DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

A GENETIC ALGORITHM BASED APPROACH FOR SIMULTANEOUSLY SOLVING U-SHAPE MIXED-MODEL ASSEMBLY LINE BALANCING AND SEQUENCING PROBLEM

by
Alper HAMZADAYI

June, 2010
İZMİR

A GENETIC ALGORITHM BASED APPROACH FOR SIMULTANEOUSLY SOLVING U-SHAPE MIXED-MODEL ASSEMBLY LINE BALANCING AND SEQUENCING PROBLEM

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

by
Alper HAMZADAYI

June, 2010
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"A GENETIC ALGORITHM BASED APPROACH FOR SIMULTANEOUSLY SOLVING U-SHAPE MIXED-MODEL ASSEMBLY LINE BALANCING AND SEQUENCING PROBLEM"** completed by **ALPER HAMZADAYI** under supervision of **ASSIST. PROF. DR. GÖKALP YILDIZ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Gökalp YILDIZ

Supervisor

(Jury Member)	(Jury Member)

Prof.Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratefulness to people who helped me on my research and thesis.

First of all, I would like to point out my gratitude to my advisor Assist. Prof. Dr. Gökalp Yıldız for his great patience, inspiration and support throughout this master thesis. His wisdom, encouragement and guidance always gave me the direction during the research.

I would like to thank Assist. Prof. Dr. Ceyhun Araz, for his comments on my thesis and also to my friends for their support, whenever I need, and listening to my complaints during this period.

Last, but the most, I would like to show my deepest appreciation to my parents, Saffet Hamzadayı and Züleyha Hamzadayı, for their endless love and support in my whole life and also to brothers, Süleyman Hamzadayı and Erdinç Hamzadayı for confidence, encouragement and endless support.

Alper HAMZADAYI

A GENETIC ALGORITHM BASED APPROACH FOR SIMULTANEOUSLY SOLVING U-SHAPE MIXED-MODEL ASSEMBLY LINE BALANCING AND SEQUENCING PROBLEM

ABSTRACT

Two important problems occur routinely on mixed-model production lines, regardless of whether the lines are traditional or U-shaped. The first one is the problem of how to assign tasks to stations on the line and the second one is the problem of selecting the order or sequence in which different models will be produced. Line balancing and model sequencing problems are tightly interrelated with each other for the mixed-model U-shape assembly line (MMUL), because different models require different tasks and the same tasks have different completion times for different models.

In this thesis, a Priority-Based Genetic Algorithm (PGA) based solution approach is proposed in order to overcome implementation difficulties of the mixed-model U-shape assembly line balancing/sequencing problem (MMUL/BS) simultaneously. In proposed algorithm, Simulated Annealing (SA) algorithm based fitness evaluation approach is developed for being able to make fitness function calculations easily and effectively. In proposed approach, new neighborhood generation logic is developed in order to handle line balancing and model sequencing problems simultaneously. The proposed PGA based algorithm is able to address some particular features of the assembly process very common in real mixed-model assembly lines such as use of parallel workstations, zoning constraints. Parallel work stations and zoning constraints have not been used together in MMUL/BS solution so far.

Moreover, new fitness function is developed for the cases where parallel workstations are used and not used. New fitness function minimizes the number of stations as primary objective, and ensures the workload balance within and between workstations at the end of all cycles as secondary objective.

Eventually, in order to identify the efficient control parameters, an experimental design is conducted and these new procedures are illustrated with a numerical example. Performance of the proposed approach is tested through a set of test problems with generated minimum part sets.

Keywords: Mixed-model U-shape balancing/sequencing problem; Genetic algorithm; Simulated annealing algorithm; Fitness evaluation-relaxation; Parallel workstation assignment; Zoning constraints

U-ŞEKİLLİ KARIŞIK MODELLİ MONTAJ HATLARINDA HAT DENGELEME VE MODEL SIRALAMA PROBLEMLERİNİN EŞZAMANLI ÇÖZÜMÜ İÇİN GENETİK ALGORİTMA TABANLI BİR YAKLAŞIM

ÖZ

Karışık modelli montaj hatlarında (KMMH), hattın geleneksel veya U-şeklinde olmasına bakılmayarak, iki önemli problem oluşur. Bu problemlerden ilki işlerin iş istasyonlarına nasıl atanacağı ve ikincisi hatta üretilecek farklı modellerin hangi sırayla üretileceğinin seçilmesidir. Hat dengeleme ve model sıralama problemleri U-şekilli karışık modelli montaj hatlarında (UŞKMMH) birbirlerine sıkıca bağlıdır, çünkü farklı modeller farklı işler gerektirir ve aynı işler farklı modeller için farklı iş zamanlarına sahiptir.

