in Figure 4.3. Also imagine that there are four opened workstations in the assembly line while the maximum number of workstations is equal to seven. Therefore, these seven tasks should be assigned to four workstations according to the combined precedence diagram. One alternative feasible solution for this segment can be (2 2 2 1 0 0 0), which means that 2 tasks are assigned to the first, second, and third stations separately, and one task is assigned to the fourth station while the rest of the stations are not open in the assembly line. Therefore, the above encoded solution satisfies the combined precedence diagram, Eq. (4.2), and also Eq. (4.3).

Segment-4 gives information about the sequence of tasks. The length of the chromosome in this segment is equal to the total number of tasks. The value

of each gene in the chromosome shows the task number that has been assigned to a particular sequence. As with the previous segment, segment-4 employs the combined precedence diagram to determine the sequence of tasks. For example, one alternative feasible solution for this segment can be (1 2 3 4 5 6 7) based on Figure 4.3.

A combination of chromosomes in segment-1, segment-2, segment-3, and segment-4 helps us to decode binary variables of $x_{m,t,s,k}$ and $a_{t,k}$. One example is presented here to show how these two binary variables are decoded. To explain this example, the combined precedence diagram in Figure 4.3 is used. We assume that the maximum number of workstations is equal to 7 while only 4 workstations are open in the assembly line. In addition, solutions which have been obtained for segment-1, segment-3, and segment-4 are utilized in this example, and are shown in Figure 4.4. Then, Table 4.1 shows the tasks' assignment to each workstation based on Figure 4.4. As is clear in Table 4.1, the models' sequence corresponds with the chromosome of segment-1 in Figure 4.4, and the tasks' sequence corresponds with the chromosome of segment-4 in Figure 4.4. Each number inside Table 4.1 determines a particular workstation to which a particular task has been assigned. This number is obtained from chromosomes of segment-3 and segment-4 in Figure 4.4. For example, the first gene of the chromosome in segment-3 assigns the first two tasks to the first station. Based on the chromosome of segment-4, the first two tasks are task 1 and task 2. Therefore, task 1 and task 2 of all models across all sequences are assigned to station 1 in Table 4.1. The rest of the tasks' assignment in Table 4.1 are carried out based on the above logic. The binary variable of $a_{t,k}$ is decoded based on the tasks' assignment in Table 4.1 as follows: $a_{1,1}$=1, $a_{2,1}$=1, $a_{3,2}$=1, $a_{4,2}$=1, $a_{5,3}$=1, $a_{6,3}$=1, and $a_{7,4}$=1.

In addition, the binary variable of $x_{m,t,s,k}$ is decoded by extracting a table from Table 4.1. Therefore, Table 4.2 shows the results for decoding the binary variables of $x_{m,t,s,k}$. As is clear in Table 4.2, a particular task of all models has

been assigned to a particular station across all sequences. For example, the first column of tasks' assignment in Table 4.2 shows that task 1 of all models has been assigned to station 1 across all sequences. Therefore, t and k are constant and equal to 1 in the binary variable of $x_{m,t,s,k}$ in the first column of Table 4.2 while the models' numbers and their sequences change. The rest of the assignments are carried out based on the above logic. Finally, the decoded values of $x_{m,t,s,k}$ and $a_{t,k}$ can satisfy constraint sets in Eqs. (3.5) and (3.8) and also constraint sets from Eq. (3.10) to Eq. (3.12) in section 3.2.

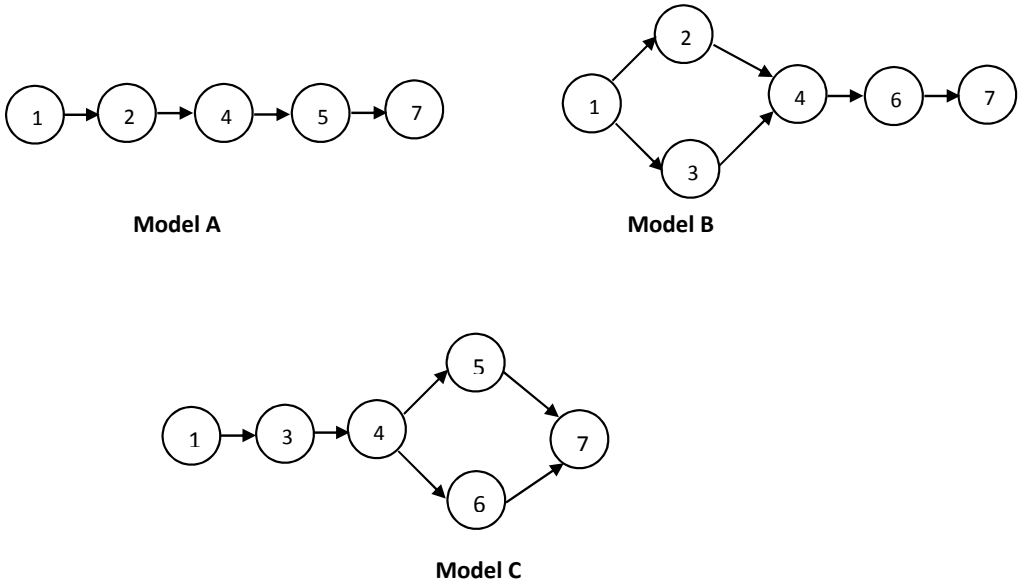


Figure 4.2: Precedence diagram for models A, B and C

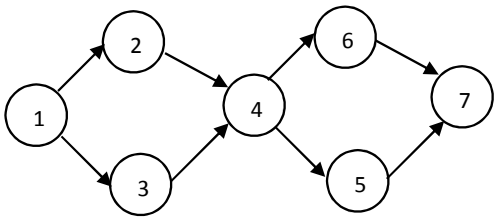


Figure 4.3: Combined precedence diagram for models A, B, and C

A1	B1	C1	C2	A2		
2	2	2	1	0	0	0
1	2	3	4	5	6	7

Chromosome of Segment- 1
Chromosome of Segment- 3
Chromosome of Segment- 4

Figure 4.4: Obtained chromosomes from segment-1, segment-3, and segment-4

Table 4.1: Assignment of a particular task to a particular station for all models accross all sequences (Phase one)

	Tasks						
Model's Sequence	1	2	3	4	5	6	7
A_1	1	1	2	2	3	-	4
B_1	1	1	-	2	-	3	4
C_1	1	-	2	2	3	3	4
C_2	1	-	2	2	3	3	4
A_2	1	1	2	2	3	-	4

Table 4.2: Decoded binary variable of $x_{m,t,s,k}$ from Table 4.1

	Assignment of a particular task to the particular station for all models				
	$t = 1, k = 1$	$t = 2, k = 1$	$t = 3, k = 2$	.....	$t = 7, k = 4$
Model's Sequence	$x_{m,t,s,k}$	$x_{m,t,s,k}$	$x_{m,t,s,k}$		$x_{m,t,s,k}$
A_1	(1,1,1,1)	(1,2,1,1)	-	.....	(1,7,1,4)
B_1	(2,1,2,1)	(2,2,2,1)	(2,3,2,2)	.....	(2,7,2,4)
C_1	(3,1,3,1)	-	(3,3,3,2)	.....	(3,7,3,4)
C_2	(3,1,4,1)	-	(3,3,4,2)	.....	(3,7,4,4)
A_2	(1,1,5,1)	(1,2,5,1)	-	.....	(1,7,5,4)

Segment-5, Segment-6

These two segments are used in the second and third phase of the proposed algorithm. In the second phase, common tasks between models can be assigned to different workstations but common tasks at different occurrences of one model are assigned to only one workstation. However, in the third phase, all common tasks are allowed to be assigned to different workstations. This phase includes the assignment of common tasks of one model to different workstations at different occurrences of that model in the sequence.

Segment-5 determines a feasible tasks' assignment of the first model to different workstations at the first occurrence of the model in the sequence. Specifically, this segment determines how many tasks of the first model are assigned to different workstations at the first occurrence of the model. In contrast, segment-6 specifies the sequence of tasks of the first model at its first occurrence in the sequence. With these explanations, the length of the chromosome is equal to the maximum number of workstations in segment-5 while that length is equal to the total number of tasks in segment-6. The value of each gene in the chromosome of segment-5 shows the number of tasks of the first model that have been assigned

to a particular workstation at the first occurrence of the model in the sequence. However, the value of each gene in the chromosome of segment-6 shows the task number of the first model that has been assigned to a particular sequence at the first occurrence of the model. Precedence diagram of the first model should be considered in both segments. Moreover, the encoded solution in segment-5 should satisfy Eqs. (4.4) and (4.5).

$$\beta_{m,o,1} + \beta_{m,o,2} + ... + \beta_{m,o,\theta} = T \tag{4.4}$$

$$\beta_{m,o,\theta+1} + \beta_{m,o,\theta+2} + ... + \beta_{m,o,K} = 0 \tag{4.5}$$

Eq. (4.4) ensures that the sum of the number of tasks in the first model that have been assigned to different workstations at the first occurrence of the model are equal to the total number of tasks, T. This means that all tasks are used to be assigned in segment-5 and also segment-6. Then, the redundant tasks will be ignored in the investigation process based on the precedence diagram of each model. Eq. (4.5) states that the remaining workstations should be equal to zero if they are more than θ. As with the previous two segments, binary variables of $x_{m,t,s,k}$ and $a_{t,k}$ are decoded from defined chromosomes in segment-1, segment-2, segment-5, and segment-6, but the major difference is that these two binary variables show the assignment of common tasks to different workstations in segment-5, and segment-6.

One example is presented here to decode binary variables of $x_{m,t,s,k}$ and $a_{t,k}$ for phase two. This example is based on the assumptions of the example provided in segment-3 and segment-4 to decode these two binary variables. First, we define a solution for segment-5. As was demonstrated in the chromosome of segment-1 in Figure 4.4, the first model is A in the models' sequence. Therefore, the precedence diagram for model A, which was shown in Figure 4.2, is used to define a feasible solution for segment-5. This solution is (1,2,1,1,0,0,0), which

determines the number of tasks of the first model, A, that have been assigned to different workstation at the first occurrence of model A. The solution of segment-5 for model A remains constant for the rest of the occurrences of this model, because common tasks in different occurrences of one model are assigned to only one workstation in phase two.

On the other hand, one alternative feasible solution for segment-6 can be (1,2,4,5,7,0,0), which shows the sequence of tasks of model A at its first occurrence. This solution, which is constant in different occurrences of model A, is based on the precedence diagram of model A. This process is carried out for the rest of the models, B and C, to obtain related chromosomes. Results of defining chromosomes for these two segments for all three models are shown in Figure 4.5. Table 4.3 shows the tasks' assignment to each workstation based on Figure 4.5. As is clear in Table 4.3, models' sequence corresponds with the chromosome of segment-1 in Figure 4.4, and the tasks' sequence corresponds with the chromosomes of segment-6, segment-10, and segment-12 in Figure 4.5. Each number inside Table 4.3 determines a particular workstation to which a particular task of a particular model has been assigned. Table 4.3 shows that common tasks can be assigned to different workstations while common tasks in different occurrences of one model are assigned only to one workstation. The binary variable of $a_{t,k}$ is decoded based on the tasks' assignment in Table 4.3 as follows: $a_{1,1}$=1, $a_{2,1}$=1, $a_{2,2}$=1, $a_{3,2}$=1, $a_{3,1}$=1, $a_{4,2}$=1, $a_{4,3}$=1, $a_{5,2}$=1, $a_{5,3}$=1, $a_{6,3}$=1, and $a_{7,4}$=1.

In addition, the binary variable of $x_{m,t,s,k}$ is decoded by using data from Table 4.3 to create Table 4.4. Therefore, Table 4.4 shows results for decoding the binary variable of $x_{m,t,s,k}$ in phase two. As is clear in Table 4.4, common tasks have been assigned to different workstations, but common tasks at different occurrences of one model have been assigned to only one workstation. For example, task 3 of model B and model C has been assigned to stations 2 and 1 respectively, but task 3 of model C has been assigned only to station 1 at different occurrences of

model C.

Binary variables of $x_{m,t,s,k}$ and $a_{t,k}$ are decoded in a similar way to what has been explained above in the third phase, but the main difference is that common tasks of one model are also permitted to be assigned to various workstations in different occurrences of that model. One example is also presented here to decode binary variables of $x_{m,t,s,k}$ and $a_{t,k}$ for phase three. This example is based on the assumptions of the example provided in segment-3 and in segment-4 to decode these two binary variables. Like phase two, in phase three we define solutions for different segments. As was demonstrated in the chromosome of segment-1 in Figure 4.4, the first model is A in the models' sequence. Therefore, the precedence diagram for model A, which was shown in Figure 4.2, is used to define a feasible task's assignment for segment-5. Tasks' assignment can be varied for different occurrences of model A, because common tasks can be assigned to different workstations even in various occurrences of a model in phase three. Therefore, the solution of segment-5, which is related to the tasks' assignment of the first occurrence of model A, varies from the solution of segment-7, which is related to the tasks' assignment of the second occurrence of model A. The above logic is also used for tasks' assignment in different occurrences of model B and model C. The alternative feasible solutions for tasks' assignment in different occurrences of all three models are illustrated in Figure 4.6. Similarly, the tasks' sequence for each model is determined based on the precedence diagram of each model. The alternative feasible solutions for tasks' sequence of all three models are also shown in Figure 4.6.