Bu tezde U-şekilli karışık modelli montaj hatlarındaki hat dengeleme/model sıralama (UŞKMMH/HDMS) problemlerinin eşzamanlı uygulanmasındaki zorluklarının üstesinden gelebilmek için, Öncelik Tabanlı Genetik Algoritma (ÖTGA) tabanlı bir yaklaşım önerildi. Önerilen algoritmada, çözüm değerlendirmelerinin kolay ve etkili bir biçimde yapılabilmesi için Tavlama Benzetimi (TB) algoritması tabanlı çözüm değerlendirme yaklaşımı geliştirildi. Önerilen yaklaşımda, hat dengeleme ve model sıralama problemlerinin eşzamanlı ele alınabilmesini sağlamak amacıyla yeni komşuluk üretme mekanizması geliştirildi. Önerilen ÖTGA tabanlı yaklaşım, gerçek hayat montaj hatlarında sıkça rastlanan paralel istasyon ve bölgesel kısıtlar gibi özellikleri ele alabilecek niteliktedir. UŞKMMH/HDMS çözümünde paralel istasyon ve bölgesel kısıtlar daha önce hiçbir çalışmada beraber ele alınmadı.

Ayrıca paralel iş istasyonlarının kullanıldığı ve kullanılmadığı durumlar için ayrı değerlendirme fonksiyonları geliştirildi. Yeni değerlendirme fonksiyonu birincil amaç olarak istasyon sayısını minimize etmekte ve ikincil amaç olaraktan bütün çevirimler sonunda istasyon içi-arası iş yükü dengesini sağlamaktadır.

Son olarak, etkin kontrol parametrelerini saptamak amacıyla deney tasarımı kuruldu ve bu yeni prosedürler sayısal örnekle gösterildi. Önerilen yaklaşımın performansı test problemleri ve üretilen en küçük kısım setleriyle test edildi.

Anahtar Kelimeler: U-şekilli karışık model montaj hattı dengeleme/sıralama problemi; Genetik algoritma; Tavlama benzetimi algoritması; Çözüm değerlendirme-esnetme; Paralel iş istasyonu ataması; Bölgesel kısıtlar

CONTENTS **Page**

M.Sc THESIS EXAMINATION RESULT FORM ii

ACKNOWLEDGMENTS iii

ABSTRACT iv

ÖZ vi

CHAPTER ONE - INTRODUCTION 1

1.1 Relevance of the Problem 1

1.2 Objective of the Thesis 3

1.3 Structure of the Thesis 4

CHAPTER TWO - MAIN CHARACTERISTICS OF ASSEMBLY LINE SYSTEMS AND ASSEMBLY LINE BALANCING 6

2.1 Introduction 6

2.2 Main Characteristics of Assembly Line Systems 6

2.2.1 Basic Concepts of Assembly Lines 7

2.2.2 Additional Characteristics of Assembly Lines 8

2.2.2.1 Number of Products 9

2.2.2.2 Line Control 10

2.2.2.3 Variability of Task Times 11

2.2.2.4 Assignment Constraints 11

2.2.2.5 Line Layout 12

2.2.3 Performance Measures of Assembly Lines 14

2.3 Assembly Line Balancing 15

2.4 Solution Approaches for Assembly Line Balancing Problems 20

2.4.1 Exact Methods 22

2.4.1.1 Branch and Bound 22

2.4.1.2 Dynamic Programming 22

2.4.1.3 Graph Search Technique 23

2.4.2 Approximation Methods 23

2.4.2.1 Simple Heuristics 23

2.4.2.2 Meta-Heuristics 24

2.5 Literature Review 24

CHAPTER THREE - BACKGROUND INFORMATION FOR SOLUTION METHODS: GENETIC ALGORITHM AND SIMULATED ANNEALING ... 37

3.1 Introduction 37

3.2 Genetic Algorithms 37

3.3 Simulated Annealing Algorithms 43

CHAPTER FOUR - PROPOSED GENETIC ALGORITHM BASED APPROACH FOR SIMULTANEOUSLY SOLVING U-SHAPE MIXED-MODEL ASSEMBLY LINE BALANCING AND SEQUENCING PROBLEM 46

4.1 Chapter Introduction 46

4.2 Characteristics of U-shaped Assembly Lines 47

4.3 Problem Statement of the MMUL/BS 49

4.3.1 Model Assumptions 51

4.3.2 Notation and Equations 51

4.3.3 New Objective Function 54

4.3.4 Proposed GA-Based Approach 56

4.3.4.1 Selected Chromosome Representation and Initialization of Population 57

4.3.4.2 Selected Selection Scheme 61

4.3.4.3 Selected Genetic Operators 61

4.3.4.3.1 Crossover Operator. 62

4.3.4.3.2 Mutation Operator 62

4.3.4.4 Selected Survival Scheme 63

4.3.4.5 Selected Termination Criteria 63

4.3.5 The Proposed Simulated Annealing Algorithm Based Fitness Evaluation Approach 63

4.3.5.1 Initial Solution (LB_0 - MS_0) 72
4.3.5.2 Neighboring Solutions (LB_n - MS_n) 73
4.3.5.2.1 Line Balancing (LB_n). 73
4.3.5.2.2 Sequencing (MS_n). 82
4.3.5.3 Checking Feasibility of Workstation Times 82
4.3.6 Identifying Efficient Control Parameters 83
4.3.7 Numerical Illustration 88
4.3.8 Computational Experiments and Analysis 93
4.4 Use of Parallel Workstations and Zoning Constraints 94
4.4.1 Assumptions 96
4.4.2 Notations and Equations 97
4.4.3 New Objective Function 98
4.4.4 Numerical illustration 100
4.4.5 Computational Experiments and Analysis 106

CHAPTER FIVE - CONCLUSION 107

REFERENCES 109

CHAPTER ONE
INTRODUCTION

1.1 Relevance of the Problem

Assembly work has a long history, and ancient people know how to create useful objects composed of multiple parts. The evolution of manufacturing has passed through distinct stages at various periods of history (Rekiek and Delchambre, 2006). The most important milestone in assembly is the invention of assembly lines (ALs). In 1913, Henry Ford invented the ALs in automobile manufacturing for the first time, which revolutionized the concept of assembly. He also was the first to introduce a moving belt in the factory. Employees were able to build cars one piece at a time instead of one car at a time. This concept changed the type of manufacturing system and reduced the cost of production.

ALs are production systems which consist of succeeding stations, connected by a material handling system, usually a conveyor belt, performing a set of tasks on the product passing through them.

Over the years, the problem of designing efficient assembly lines received considerable attention of both companies and academicians. A well-known assembly design problem is the assembly line balancing problem (ALBP). ALBP deals with the allocation of tasks among workstations for minimizing/maximizing a given objective function.

Until now, the role of assembly lines has been changed. Assembly lines were firstly used to produce a low variety of products in high volumes. However, customers were introduced to the new marketing strategies by TV, radio etc. at their houses. So, the emergence of new advertising channels increased the customer requirements for goods. People wanted to choose different models with a variety of features in different sizes and colors. For example, in the automobile industry, each model has some options and customers can choose any model based on their requirements and their purchasing power: options of engine power, kinds of fuel, and so on. Manufacturers were confronted with the need for offering a variety of features and finding a way to react quickly to market trends or lose market shares. Therefore, the life cycles of products become shorter. The rapid qualitative and quantitative changes in market demands caused manufacturers to seek the most efficient methods for managing their assembly lines so as to produce more sophisticated and more competitive products. Approaches like flexible manufacturing, just-in-time, and group technology arose at that moment. In such environments, mixed-model assembly lines (MMAL) appear to be the most appropriate ones. In MMAL, a set of similar models of a product, which may differ from one model to another with respect to size, color etc., can be assembled simultaneously in the same line, in order to avoid unnecessary inventories and increase manufacturing flexibility for responding to the changing demands of the customers.

MMAL is a production line on which a variety of product models having similar characteristics are assembled. The produced products in MMALs usually have differences in the amount of production, work contents, and assembly time depending on the models. In such environments, an important decision problem, i.e., mixed-model assembly line balancing problem (MMAL/BP) arises. This problem deals with the allocation of the assembly tasks equally among workstations so that the given objective function is minimized/maximized and the precedence relations are satisfied. MMAL/BP is NP-hard and multi-objective in nature.