Table 4.5 shows the tasks' assignment to each workstation based on Figure 4.6. As is clear in Table 4.5, the models' sequence corresponds with the chromosome of segment-1 in Figure 4.4, and the tasks' sequence corresponds with defined chromosomes for each model in Figure 4.6. Each number inside Table 4.5 determines a particular workstation to which a particular task of a particular

model in a particular sequence has been assigned. Table 4.5 shows that common tasks can be assigned to different workstations, even common tasks in different occurrences of one model. For example, task 6 of model B has been assigned to station 4 while task 6 of model C has been assigned to two different stations in different occurrences. These two stations are station 3 at the first occurrence of model C and station 4 at the second occurrence of model C. The binary variable of $a_{t,k}$ is decoded based on the tasks' assignment in Table 4.5 as follows: $a_{1,1}$=1, $a_{2,1}$=1, $a_{2,2}$=1, $a_{3,1}$=1, $a_{3,2}$=1, $a_{4,2}$=1, $a_{4,3}$=1, $a_{5,2}$=1, $a_{5,3}$=1, $a_{6,3}$=1, $a_{6,4}$=1 and $a_{7,4}$=1.

In addition, the binary variable of $x_{m,t,s,k}$ is decoded by using data from Table 4.5 to create Table 4.6. Therefore, Table 4.6 shows results for decoding the binary variable of $x_{m,t,s,k}$ in phase three. In Table 4.6, a particular task of a particular model has been assigned to a particular station in a particular sequence. Specifically, Table 4.6 shows the main feature of phase three that is all common tasks can be assigned to different workstations even common tasks in different occurrences of a model. For example, the second column in Table 4.6 shows that task 2 of model A has been assigned to station 2 at the first occurrence of model A while this task has been assigned to station 1 at the second occurrence of model A. However, task 2 of model B has been assigned to station 2 at its first occurrence. Decoded values of $x_{m,t,s,k}$ and $a_{t,k}$ can satisfy constraint sets in Eqs. (3.5) and (3.8) and also constraint sets from Eq. (3.10) to Eq. (3.12).

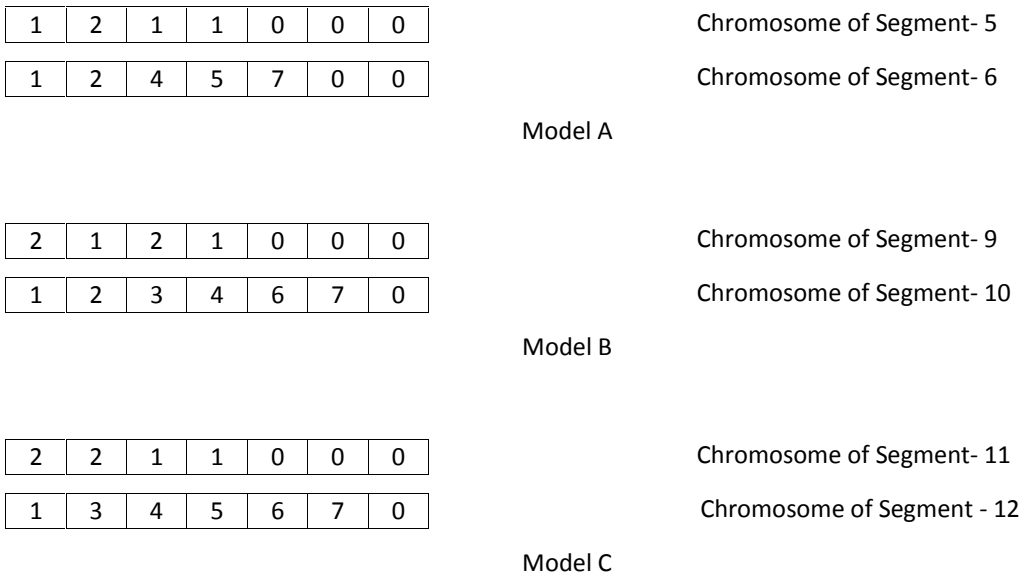


Figure 4.5: Obtained chromosomes for segment-5 to segment-12 for three models, A, B, and C (Phase two)

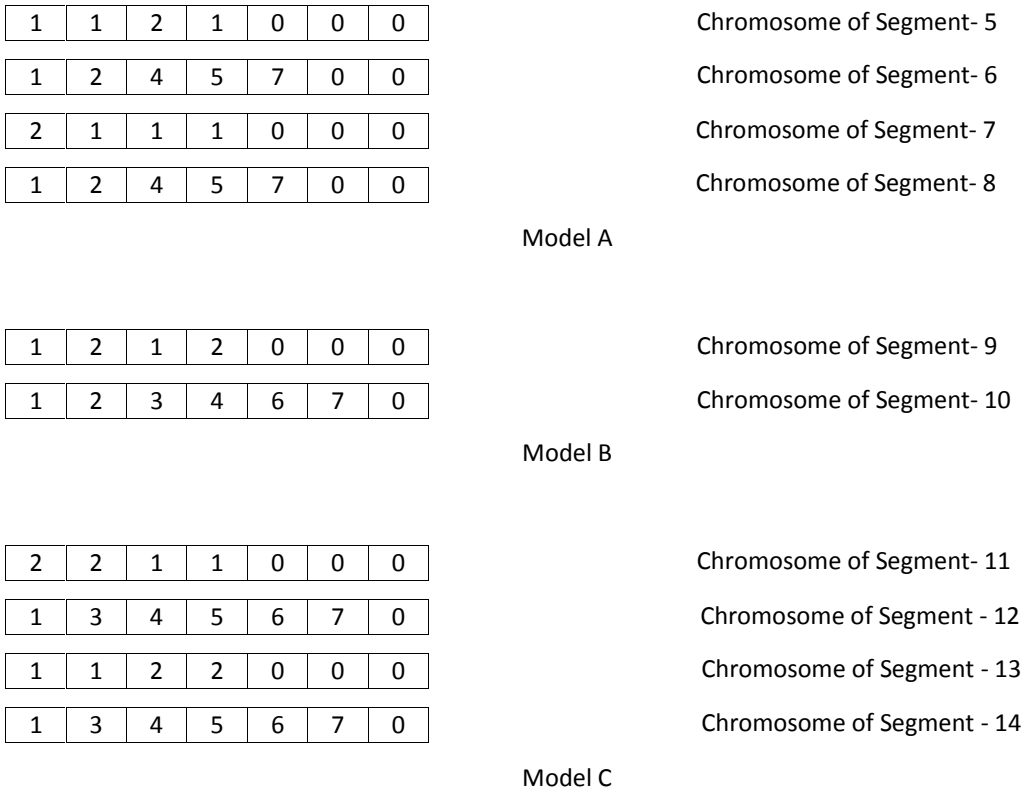


Figure 4.6: Obtained chromosomes for segment-5 to segment-14 for three models, A, B, and C (Phase three)

Table 4.3: Assignment of a particular task from a particular model to a particular station (Phase two)

	Tasks						
Model's Sequence	1	2	3	4	5	6	7
A_1	1	2	-	2	3	-	4
B_1	1	1	2	3	-	3	4
C_1	1	-	1	2	2	3	4
C_2	1	-	1	2	2	3	4
A_2	1	2	-	2	3	-	4

Table 4.4: Decoded binary variable of $x_{m,t,s,k}$ based on Table 4.3 for phase two

	Assignment of a particular task of a particular model to the particular station				
Model's Sequence	$x_{m,t,s,k}$	$x_{m,t,s,k}$	$x_{m,t,s,k}$	.....	$x_{m,t,s,k}$
A_1	(1,1,1,1)	(1,2,1,2)	-	.....	(1,7,1,4)
B_1	(2,1,2,1)	(2,2,2,1)	(2,3,2,2)	.....	(2,7,2,4)
C_1	(3,1,3,1)	-	(3,3,3,1)	.....	(3,7,3,4)
C_2	(3,1,4,1)	-	(3,3,4,1)	.....	(3,7,4,4)
A_2	(1,1,5,1)	(1,2,5,2)	-	.....	(1,7,5,4)

Table 4.5: Assignment of a particular task from a particular model to a particular station in a particular sequence (Phase three)

	Tasks						
Model's Sequence	1	2	3	4	5	6	7
A_1	1	2	-	3	3	-	4
B_1	1	2	2	3	-	4	4
C_1	1	-	1	2	2	3	4
C_2	1	-	2	3	3	4	4
A_2	1	1	-	2	3	-	4

Table 4.6: Decoded binary variable of $x_{m,t,s,k}$ based on Table 4.5 for phase three

	Task's assignment of a model to the particular station in a particular sequence				
Model's Sequence	$x_{m,t,s,k}$	$x_{m,t,s,k}$	$x_{m,t,s,k}$	.....	$x_{m,t,s,k}$
A_1	(1,1,1,1)	(1,2,1,2)	-	.....	(1,7,1,4)
B_1	(2,1,2,1)	(2,2,2,2)	(2,3,2,2)	.....	(2,7,2,4)
C_1	(3,1,3,1)	-	(3,3,3,1)	.....	(3,7,3,4)
C_2	(3,1,4,1)	-	(3,3,4,2)	.....	(3,7,4,4)
A_2	(1,1,5,1)	(1,2,5,1)	-	.....	(1,7,5,4)

4.1.3. Linear programming subproblem

In the previous subsection, the solution representation was presented to determine the values of binary variables for the $MILP$ model in section 3.2. In this section, a linear programming (LP) subproblem is introduced to obtain the optimal values of continuous variables corresponding to the values of binary variables. The proposed linear programming subproblem is formulated based on the $MILP$ model in section 3.2. Since the values of binary variables have been determined already, these variables are known in formulating the LP-subproblem. As a result, constraint sets from Eq. (3.5) to Eq. (3.8) and constraint sets in Eqs. (3.10) and (3.11) and Eq. (3.12) from the $MILP$ model in section 3.2 are removed in formulating the LP model, because they have been composed of only binary variables. On the other hand, constraint sets in Eqs. (3.2) and (3.3) and Eq. (3.4) are modified by setting $x_{m,t,s,k} = 1$ as the known binary variable; then, these constraints are inserted into the LP-subproblem. A similar logic is applied for the constraint set in Eq. (3.9) by setting $z_k = 1$; then, the modified constraint is inserted into the LP-subproblem. Moreover, knowing the binary variables of z_k and $a_{t,k}$ leads to removing the station cost term and the task duplication cost

term from the objective function of the $MILP$ model in section 3.2, because these two terms can be computed before solving the LP-subproblem. The complete LP-subproblem is presented below.

LP: given $(x_{m,t,s,k},\ z_k,\ y_{m,t},\ a_{t,k})$ **for all** (m,t,s,k)

Minimize:

$$Objective = \sum_k w_k \tag{4.6}$$

Subject to:

$$p_{s,k} + (\sum_m \sum_t D_{m,t}) \cdot v \leq w_k \ ; \quad \forall (s,k) \tag{4.7}$$

$$p_{s,k} + (\sum_m \sum_t D_{m,t}) \cdot v - Lr \cdot v \leq p_{s+1,k} \ ; \quad \forall (s,k) | (s < S) \tag{4.8}$$

$$p_{S,k} + (\sum_m \sum_t D_{m,t}) \cdot v - Lr \cdot v \leq p_{1,K} \ ; \quad \forall (k) \tag{4.9}$$

$$w_k \leq B \ ; \quad \forall (k) \tag{4.10}$$

In the above LP-subproblem, Eq. (4.10) can be removed if the maximum number of k is equal to the number of opened workstations, θ, in segment-2.

4.1.4. Fitness function

The fitness value of each chromosome of a population is evaluated by a fitness function, which is similar to the objective function of the proposed model in

section 3.2. Eq. (4.11) shows the fitness function, which is used for each chromosome after obtaining the binary variables and continuous variables from the genetic algorithm and LP- subproblem respectively.

$$Minimize(V) = f_1 \cdot \sum_k w_k + f_2 \cdot \sum_k SC \cdot z_k + f_3 \cdot \sum_t TC \cdot ((\sum_k a_{t,k}) - 1) \quad (4.11)$$

The population convergence toward the optimal solution stops the algorithm. Different stopping conditions are considered to be the convergence criteria such as evolving the maximum number of generations (Sivanandam and Deepa, 2007). In this thesis, the changing phase criteria, which leads to entering phase two from phase one and phase three from phase two, and the stopping criteria in phase three are determined based on two important conditions: evolving the maximum number of generations and finding no improvement in the value of the fitness function for a given number of generations.

4.1.5. Selection operator

The selection process is a strategy to select two individuals (chromosomes) as parents from the population for producing offspring for the next generation. Finding the fitter chromosomes, which generate offspring with the higher fitness, is the main goal of the selection process (Sivanandam and Deepa, 2007). Two important selection strategies are as follows: roulette wheel and tournament. The first selection method focuses on finding two parents based on a probability, which is proportional to the fitness of each chromosome. A roulette wheel, which has different sections corresponding to the probability of each individual, is spun M times; M is equal to the size of the population. In each spin, one individual is selected. Those individuals, which have the larger space (largest fitness) on the roulette wheel have more chance to be selected (Holland, 1975). Different steps in this method are described as follows (Gen and Cheng, 1997):

- Calculating the fitness value of each chromosome
- Summing the fitness values of chromosomes in the population to obtain the total fitness value of the population
- Dividing the fitness value of each chromosome by the above sum to obtain the selection probability of each chromosome
- Calculating the cumulative probability for each chromosome
- Generating a random number between 0 and 1
- Matching each random number with a chromosome based on the random number being within the range of its cumulative probability

In the second strategy, which is the tournament selection, a set of k individuals is randomly selected from the population; k is the size of the tournament. Then, a competition is held among k individuals. Specifically, an individual with the highest fitness is selected as the best one for producing offspring for the next generation. The number of repetitions of this procedure depends on the population size (Goldberg and Holland, 1988).

One important point that must be considered for these two methods is how to use the fitness function. Specifically, the roulette wheel selection method uses the objective function value as the fitness value for each chromosome in the maximization models. However, the objective function value needs to be converted if the purpose of the model is minimization. In this thesis, the proposed objective function in section 3.2 aims to minimize different objectives. Therefore, the fitness function in Eq. (4.11) needs to be converted to Eq. (4.12) if the selected method is the roulette wheel:

$$V' = \begin{cases} 1 & ; \text{if } \ V = V_{min} \\ \frac{V_{max} - V}{V_{max} - V_{min}} & ; \text{if } \quad V_{min} \leq V \leq V_{max} \\ 0 & ; \text{otherwise} \end{cases} \tag{4.12}$$

However, there is no need to convert the objective function in the tournament selection method, neither for maximization problems nor for minimization problems.

In this thesis, the tournament selection method is employed. The parent selection is made using replacement, which helps the best individual to be selected more often. The tournament size is two, which means that two individuals are randomly selected from the population; then, one individual with smallest fitness is selected as the best one, which is inserted into mating pool for producing an offspring. This procedure is repeated based on the total number of individuals in each population. In the mating pool, a random number is generated and assigned to each parent. Then, the defined random numbers are sorted based on the descending order. Following that, two parents are randomly selected in order to produce a new population by crossover operators. After randomly selecting two parents for reproduction generation, random numbers are created again for each individual and sorted to select the new two parents. The repetition of this procedure in the mating pool depends on the size of the mating pool.

4.1.6. Crossover operators

A crossover operator is a reproduction strategy which generates new offspring from two parents. Specifically, two parents are randomly selected from the mating pool as was mentioned in the previous subsection. Then, one random point, which is called a crossover point, is selected in the chromosome of each parent. Finally,

genes after this random point are exchanged between two parents to create new offsprings which have some features of both parents. This is the process that occurs when a single point cross over is used to create the new offspring (Talbi, 2009). However, there are various types of crossover operators with different steps such as two-point crossover, which two crossover points are randomly selected in the chromosome of each parent; then, genes between these two points are exchanged between both parents. The main goal of using a crossover operator is to generate a better offspring in terms of the fitness value (Sivanandam and Deepa, 2007). One important point about crossover operators is that they should generate a valid offspring (Talbi, 2009). This matter becomes significant when we deal with solving balancing and sequencing problems simultaneously, because the generated offspring should satisfy some constraints like precedence restrictions and demand of each model.

In this thesis, four crossover operators are presented in such a way that a whole segment is exchanged between the chromosomes of two parents to generate two new offsprings. The first crossover operator is performed in segment-2 of parents chromosome. Specifically, the number of opened workstations, θ, is exchanged between two parents in order to generate new offsprings in different phases but this exchange will affect different segments in three phases as follows: in phase one, the exchanging θ between two parents affects segment-3. Therefore, both segments-2 and 3 are exchanged between two parents while the remaining segments in each parent will be preserved in it's corresponding child. In phase two, the exchanging θ between two parents affects the first occurrence of each model. This means that chromosomes in segment-5, segment-9, and segment-11 will change in two parents. Therefore, all these segments with θ are exchanged between two parents in phase two while the remaining segments in each parent will be retained in its corresponding child. In phase three, all occurrences of models will change if the value of segment-2 is exchanged between two parents.

Therefore, the chromosomes in segment-5, segment-7, segment-9, segment-11, and segment-13 will also be exchanged between two parents.

The second operator is a sequence crossover operator, which is employed in all three phases. Specifically, this operator is performed in segment-1 of parents chromosome in order to exchange the whole segment between two parents. The rest of the segments in each parent are copied in it's respective child.

The third operator is an order crossover operator, which affects the sequence of tasks in different phases. Specifically, the order crossover operator exchanges segment-4 between two parents in phase one while the remaining segments in each parent are preserved in it's corresponding child. In phase two, one model is selected randomly; then, the order of tasks in the first occurrence of that model is exchanged between two parents. For example, model 1 is randomly chosen; then, segment-6 is exchanged between two parents to generate two new offsprings. In phase three, one model is randomly selected; then, one occurrence of that model is arbitrary chosen. Finally, the order of tasks in that occurrence is exchanged between two parents. The remaining segments in each parent are copied in the corresponding child. For instance, model 1 is randomly chosen; then, the second occurrence is arbitrary selected. Following that, segment-8 is exchanged between two parents.

The last operator is a model crossover operator, which is employed only in phase three if parents chromosomes have the same value of θ in segment-2. Specifically, one model is randomly selected in the chromosome of parents; then, the whole model is exchanged between two parents. The remaining segments in each parent will be retained in it's respective child.

A predefined crossover probability, which is equal for all crossover operators, is used to perform the above operators. Therefore, each crossover operator is applied based on the probability. In addition, a random number is generated in the range of [0,1] whenever the crossover operator is applied. Then, for performing

each crossover operator, the probability of that operator should be greater than the generated random number. Otherwise, that operator is not able to generate a new offspring.

4.1.7. Mutation operators

A mutation operator is a genetic operator that makes small changes in an individual of the population (Talbi, 2009). Specifically, this operator acts on the obtained chromosome from the crossover step to maintain population diversity (Sivanandam and Deepa, 2007; Akgunduz and Tunali, 2011). A crossover operator uses the current solution to create a better offspring in terms of fitness value while a mutation operator acts as a simple search operator, which is exploited for the whole search space exploration (Sivanandam and Deepa, 2007). The mutation process avoids the local minima's trap by preventing similar chromosomes in a population. Therefore, it randomly modifies some genes in the chromosome to maintain diversity (Sivanandam and Deepa, 2007; Akgunduz and Tunali, 2011). As with the crossover operator, the mutation operator should generate a valid solution (Talbi, 2009).

In this thesis, four mutation operators are presented. The first operator is a sequence swap mutation operator, which is performed in segment-1 of parents chromosome in all three phases. Specifically, this operator selects two random points on the chromosome of segment-1. Then, values of two genes corresponding to those random points are swapped. The obtained chromosome should satisfy demands of each model.

The second operator is an order mutation operator, which is exploited to make changes in the order of tasks in different phases. This operator affects segment-4 in phase one by selecting a random position in the chromosome of segment-4. This random position determines a gene that its value should be swapped with the value of another gene. Therefore, we move to the left and

right of the random position in order to find feasible locations (genes). Finding a feasible location is based on the satisfying the combined precedence diagram of models. After finding different locations, one location (gene) is randomly chosen and its value is swapped with the value of the first selected gene. The same procedure is applied in phase two but one model is randomly chosen at first; then, the above procedure is used to change the sequence of tasks in the first occurrence of that model. For example, model 3 is arbitrary selected; then, the above procedure is applied in segment-12 in occurrence 1 of model 3. Similarly, the defined procedure is used in phase three but after randomly selecting one model, one occurrence of that model is arbitrary chosen; then, the above procedure is used to change the order of tasks in that occurrence. For instance, model 1 is randomly selected. Then, occurrence 2 is arbitrary chosen. Finally, the order mutation operator is performed in segment-8.

The third operator is the number of tasks per station mutation operator, which is applied in three phases. In phase one, this mutation operator is performed in segment-3 in such a way that one position is arbitrary chosen. Then, a gene in the left side or right side of that position is randomly selected. Following that, a decision is made to randomly increase or decrease the value of the selected position. If an increasing value is preferred, the value of the gene, which is arbitrary chosen in the left or right side of that position is decreased, and vice versa. In addition, if the gene in each side of the selected position has zero value, another side of that position is selected to perform the above procedure. Moreover, if the selected position has zero value, its value will only be increased. The above procedure is used in phase two but at first, one model is arbitrary chosen. Then, the above procedure is applied to change the number of tasks per station in the first occurrence of that model. For example, model 3 is randomly selected; then, segment-11 in the first occurrence of model 3 is used to perform the above procedure. Similarly, the defined procedure is exploited in phase three

but after randomly selecting one model, one occurrence of that model is arbitrary chosen. Then, the above procedure is performed to alter the number of tasks per station in that occurrence. For instance, model 1 is randomly selected. Then, the second occurrence of model 1 is arbitrary chosen. Finally, the defined procedure is performed in segment-7.

Finally, the last operator is the number of opened stations mutation operator, which decreases the value of segment-2, θ, by one in three phases. By doing this, different segments will change in three phases. Decreasing θ affects segment-3 in phase one. Specifically, if θ decreases by one, one station will get zero value in the chromosome of segment-3. Therefore, tasks of that station are distributed equally between the rest of the opened workstations. This logic is exploited in the remaining phases. In phase two, the number of tasks of each model in each station will change if θ reduces by one. However, this matter occurs only in the first occurrence of models. Therefore, the above procedure is performed in segment-5, segment-9, and segment-11 if θ reduces. In phase three, decreasing θ affects all occurrences of each model. Therefore, the defined procedure is applied in segment-5, segment-7, segment-9, segment-11, and segment-13. One important point that should be considered in decreasing θ is to have a valid solution after performing the mutation operator. If decreasing θ leads to having an infeasible solution, a big penalty is applied for the first term of the objective function. This is caused to remove the infeasible solution in the rest of the process.

Like the crossover operator, each mutation operator is applied with a small pre-specified probability on different segments. In addition, a random number is generated in the range of [0,1] whenever the mutation operator is applied. Then, for performing each mutation operator, the probability of that operator should be greater than the generated random number. Otherwise, that operator is not able to be performed.

4.2. Flowchart of the hybrid genetic algorithm

Steps for applying the hybrid genetic algorithm, which is a combination of the genetic algorithm and linear programming algorithm, are illustrated in a flowchart in Figure 4.7. This flowchart shows that the binary variables are determined by using the genetic algorithm while the continuous variables are determined by solving the LP-subproblem corresponding to the values of binary variables. The defined steps have been coded in C++. An ILOG-CPLEX modelling environment is used to solve the linear programming model. Specifically, the CPLEX solver uses the simplex algorithm to solve the LP-subproblem. The notations which are used in the flowchart are as follows:

p	Population size
c	Index for a chromosome
g	Generator counter
$maxg$	Maximum number of generations
$Phase$	An indicator number that takes number 1, 2, and 3 based on three defined phases
H	Number of iterations which generate populations successfully without any improvement in the best fitness function value so far obtained
H_{max1}	Maximum number of H that leads to entering to the second phase if the previous Phase was equal to 1
H_{max2}	Maximum number of H in the second phase that leads to entering to the third phase if the previous Phase was equal to 2
H_{max3}	Maximum number of H in the third phase that leads to stopping the third phase

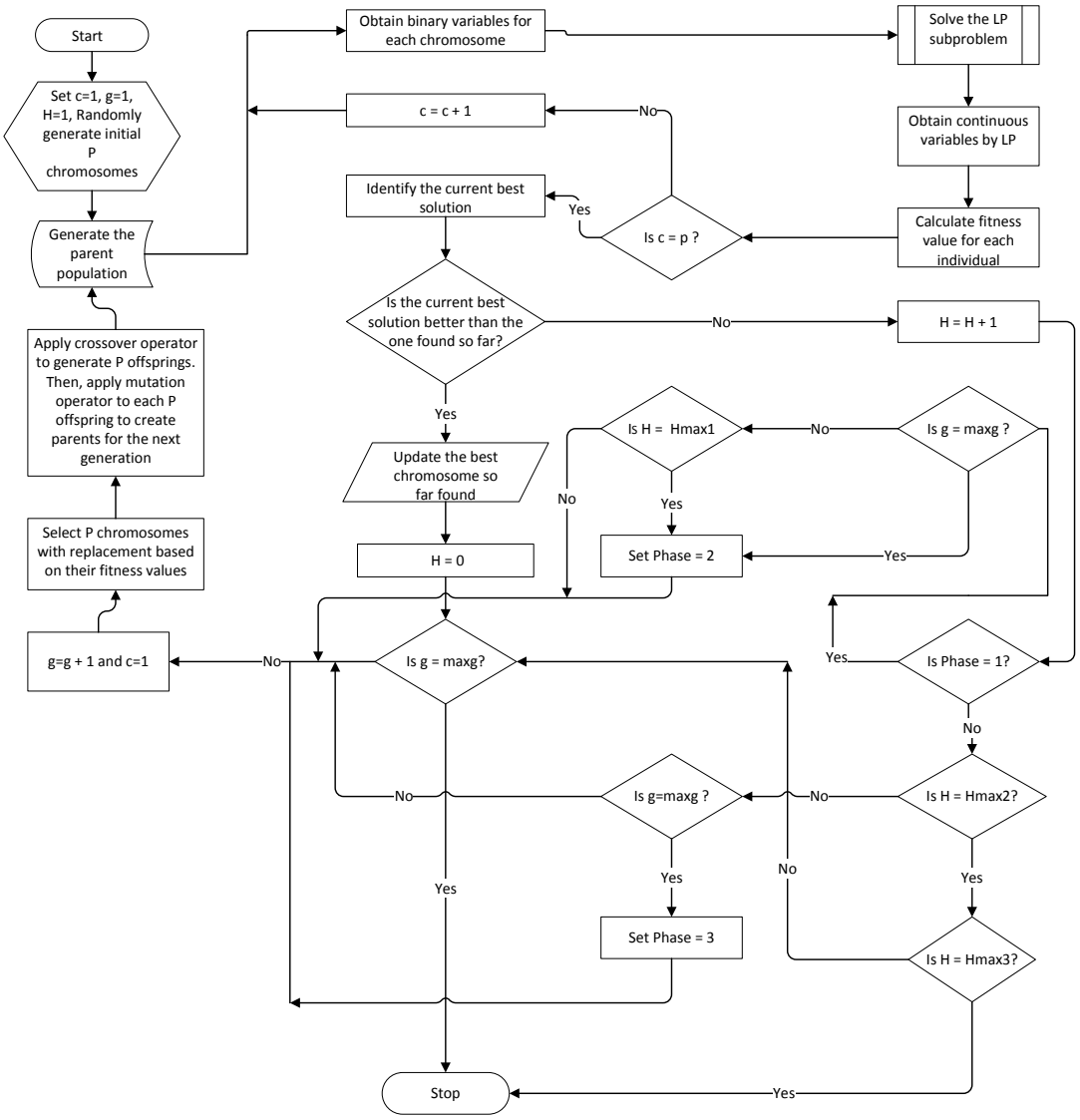


Figure 4.7: Flowchart of the hybrid genetic algorithm

Chapter 5

Numerical Example

5.1. Model illustration

In this section, one example is presented to explain and elaborate the behaviour of the proposed models in sections 3.2 and 3.3. Therefore, the following example shows how the proposed models act when the assembly line has a continuous motion as well as an intermittent motion. Then, the solutions which have been obtained from the branch and bound algorithm will be explained in detail.

5.1.1. Continuous motion

The example which is illustrated in this subsection addresses the proposed model in section 3.2. Inputs of our mathematical model are as follows in this example: total number of models of a product in an assembly line; demand of each model in the entire planning horizon and following that in the MPS; total number of tasks; total number of sequences; maximum number of workstations; speed of conveyor; launching rate; a precedence diagram, where each node represents a task and each arc represents a precedence relation; operation time of each task in each model; workstation cost; and finally, the task duplication cost. Assuming

that there are three models of a product A, B and C with demands of 100, 50 and 100 in the entire planning horizon respectively, the demand of each model in the MPS will be $de = (2, 1, 2)$. The launching rate value is equal to 6.39. The total number of sequences is equal to 5 based on the sum of demands in the MPS. The speed of conveyor is 1 unit distance per unit time and the workstation cost is equal to 200 cost units. Also, it is assumed that there are seven tasks in our example. Operation time (in time units) of each task in each model as well as task duplication cost (in cost units) have been given in Table 5.1. The precedence diagrams of three models are based on the diagrams which were illustrated in Figure 4.2. According to the precedence diagram, models have common tasks. The maximum number of workstations is equal to 7 in our example. Theoretically, the maximum number of workstations is equal to the total number of tasks, which are 7 in our example. But, we select the maximum number of workstations by trial and error to make sure that we will reach the optimum number of opened workstations in our result. With this way, at least one workstation will be closed before reaching the defined maximum number of workstations.

Table 5.1: Operation time of each task in each model and task duplication cost

	Tasks						
Models	1	2	3	4	5	6	7
A	3	6	0	3	4	0	4
B	4	6	5	2	0	1	6
C	5	0	4	6	3	5	2
Task duplication cost	10	11	14	12	9	15	10

We solve the above example by the branch and bound algorithm in order to compare two situations. In the first situation, common tasks are assigned to one workstation while in the second situation common tasks can be assigned to different workstations. With this comparison, we will clarify how the proposed model will improve the objective function value by assigning common tasks to different

workstations. Specifically, we compare these two situations in order to show the effectiveness of the proposed model in terms of different factors, which are objective function values, workstations length, the number of opened workstations in the assembly line, launching rate, the sequence of models, and distribution of tasks in different workstations. Unknown variables are determined by using the branch and bound algorithm. These variables are the length of each workstation, the number of opened workstations in the line, the starting position of the operator in each sequence for each workstation, the models' sequence and also the sequence of task assignment in each workstation. The objective function values, which are determined by the branch and bound algorithm, show that the proposed model gives us the smaller value by assigning common tasks to different workstations. Objective function values are 1033.61 for the first situation, which assigns common tasks to only one workstation, compared with 856.22 for the second situation, which assigns common tasks to different workstations. In addition, the number of workstations has decreased from 5 stations in the first situation to 4 stations in the second situation. Therefore, our proposed model has diminished the number of workstations. This leads to a decreased station cost too. Table 5.2 shows the length of each workstation and also the total number of workstations in both situations.

On the other hand, in the first situation, which we have forced performing each common task to only one workstation, there is no task duplication. Therefore, $a_{t,k}$ is equal to 1 for each task. Hence, we do not have any task duplication cost in this situation. Table 5.3 provides information about tasks' assignment to each workstation in the first situation. Each number inside the table determines the workstation to which the task of a specified model has been assigned. However, in the second situation, in which common tasks can be assigned to different stations, two common tasks have been assigned to more than one station. Table

5.4 provides more details about task's assignment to each workstation in the second situation. Each number inside the table determines the workstation to which the task of a specified model has been assigned. As is clear in Table 5.4, task 3 of model C has been assigned to station 1, and task 3 of model B has been assigned to station 3. In addition, task 4 of model C has been assigned to station 2, and task 4 of models A and B has been assigned to station 3. Therefore, tasks 3 and 4 have one duplication. Information about models' sequence in each situation is also provided in Table 5.3 and Table 5.4. In addition, Figure 5.1 and Figure 5.2 illustrate the information of Table 5.3 and Table 5.4 respectively in two graphs.

Table 5.2: Length of each workstatin for two types of situations

	Number of workstations						
Type of situation	1	2	3	4	5	6	7
Assigning common tasks to one station	5	11	6	4	7.61	0	0
Assigning common tasks to different stations	9	6	7.61	7.61	0	0	0

Table 5.3: Task's assignment of each model to each workstation for the first situation

	Tasks						
Model's Sequence	1	2	3	4	5	6	7
A_1	1	2	-	3	4	-	5
B_1	1	2	2	3	-	5	5
A_2	1	2	-	3	4	-	5
C_1	1	-	2	3	4	5	5
C_2	1	-	2	3	4	5	5

One of the main factors in the proposed mathematical model is the launching rate value. Results show that if we decrease launching rate, the number of workstations will be increased in the situation, in which common tasks can be

assigned to different stations. In addition, the objective function value will go up significantly and common tasks will be distributed in more stations. Similarly, decreasing launching rate leads to increase the number of workstations as well as the objective function value in the situation, in which common tasks are assigned to one station. However, there are some values of launching rate that the proposed mathematical model does not have any feasible solutions if common tasks are assigned to only one workstation. This lack of feasible solution is because the design principle that the total operation time in every 5 sequences in each workstation should not exceed launching rate multiplied by the total number of sequences. According to this principle, there are some cases, in which launching rate satisfies the proposed mathematical model by assigning common tasks to different stations, while the launching rate dose not satisfy the proposed mathematical model if common tasks are assigned to only one station.

Table 5.4: Task's assignment of each model to each workstation for the second situation

	Tasks						
Model's Sequence	1	2	3	4	5	6	7
A_1	1	2	-	3	3	-	4
C_1	1	-	1	2	3	4	4
B_1	1	2	3	3	-	4	4
A_2	1	2	-	3	3	-	4
C_2	1	-	1	2	3	4	4

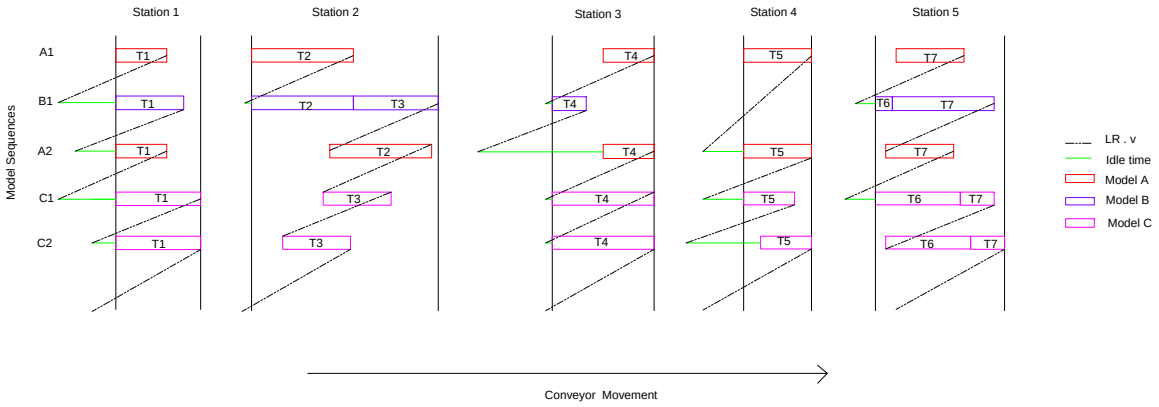


Figure 5.1: Task's assignment and model's sequence based on Table 5.3

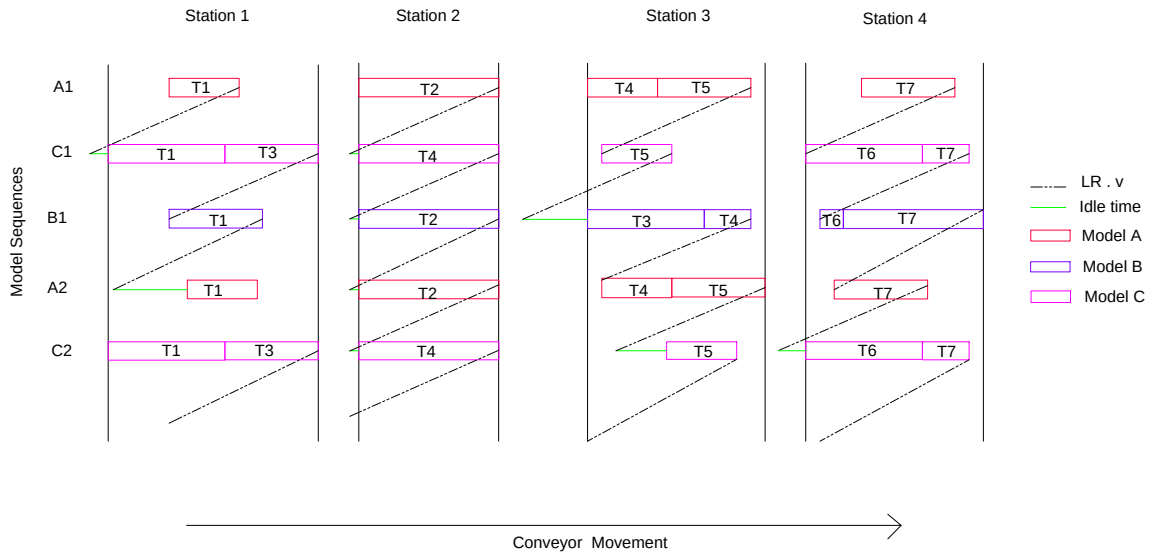


Figure 5.2: Task's assignment and model's sequence based on Table 5.4

Table 5.5 shows whether or not the proposed mathematical model has a feasible solution for different values of launching rate in two situations. F shows that the mathematical model has a feasible solution, while Inf shows that the mathematical model does not have any feasible solution. As Table 5.5 highlights, the proposed mathematical model does not have any feasible solution when the

launching rate value is equal to or less than 3.39. However, if the launching rate value goes up, the mathematical model can satisfy the situation that common tasks are assigned to different workstations while it does not have any feasible solution for the situation that common tasks are forced to be assigned to only one workstation. This matter is true for launching rate values between 3.40 and 3.99. Therefore, according to this result, our proposed mathematical model can satisfy demands of different models of a product by assigning common tasks to different stations for launching rate values between 3.40 and 3.99, while our mathematical model does not satisfy demands when common tasks are assigned to only one workstation.

Table 5.5: Status of obtained solution from the branch and bound algorithm for two types of situations

	Launching rates					
type of situation	3	3.20	3.40	3.60	3.80	4
Assigning common tasks to one station	Inf	Inf	Inf	Inf	Inf	F
Assigning common tasks to different stations	Inf	Inf	F	F	F	F

The next factor that affects the proposed mathematical model is conveyor speed. We assumed that a conveyor moves along workstations with a constant speed. Therefore, a conveyor moves work pieces steadily from station to station. Now, we increase the conveyor speed in our mathematical model and solve the model by the branch and bound algorithm. Results show that workstations will have larger length if the conveyor speed goes up by more than 1. However, the number of stations will not change by increasing the conveyor speed.

5.1.2. Intermittent motion

The numerical example, which is used in this subsection, is the same as that presented in the previous subsection to address the proposed model in section 3.3. Also, the same precedence diagrams are depicted here. The obtained objective function value from branch and bound algorithm is equal to the 1032 with 5 opened workstations in the line. Also, Table 5.6 provides information about the task's assignment to each workstation as well as the sequence of models in the synchronous line. Each number inside the table determines the workstation to which the task of a specified model has been assigned. As is clear in Table 5.6, three common tasks, which are tasks 2, 4, and 5, have been assigned to more than one station. Specifically, task 2 has been assigned to stations 2 and 3, task 4 has been assigned to stations 3 and 4, and task 5 has been assigned to stations 4 and 5. Therefore, these three tasks have one duplication.

Table 5.6: Task's assignment of each model to each workstation for the synchronous line

	Tasks						
Model's Sequence	1	2	3	4	5	6	7
A_1	1	2	-	3	4	-	5
C_1	1	-	2	3	5	4	5
C_2	1	-	2	3	5	4	5
B_1	1	3	2	4	-	4	5
A_2	1	2	-	3	4	-	5

To summarise, it has been shown that the proposed mathematical model in section 3.2 can satisfy the continuous assembly line as well as the synchronous assembly line only by changing some assumptions.

5.2. Branch and bound algorithm versus hybrid genetic algorithm

In this section, several examples with different sizes are presented to show the advantage of the proposed hybrid genetic algorithm (HGA) in comparison with the branch and bound (BB) algorithm. The first example, which is solved in two versions, is a small example with three models and four tasks. In the first version, demands of models in the entire planning horizon are defined as follows: 2 units for model A, 1 unit for model B, and 2 units for model C. Therefore, there are 5 sequences in this version. The launching rate value is equal to 20. The maximum number of stations is equal to 6 and the station cost is equal to 50 cost units. The speed of conveyor is equal to 1. Table 5.7 provides information about the operation time (in time units) of each task in each model as well as the task duplication cost (in cost units). In addition, Figure 5.3 illustrates the precedence diagram of each model.

Table 5.7: Operation time of each task in each model and task duplication cost, first example

	Tasks			
Models	1	2	3	4
A	14	22	14	11
B	25	16	21	14
C	11	8	20	17
Task duplication cost	10	9	7	12

This example is solved once by the BB algorithm and once by the HGA to compare their convergences behaviour. The $CPLEX$ solver is used to solve the BB algorithm. Figure 5.4 shows the convergence behaviour of the BB algorithm while Figure 5.5 illustrates the convergence behaviour of the HGA. 500 population and three generations were used to obtain the convergence graph of HGA. As has been shown in two figures, an improvement in the objective function value

is obtained as the time is increased. The best objective function value is 4169. The BB algorithm obtained this value in one minute while the value of the lower bound was 4021. However, the BB algorithm continued to solve the example and declared that the obtained value is an optimal value at t= 0:02:23. In contrast, the HGA could reach the optimal value of 4169 only in 7 seconds. Therefore, our proposed HGA could solve this example with much less computational time compared with the BB algorithm.

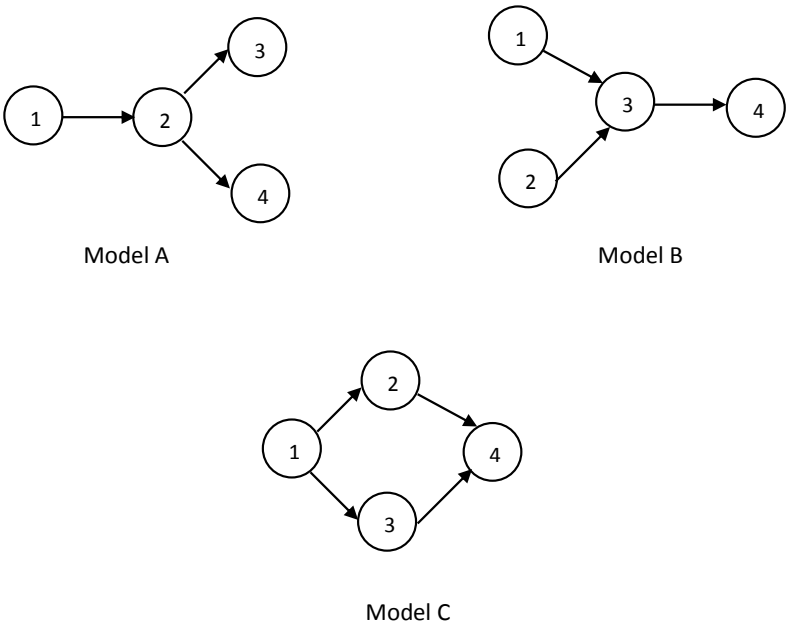


Figure 5.3: Precedence diagram for models A, B and C

In the second version, the number of demands for each model will be increased because increasing demands leads to having more variables that make our example more complicated. Therefore, the second version of the example is solved by following demands: 3 units for model A, 2 units for model B, and 5 units for model C. Therefore, the total number of sequences is equal to 10 in this version. The rest of the data are similar to the first version of the example. Figure 5.6

and Figure 5.7 illustrate the convergence graphs which have been obtained from the BB algorithm and HGA respectively. The best objective function value is still 4169. The BB algorithm reached to this value in one minute while the value of the lower bound was 4000. However, the BB algorithm continued to solve the example for three hours but it only could improve the lower bound to 4069. On the other hand, the HGA could reach the optimal value of 4169 only in 12 seconds. Therefore, our proposed HGA could also solve this version of the example with much less computational time in comparison with the BB algorithm.

Hence, the above example shows the correctness of the proposed HGA. In addition, this example illustrates that the solution representation does not exclude the optimal solution, which means that the proposed HGA is potentially able to find optimal solutions for larger problems as well.

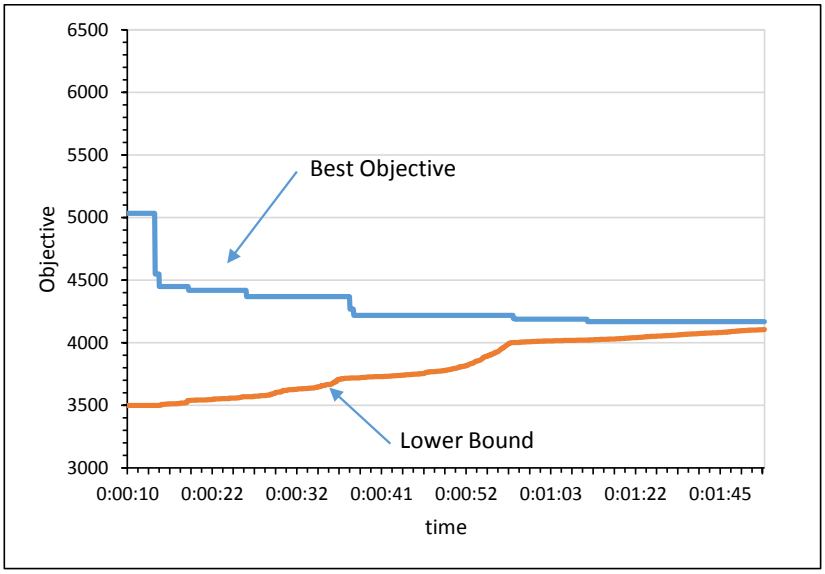


Figure 5.4: Convergence graph for the BB algorithm, first version of the first example

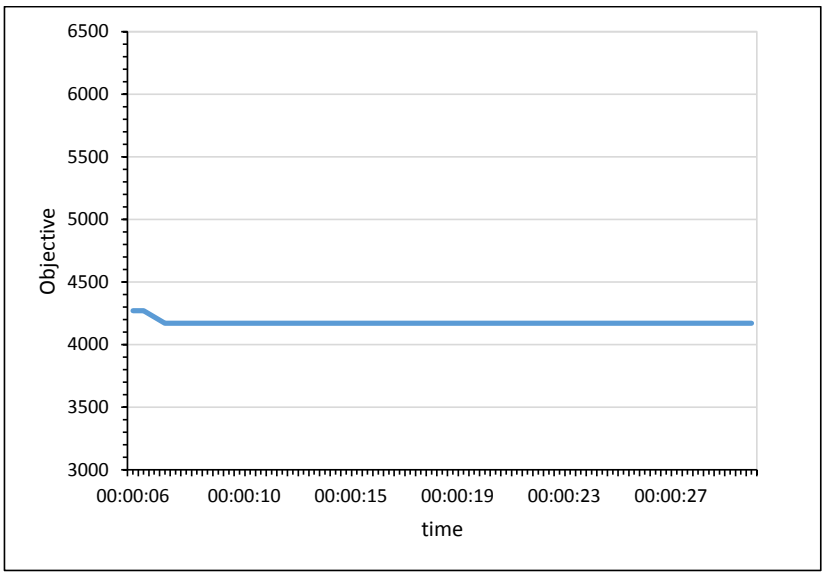


Figure 5.5: Convergence graph for the *HGA*, first version of the first example

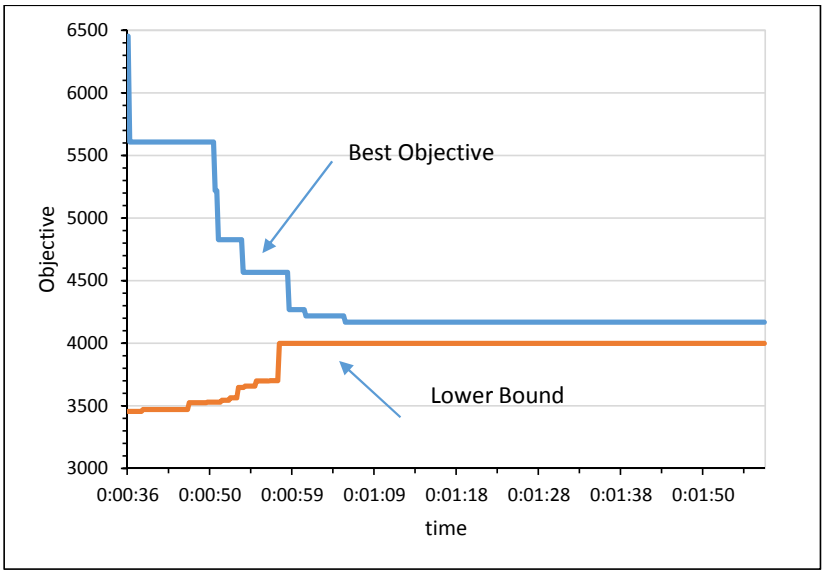


Figure 5.6: Convergence graph for the *BB* algorithm, second version of the first example

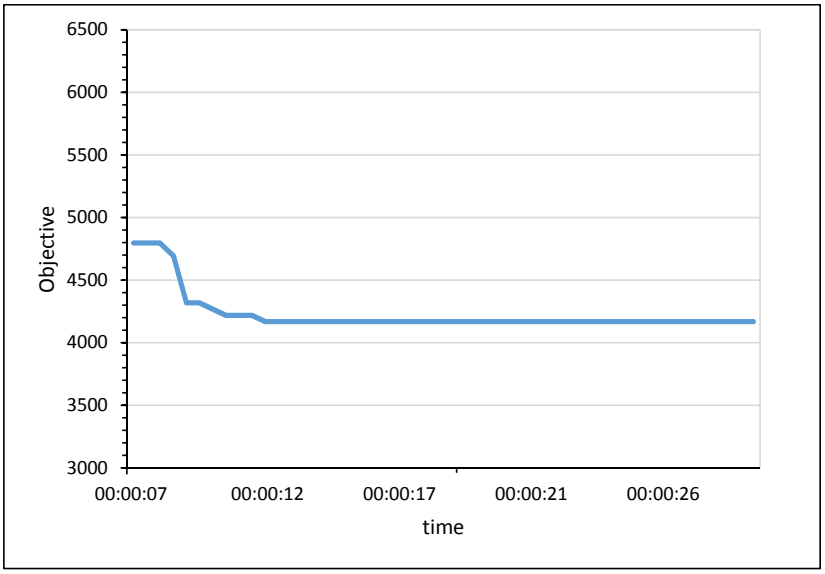
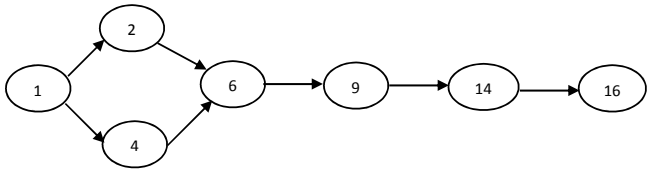


Figure 5.7: Convergence graph for the HGA, second version of the first example

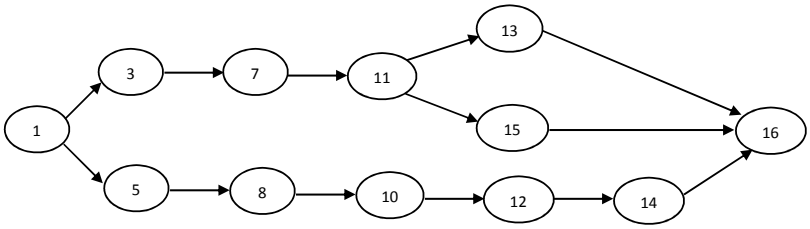
In the next example, the number of tasks is four times more than the number of tasks in the first example. Therefore, 16 tasks are defined for the second example while we still have 3 models. Similar to the first example, the second example is solved in two versions with different demands. Demands of models in the entire planning horizon are defined as follows in the first version of the second example: 2 units for model A, 1 unit for model B, and 2 units for model C. Therefore, there are 5 sequences in this example. The launching rate value is equal to 6. The maximum number of stations is equal to 10 and the station cost is equal to 500 cost units. The speed of conveyor is equal to 1. Table 5.8 provides information about operation time (in time units) of each task in each model as well as tasks duplication cost (in cost units). In addition, Figure 5.8 illustrates the precedence diagram of each model. Figure 5.9 and Figure 5.10 illustrate the convergence graphs which have been obtained from the BB algorithm and HGA respectively.

Table 5.8: Operation time of each task in each model and task duplication cost, second example

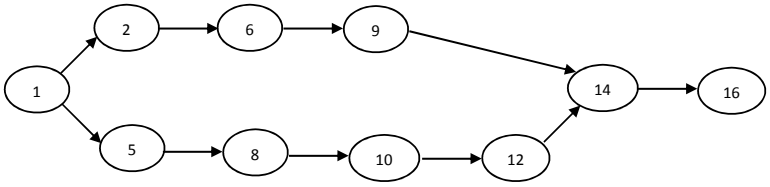
	Tasks															
Models	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	2	2	0	3	0	3	0	0	2	0	0	0	0	3	0	2
B	1	0	2	0	2	0	4	3	0	2	3	3	5	3	4	3
C	1	3	0	0	4	3	0	1	3	4	0	4	0	2	0	1
Task duplication cost	10	11	14	12	9	15	10	9	9	10	8	13	14	11	20	10



Model A



Model B



Model C

Figure 5.8: Precedence diagram for models A, B and C

As figures 5.9 and 5.10 show, the best objective function value is 4250. The

BB algorithm obtained this value in about three minutes. In contrast, the HGA could reach the optimal value of 4250 in only 40 seconds. In the second version of the second example, the demand of each model is defined as follows: 2 units for model A, 3 units for model B, and 5 units for model C. Therefore, the total number of sequences is equal to 10. The remaining data are similar to the first version of this example. Figure 5.11 and Figure 5.12 illustrate the convergence graphs which have been obtained from the BB algorithm and HGA respectively. As two convergence graphs illustrate, the best objective value is equal to 4359. The BB algorithm reached the optimal solution after 1 hour and 40 minutes while the time required to get the optimal value in the HGA was only doubled. Therefore, although the total number of variables and constraints are increased by nearly 100 percent in this version of the example compared with the first version, the proposed HGA could solve this version with consuming shorter time compared with the BB algorithm. As with the first example, the second example showed that our proposed HGA can solve two versions of example with much less computational time in comparison with the BB algorithm.

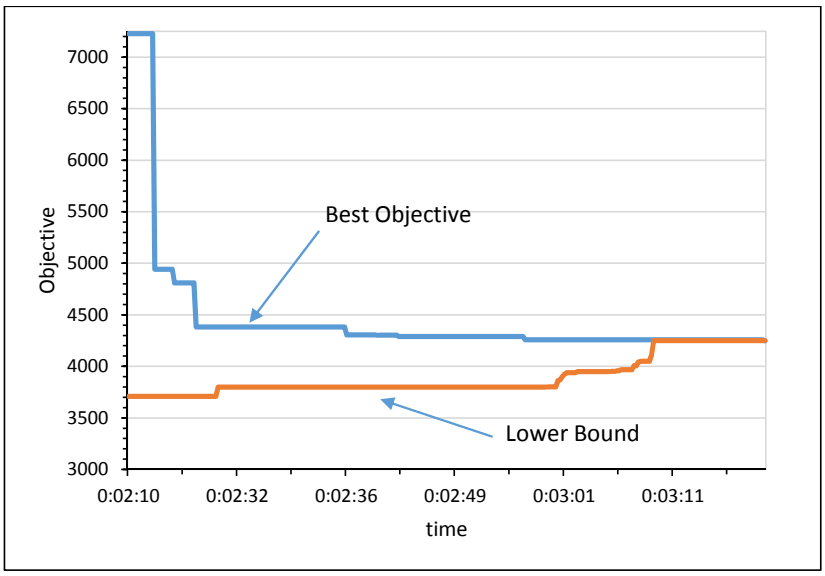


Figure 5.9: Convergence graph for the BB algorithm, first version of the second example

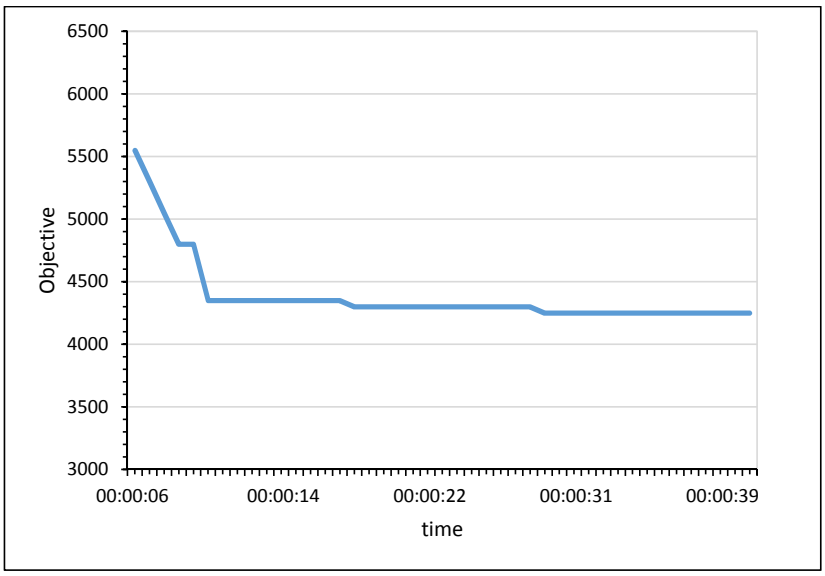


Figure 5.10: Convergence graph for the HGA, first version of the second example

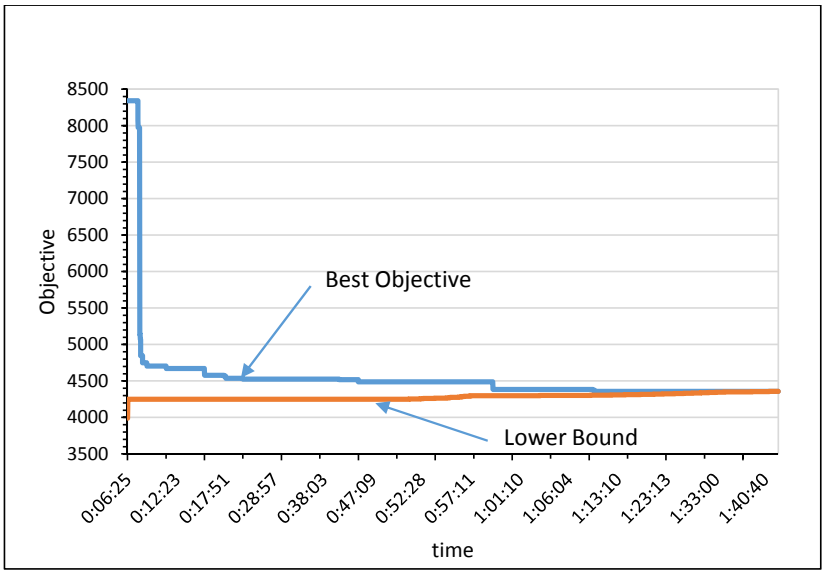


Figure 5.11: Convergence graph for the BB algorithm, second version of the second example

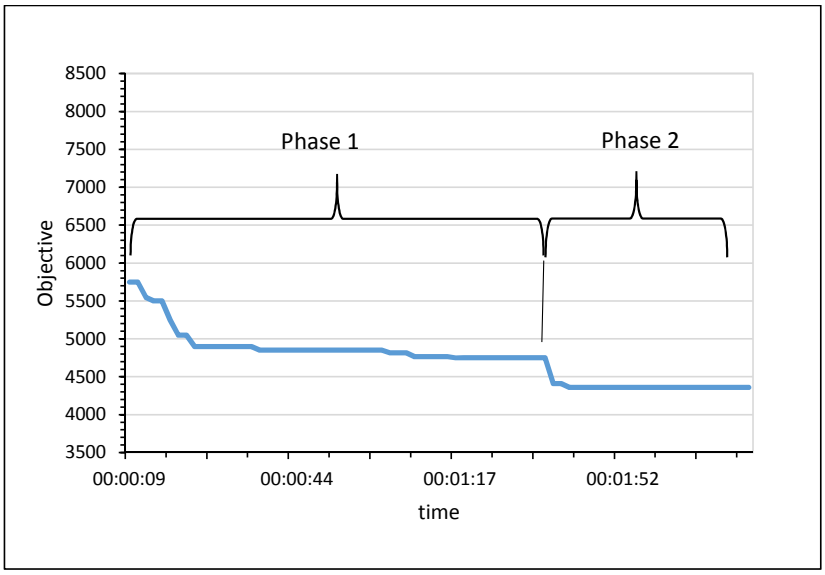


Figure 5.12: Convergence graph for the *HGA*, second version of the second example

Finally, in the last example, the number of tasks is significantly more than the number of tasks in the second example. Therefore, 33 tasks are defined for the third example while we still have 3 models. This example is solved in two versions with different demands. In the first version of this example, we have defined one unit demand for each model. Therefore, there are 3 sequences in this version. Table 5.9 provides information about operation time (in time units) of each task in each model as well as tasks duplication cost (in cost units). In addition, Figure 5.13, Figure 5.14, and Figure 5.15 illustrate the precedence diagrams of models. Figure 5.16 and Figure 5.17 show the convergence graphs which have been obtained from the *BB* algorithm and *HGA* respectively.

Table 5.9: Operation time of each task in each model and task duplication cost, third example

	Tasks																																
Models	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
A	2	3	2	2	2	0	3	4	2	3	0	2	3	4	2	2	3	2	1	1	2	0	3	5	3	2	3	2	2	2	1	2	4
B	3	2	3	2	2	1	0	3	2	1	2	3	2	2	0	1	2	3	2	1	2	0	2	2	3	1	1	4	5	2	3	2	1
C	2	2	2	2	2	4	5	3	3	0	3	3	2	2	2	1	1	1	1	3	2	3	2	3	2	0	2	4	3	2	2	1	5
Task duplication cost	10	11	14	12	9	15	10	9	9	7	15	13	10	14	21	22	19	14	13	10	20	13	12	11	7	8	9	15	12	15	16	18	20

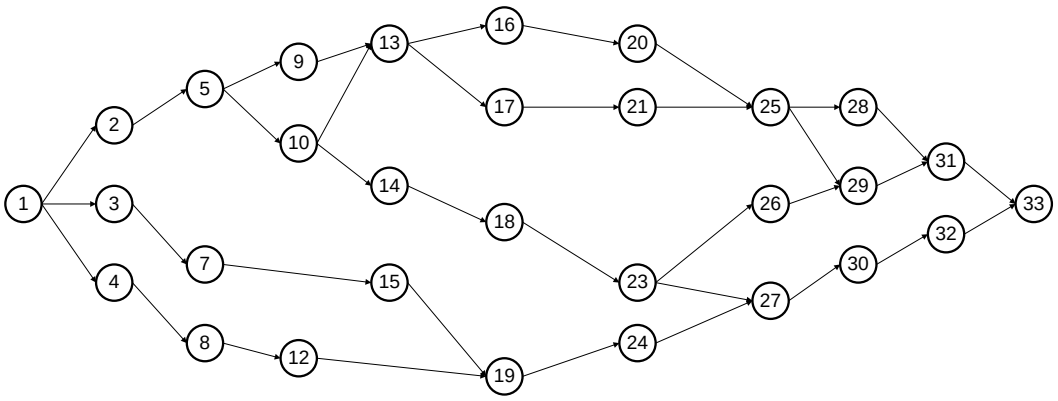


Figure 5.13: Precedence diagram for model A, third example

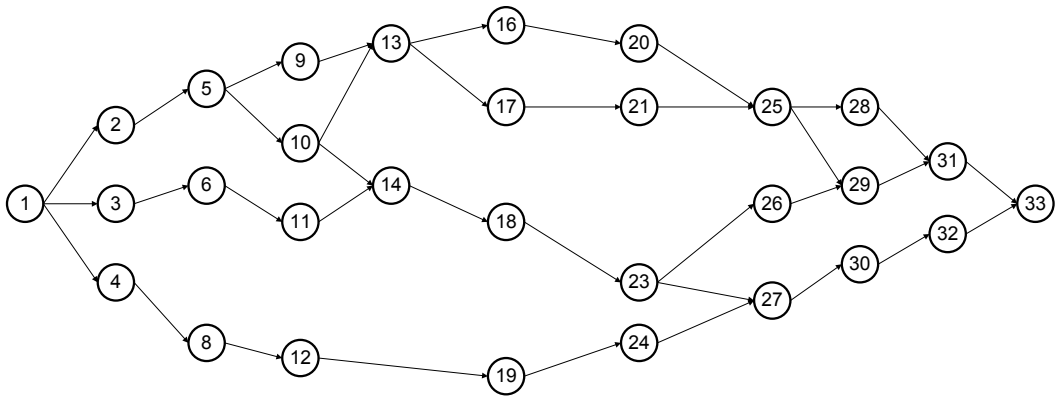


Figure 5.14: Precedence diagram for model B, third example

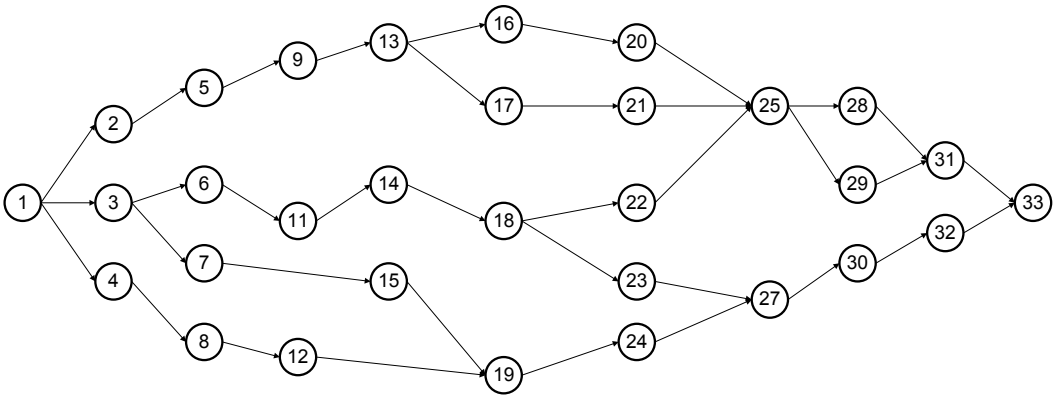


Figure 5.15: Precedence diagram for model C, third example

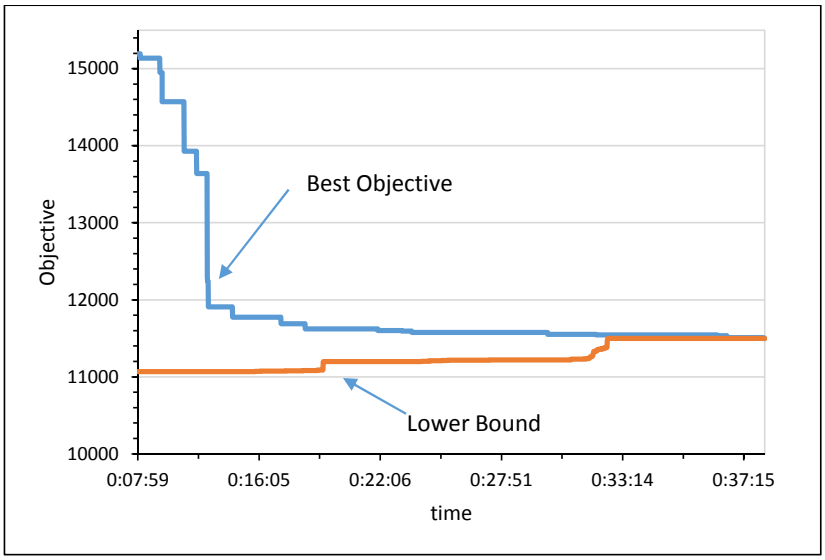


Figure 5.16: Convergence graph for the BB algorithm, first version of the third example

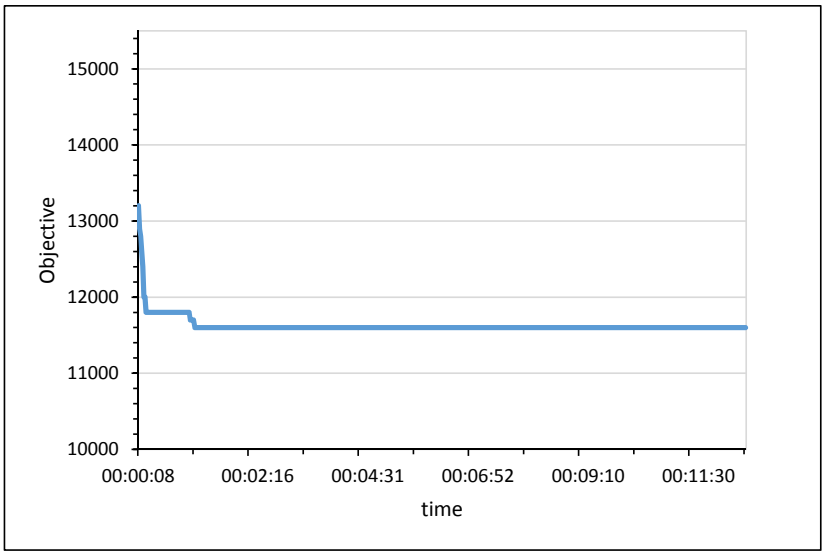


Figure 5.17: Convergence graph for the HGA, first version of the third example

As figures 5.16 and 5.17 show, the BB algorithm could obtain the first feasible solution, which is equal to 15195, in 8 minutes while the HGA could find the better solution in a shorter time, 13200 in only 8 seconds. Then, the BB algorithm continued to solve the example and after 38 minutes it obtained the optimal value, which is equal to 11500. In contrast, the HGA obtained the value of 11600 in only 1 minutes. Therefore, although the HGA did not obtain the optimal value, it could reach the first feasible solution and also the acceptable good value of objective in much less time compared with the BB algorithm.

In the second version of this example, demand of each model was increased as follows: 3 units for model A, 2 units for model B, and 5 units for model C. Figure 5.18 and Figure 5.19 illustrate the convergence graphs for the BB algorithm and HGA respectively for this version. The BB algorithm could obtain the first feasible solution, which is equal to 13337, in approximately 47 hours while our proposed algorithm reached the first feasible solution, which is equal to 13300, in only 8 seconds. The BB algorithm continued to solve the problem and reached

the best objective, which is equal to 11557, in t = 104:57:25 and finally, after 115 hours declared that this value is the optimal value. In contrast, the proposed HGA could obtain the acceptable good solution, which is equal to 11827, in only 5 minutes.

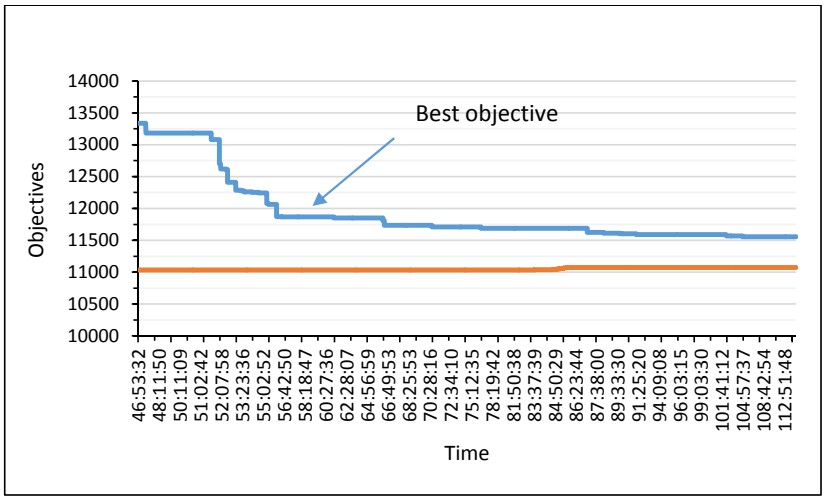


Figure 5.18: Convergence graph for the BB algorithm, second version of the third example

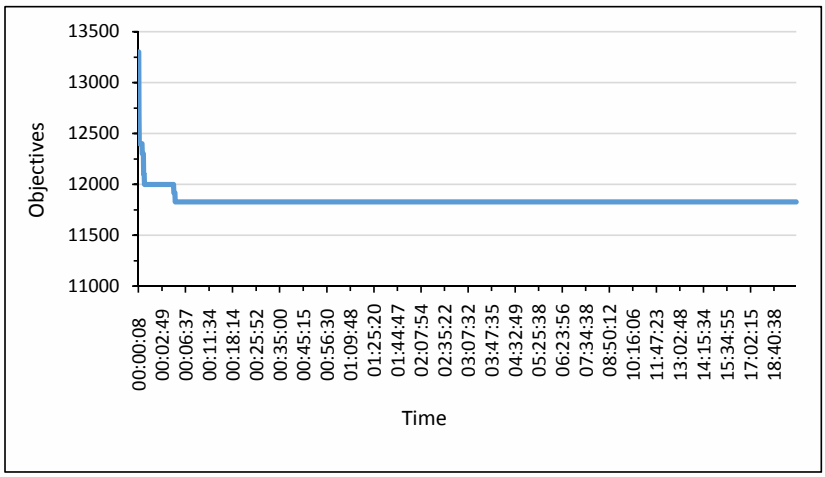


Figure 5.19: Convergence graph for the HGA, second version of the third example

To sum up, two versions of the last example showed that our proposed HGA could reach the first feasible solution in a shorter amount of time compared with the BB algorithm. In addition, this solution was significantly better in the HGA compared with the BB algorithm. Moreover, the HGA could obtain an acceptable good solution, which was near to the optimal solution, in much less time compared with the BB algorithm. Finally, it has been demonstrated that the proposed algorithm is able to solve larger problems, because the third example was significantly lager than the first two examples. Therefore, it can be declared that the proposed HGA can solve problems even larger than the third example.

Chapter 6

Research Outline

6.1. Conclusion

In recent years, the mixed model assembly line has attracted more attention in the manufacturing environment than the single model assembly line because this type of the assembly line enables companies to satisfy different demands of customers by producing several models of a product. Two indispensable challenges in the mixed model assembly line are balancing problem and sequencing problem. In this thesis, we developed a mixed integer linear programming model to solve balancing and sequencing problems simultaneously in the mixed model assembly line. In the developed mathematical model, we assumed that the assembly line has a continuous motion, and common tasks between various models of a product can be assigned to different workstations. The main objectives of this model are minimizing the workstations length, minimizing the workstations cost, and minimizing the tasks duplication cost. The branch and bound algorithm was used to solve the proposed model in two types of situations: assigning common tasks to different workstations and assigning common tasks to only one workstation. The results showed that the objective function decreases by assigning common tasks to different workstations.

The next result was that changing the launching rate can greatly affect the proposed mathematical model for satisfying demands. Specifically, our mathematical model did not have any feasible solutions for some values of launching rate if common tasks were assigned to only one workstation, but this model had a feasible solution for those values of launching rate when common tasks were assigned to different workstations. Therefore, the proposed model satisfied demands of different models if common tasks were assigned to different workstations while this matter was not possible for some values of launching rates if common tasks are assigned to only one workstation .

At the next step, we changed the conveyor motion from a continuous motion to an intermittent motion. With this approach, we extended the proposed mathematical model with the assumption that the mixed model assembly line has a synchronous configuration. The extended model was solved by the branch and bound algorithm; results showed that our proposed mathematical model satisfies the continuous motion as well as the intermittent motion in the mixed model assembly line.

Finally, a hybrid genetic algorithm, which was a combination of a genetic algorithm and a linear programming algorithm, was employed to solve the proposed mathematical model for large size problems. In the genetic algorithm, a solution representation was introduced in three phases to determine the models' sequence as well as tasks' assignment to different workstations. Therefore, the binary variables were obtained by solving the genetic algorithm. On the other hand, the linear programming subproblem was solved to obtain the continuous variables corresponding to the values of binary variables. To show the effectiveness of the proposed hybrid algorithm, some numerical examples were solved, and results were compared with the branch and bound algorithm. Based on the obtained results, our proposed HGA outperformed the BB algorithm.

6.2. Future research

Future research directions are suggested in this section as follows: first, other objectives can be added to the objectives of this thesis such as to minimize equipment cost for common tasks in different workstations. The equipment cost is resulted from using common equipment in different stations based on the common tasks assigned to those workstations. Second, the buffer allocation, which has not been addressed in this thesis, can be of interest in future studies. Buffers are used between some workstations to prevent starving or blocking. Third, changing the type of operation time from deterministic, which has been considered in this study, to stochastic can open another window for future study. Fourth, exploiting parallel workstations and zoning constraints would be interesting. Finally, other hybrid metaheuristic approaches can be employed to solve large problems such as combining a simulated annealing algorithm with a linear programming algorithm. Then, results can be compared with the proposed hybrid algorithm for further analysis.

Bibliography

Aase, G. R., Olson, J. R., and Schniederjans, M. J., 2004. U-shaped assembly line layouts and their impact on labor productivity: An experimental study. *European Journal of Operational Research*, **156 (3)**, 698–711.

Abdullah, T., Popplewell, K., and Page, C., 2003. A review of the support tools for the process of assembly method selection and assembly planning. *International Journal of Production Research*, **41 (11)**, 2391–2410.

Akgunduz, O. S. and Tunali, S., 2010. An adaptive genetic algorithm approach for the mixed-model assembly line sequencing problem. *International Journal of Production Research*, **48 (17)**, 5157–5179.

Akgunduz, O. S. and Tunali, S., 2011. A review of the current applications of genetic algorithms in mixed-model assembly line sequencing. *International Journal of Production Research*, **49 (15)**, 4483–4503.

Akpinar, S. and 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.

Andreasen, M., Kahler, S., and Lund, T., 1983. Design for assembly, IFS publication, Ltd., UK. Tech. rep., ISBN 0-903608-35-9.

Bagher, M., Zandieh, M., and Farsijani, H., 2011. Balancing of stochastic u-type assembly lines: an imperialist competitive algorithm. *The International Journal of Advanced Manufacturing Technology*, **54 (1-4)**, 271–285.

Bard, J., Shtub, A., and Joshi, S., 1994. Sequencing mixed-model assembly lines to level parts usage and minimize line length. *The International Journal of Production Research*, **32 (10)**, 2431–2454.

Bartholdi, J., 1993. Balancing two-sided assembly lines: A case study. *The International Journal of Production Research*, **31 (10)**, 2447–2461.

Battaia, O. and Dolgui, A., 2013. A taxonomy of line balancing problems and their solutionapproaches. *International Journal of Production Economics*, **142 (2)**, 259–277.

Baybars, I., 1986. A survey of exact algorithms for the simple assembly line balancing problem. *Management science*, **32 (8)**, 909–932.

Becker, C. and Scholl, A., 2006. A survey on problems and methods in generalized assembly line balancing. *European journal of operational research*, **168 (3)**, 694–715.

Bi, Z. and Zhang, W., 2001. Flexible fixture design and automation: review, issues and future directions. *International Journal of Production Research*, **39 (13)**, 2867–2894.

Blum, C. and Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, **35 (3)**, 268–308.

Boysen, N., Fliedner, M., and Scholl, A., 2007. A classification of assembly line balancing problems. *European Journal of Operational Research*, **183 (2)**, 674–693.

Boysen, N., Fliedner, M., and Scholl, A., 2008. Assembly line balancing: Which model to use when? *International Journal of Production Economics*, **111 (2)**, 509–528.

Boysen, N., Fliedner, M., and Scholl, A., 2009a. Level scheduling of mixed-model assembly lines under storage constraints. *International Journal of Production Research*, **47 (10)**, 2669–2684.

Boysen, N., Fliedner, M., and Scholl, A., 2009b. Sequencing mixed-model assembly lines: survey, classification and model critique. *European Journal of Operational Research*, **192 (2)**, 349–373.

Bukchin, J., 1998. A comparative study of performance measures for throughput of a mixed model assembly line in a JIT environment. *International Journal of Production Research*, **36 (10)**, 2669–2685.

Bukchin, Y. and 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 (1)**, 492–508.

Carnahan, B. J., Norman, B. A., and Redfern, M. S., 2001. Incorporating physical demand criteria into assembly line balancing. *Iie Transactions*, **33 (10)**, 875–887.

Caruso, F. R., 1965. Assembly line balancing for improved profits. *Automation*, **12 (1)**, 48–52.

Cercioglu, H., Ozcan, U., Gokcen, H., and Toklu, B., 2009. A simulated annealing approach for parallel assembly line balancing problem. *Journal of the Faculty of Engineering and Architecture of Gazi University*, **24 (2)**, 331–341.

Chow, W. M., 1990. Assembly line design: methodology and applications. Vol. 33. CRC Press,

Chutima, P. and Chimklai, P., 2012. Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge. *Computers and Industrial Engineering*, **62 (1)**, 39–55.

Dar-El, E. and Cucuy, S., 1977. Optimal mixed-model sequencing for balanced assembly lines. *Omega*, **5 (3)**, 333–342.

Ding, F. Y., Zhu, J., and Sun, H., 2006. Comparing two weighted approaches for sequencing mixed-model assembly lines with multiple objectives. *International Journal of Production Economics*, **102 (1)**, 108–131.

Drexl, A. and Kimms, A., 2001. Sequencing JIT mixed-model assembly lines under station-load and part-usage constraints. *Management Science*, **47 (3)**, 480–491.

Drexl, A., Kimms, A., and Matthießen, L., 2006. Algorithms for the car sequencing and the level scheduling problem. *Journal of Scheduling*, **9 (2)**, 153–176.

Erel, E., Sabuncuoglu, I., and Aksu, B., 2001. Balancing of u-type assembly systems using simulated annealing. *International Journal of Production Research*, **39 (13)**, 3003–3015.

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

Essafi, M., Delorme, X., and Dolgui, A., 2012. A reactive grasp and path relinking for balancing reconfigurable transfer lines. *International Journal of Production Research*, **50 (18)**, 5213–5238.

Faccio, M., Gamberi, M., and Bortolini, M., 2015. Hierarchical approach for paced mixed-model assembly line balancing and sequencing with jolly operators. *International Journal of Production Research*, , 1–17.

Fattahi, P. and Salehi, M., 2009. Sequencing the mixed-model assembly line to minimize the total utility and idle costs with variable launching interval. *The International Journal of Advanced Manufacturing Technology*, **45 (9-10)**, 987–998.

Gen, M. and Cheng, R., 1997. Genetic algorithms and engineering design, 1997. *John Wily and Sons, New York*, .

Gendreau, M. and Potvin, J. Y., 2010. Handbook of metaheuristics. Vol. 2. Springer,

Ghosh, S. and Gagnon, R. J., 1989. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *The International Journal of Production Research*, **27 (4)**, 637–670.

Goldberg, D. E. and Holland, J. H., 1988. Genetic algorithms and machine learning. *Machine learning*, **3 (2)**, 95–99.

Graves, S. C. and Redfield, C. H., 1988. Equipment selection and task assignment for multiproduct assembly system design. *International Journal of Flexible Manufacturing Systems*, **1 (1)**, 31–50.

Groover, M. P., 2007. Automation, production systems, and computer-integrated manufacturing. Prentice Hall Press,

Guschinskaya, O., Gurevsky, E., Dolgui, A., and Eremeev, A., 2011. Metaheuristic approaches for the design of machining lines. *The International Journal of Advanced Manufacturing Technology*, **55 (1-4)**, 11–22.

Haq, A. N., Rengarajan, K., and Jayaprakash, J., 2006. A hybrid genetic algorithm approach to mixed-model assembly line balancing. *The International Journal of Advanced Manufacturing Technology*, **28 (3-4)**, 337–341.

Ho, J., 2005. A proposed approach for reconfiguration of flexible assembly line systems by motion genes. *International Journal of Production Research*, **43 (9)**, 1729–1749.

Holland, J., 1975. Adaptation in artificial and natural systems. *Ann Arbor: The University of Michigan Press*, .

Hwang, R. and Katayama, 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.

Hyun, C. J., Kim, Y., and Kim, Y. K., 1998. A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines. *Computers and Operations Research*, **25 (7)**, 675–690.

Johnson, R. V., 1983. A branch and bound algorithm for assembly line balancing problems with formulation irregularities. *Management Science*, **29 (11)**, 1309–1324.

Kalayci, C. B. and Gupta, S. M., 2014. A tabu search algorithm for balancing a sequence-dependent disassembly line. *Production Planning and Control*, **25 (2)**, 149–160.

Kalayci, C. B., Polat, O., and Gupta, S. M., 2014. A hybrid genetic algorithm for sequence-dependent disassembly line balancing problem. *Annals of Operations Research*, , 1–34.

Kim, S. and Jeong, B., 2007. Product sequencing problem in mixed-model assembly line to minimize unfinished works. *Computers and Industrial Engineering*, **53 (2)**, 206–214.

Kim, Y. K., Kim, J. Y., and Kim, Y., 2000. A coevolutionary algorithm for balancing and sequencing in mixed model assembly lines. *Applied Intelligence*, **13 (3)**, 247–258.

Kim, Y. K., Kim, J. Y., and Kim, Y., 2006. An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model u-lines. *European Journal of Operational Research*, **168 (3)**, 838–852.

Kottas, J. F. and Lau, H. S., 1973. A cost-oriented approach to stochastic line balancing1. *AIIE Transactions*, **5 (2)**, 164–171.

Kouvelis, P. and Karabati, S., 1999. Cyclic scheduling in synchronous production lines. *IIE transactions*, **31 (8)**, 709–719.

Kriengkorakot, N. and Pianthong, N., 2007. The assembly line balancing problem. *KKU Enginieering Journal*, **34 (2)**, 133–140.

Malakooti, B. and Kumar, A., 1996. A knowledge-based system for solving multi-objective assembly line balancing problems. *International Journal of Production Research*, **34 (9)**, 2533–2552.

Manavizadeh, N., Rabbani, M., Moshtaghi, D., and Jolai, F., 2012. Mixed-model assembly line balancing in the make-to-order and stochastic environment using multi-objective evolutionary algorithms. *Expert Systems with Applications*, **39 (15)**, 12026–12031.

Manavizadeh, N., Rabbani, M., and Radmehr, F., 2015. A new multi-objective approach in order to balancing and sequencing u-shaped mixed model assembly

line problem: a proposed heuristic algorithm. *The International Journal of Advanced Manufacturing Technology*, **79 (1-4)**, 415–425.

Mansouri, S. A., 2005. A multi-objective genetic algorithm for mixed-model sequencing on JIT assembly lines. *European Journal of Operational Research*, **167 (3)**, 696–716.

McMullen, P. R., 1998. JIT sequencing for mixed-model assembly lines with setups using tabu search. *Production Planning and Control*, **9 (5)**, 504–510.

McMullen, P. R. and Frazier, G. V., 2000. A simulated annealing approach to mixed-model sequencing with multiple objectives on a just-in-time line. *Iie Transactions*, **32 (8)**, 679–686.

McMullen, P. R., Tarasewich, P., and Frazier, G. V., 2000. Using genetic algorithms to solve the multi-product JIT sequencing problem with setups. *International Journal of Production Research*, **38 (12)**, 2653–2670.

Miltenburg, G. and Wijngaard, J., 1994. The u-line line balancing problem. *Management Science*, **40 (10)**, 1378–1388.

Minzu, V. and Henrioud, J. M., 1998. Stochastic algorithm for tasks assignment in single or mixed-model assembly lines. *Journal européen des systèmes automatisés*, **32 (7-8)**, 831–851.

Miyakawa, S. and Ohashi, T., 1986. The hitachi assemblability evaluation method (AEM). In: Proceedings of the international conference on product design for assembly. pp. 15–17.

Mosadegh, H., Ghomi, S. F., and Zandieh, M., 2012a. Simultaneous solving of balancing and sequencing problems in mixed-model assembly line systems. *International Journal of Production Research*, **50 (18)**, 4994–5016.

Mosadegh, H., Zandieh, M., and Ghomi, S. F., 2012b. Simultaneous solving of balancing and sequencing problems with station-dependent assembly times for mixed-model assembly lines. *Applied Soft Computing*, **12 (4)**, 1359–1370.

Mozdgir, A., Mahdavi, I., Badeleh, I. S., and Solimanpur, M., 2013. Using the taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing. *Mathematical and Computer Modelling*, **57 (1)**, 137–151.

Nearchou, A. C., 2011. Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. *International Journal of Production Economics*, **129 (2)**, 242–250.

Nourmohammadi, A. and Zandieh, M., 2011. Assembly line balancing by a new multi-objective differential evolution algorithm based on TOPSIS. *International Journal of Production Research*, **49 (10)**, 2833–2855.

Ozcan, U., 2010. Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm. *European Journal of Operational Research*, **205 (1)**, 81–97.

Ozcan, U., Cercioglu, H., Gokcen, H., and Toklu, B., 2009. A tabu search algorithm for the parallel assembly line balancing problem. *Gazi University Journal of Science*, **22 (4)**, 313–323.

Poli, C. and Fenoglio, F., 1987. Designing parts for automatic assembly. *Machin Design*, **59(29)**, 140–145.

Raidl, G. R., 2006. A unified view on hybrid metaheuristics. In: Hybrid Metaheuristics. Springer, pp. 1–12.

Ramezanian, R. and Ezzatpanah, A., 2015. Modeling and solving multi-objective

mixed-model assembly line balancing and worker assignment problem. *Computers and Industrial Engineering*, **87**, 74–80.

Rekiek, B., Delchambre, A., Dolgui, A., and Bratcu, A., 2002a. Assembly line design: a survey. In: IFAC 15th Triennal World Congress.

Rekiek, B., Dolgui, A., Delchambre, A., and Bratcu, A., 2002b. State of art of optimization methods for assembly line design. *Annual Reviews in Control*, **26 (2)**, 163–174.

Salehi, M., Fattahi, P., Roshani, A., and Zahiri, J., 2013. Multi-criteria sequencing problem in mixed-model synchronous assembly lines. *The International Journal of Advanced Manufacturing Technology*, **67 (1-4)**, 983–993.

Scholl, A., 1999. Balancing and sequencing of assembly lines (heidelberg: Physica-verlag). , .

Scholl, A. and Becker, C., 2006. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, **168 (3)**, 666–693.

Scholl, A., Klein, R., and Domschke, W., 1998. Pattern based vocabulary building for effectively sequencing mixed-model assembly lines. *Journal of Heuristics*, **4 (4)**, 359–381.

Simaria, A. S., 2006. Assembly line balacing: new perspectives and procedures. Ph.D. thesis.

Simaria, A. S. and Vilarinho, P. M., 2004. A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II. *Computers and Industrial Engineering*, **47 (4)**, 391–407.

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

Sivanandam, S. and Deepa, S., 2007. Introduction to genetic algorithms. Springer Science and Business Media,

Sivasankaran, P. and Shahabudeen, P., 2014. Literature review of assembly line balancing problems. *The International Journal of Advanced Manufacturing Technology*, **73 (9-12)**, 1665–1694.

Talbi, E. G., 2009. Metaheuristics: from design to implementation. Vol. 74. John Wiley and Sons,

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

Tasan, S. O. and Tunali, S., 2008. A review of the current applications of genetic algorithms in assembly line balancing. *Journal of Intelligent Manufacturing*, **19 (1)**, 49–69.

Tempelmeier, H., 2003. Practical considerations in the optimization of flow production systems. *International Journal of Production Research*, **41 (1)**, 149–170.

Thomopoulos, N. T., 1967. Line balancing-sequencing for mixed-model assembly. *Management Science*, **14 (2)**, B–59.

Tiacci, L., 2015. Simultaneous balancing and buffer allocation decisions for the design of mixed-model assembly lines with parallel workstations and stochastic task times. *International Journal of Production Economics*, **162**, 201–215.

Torenli, A., 2009. Assembly line design and optimization. Ph.D. thesis, Chalmers University of Technology, Sweden.

Van Zante-de Fokkert, J. I. and de Kok, T. G., 1997. The mixed and multi model line balancing problem: a comparison. *European Journal of Operational Research*, **100 (3)**, 399–412.

Wang, B., Rao, Y., Shao, X., and Wang, M., 2008a. Scheduling mixed-model assembly lines with cost objectives by a hybrid algorithm. In: Intelligent Robotics and Applications. Springer, pp. 378–387.

Wang, B., Rao, Y., Shao, X., and Wang, M., 2008b. Sequencing mixed-model assembly lines with limited intermediate buffers by a GA/SA-based algorithm. In: Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques. Springer, pp. 15–22.

Wester, L. and Kilbridge, M., 1964. The assembly line model-mix sequencing problem. In: Proceedings of the third international conference on Operations Research. Dunod, pp. 247–60.

Wilhelm, W. E., 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 (10)**, 12453–12461.

Yaman, R., 2008. An assembly line design and construction for a small manufacturing company. *Assembly Automation*, **28 (2)**, 163–172.

Yano, C. A. and Rachamadugu, R., 1991. Sequencing to minimize work overload in assembly lines with product options. *Management Science*, **37 (5)**, 572–586.