Recently, U-type layouts have been utilized in many production lines in place of the traditional straight-line configuration due to the use of just-in-time production principles. This helps manufacturers to provide their customers with a variety of timely and cost effective products, also reduces the efforts for adjusting production facilities to demand changes, and increases labor productivity. U-type layouts, on which mixed-model production is performed, are called as the Mixed-Model U-Lines (MMUL).

1.2 Objective of the Thesis

In recent years, in order to provide alternative methods to traditional optimization techniques, most of the researches have directed their works towards the development of heuristics and meta-heuristics, such as simulated annealing, tabu search (TS) and genetic algorithms (GAs). Among these meta-heuristics, the applications of GAs received a considerable attention from the researchers since it provides an alternative to traditional optimization techniques by using directed random search to locate optimum solutions in complex landscapes; and it is also proven to be effective in various combinatorial optimization problems.

Workloads of workstations in MMUL depend on more factors than the other type of line balancing problems and development of solution procedures for MMUL balancing is more complex than that of other types of line balancing (Kara and Tekin, 2009).

The objective of this thesis is to present a solution method based on Priority-Based Genetic Algorithm (PGA) for being able to solve line balancing and model sequencing problems effectively in a simultaneous manner in MMULs. To efficiently implement proposed algorithm, Simulated Annealing Algorithm (SA) based fitness evaluation approach is developed. The proposed PGA is able to address some particular features of the assembly process very common in real mixed-model assembly lines such as the use of parallel workstations, zoning constraints, U-shaped layouts. Considering these features simultaneously in a single method is a major contribution of this thesis. A new fitness function is also developed in order to encompass these features. New fitness function aims at minimizing the number of workstations (Type 1) as primary goal and smoothing the workload between and within at the end of all cycles as secondary goal.

1.3 Structure of the Thesis

This thesis is divided into five chapters. The first chapter briefly introduces the theme of the study, points out the relevance of the problem and presents the main objectives of the work.

In Chapter two, an overview of the assembly line balancing problem is given. It presents the main characteristics of assembly lines and defines the assembly line balancing problem. Different types of assembly line configurations and particular features of the assembly process that may restrict the configuration of the lines are presented. Solution approaches for assembly line balancing problems are given. And then, to identify the current research issues, a literature review is presented for tackling the assembly line balancing problems.

In Chapter three, the main characteristics of the selected meta-heuristics (genetic algorithms and simulated annealing) are introduced.

In Chapter four, the simultaneous solution of balancing/sequencing (MMUL/BS) problems in U-shaped assembly line is addressed. Firstly, general characteristics of U-shape assembly lines are introduced and differences of Mixed-model U-shape assembly lines from other lines are explained in detail. The problem is presented with notations and equations so that the general characteristics of the addressed problem can be understood better. And then, our proposed fitness function minimizing the number of stations and ensuring workload balancing between-within workstations at the end of all cycles is mathematically presented, and our proposed solution method based on priority-based genetic algorithm is introduced. In our proposed genetic algorithm based solution method, simulated annealing based fitness evaluation approach is developed in order to perform fitness assessments. Experimental design is conducted in order to ensure the execution of our proposed algorithms with more efficient parameters. These new procedures are illustrated with a numerical example and its performance is tested through a set of test problems with the generated minimum part sets (MPS). Finally, the problem is expanded in a manner comprising parallel workstations and zoning constraints. These new features are given with notations and equations. Our proposed fitness function is expanded in a manner comprising the characteristics of the parallel stations. Also, these procedures are illustrated with a numerical example and its performance is tested through a set of test problems with the generated MPS.

Finally, In Chapter five, conclusions and the possible future research directions about the problem are pointed out.

CHAPTER TWO
MAIN CHARACTERISTICS OF ASSEMBLY LINE SYSTEMS AND ASSEMBLY LINE BALANCING

2.1 Introduction

In this chapter, an overview of the assembly line balancing problem is given. It presents the main characteristics of the assembly line systems and defines the assembly line balancing problem. Different types of assembly line configurations and particular features of the assembly process that may restrict the configuration of the lines are presented. Solution approaches for assembly line balancing problems are given. And then, a literature review to tackle the assembly line balancing problems is presented for identifying current research issues.

2.2 Main Characteristics of Assembly Line Systems

The concept of assembly line is quite simple; a number of stations are connected through a material handling system, usually a conveyor belt, and each station performs one or more tasks (addition of components, inspection, etc.) on partially finished product in front of it (see Figure 2.1). For a comprehensive review on assembly lines, see Boysen et al., 2008.

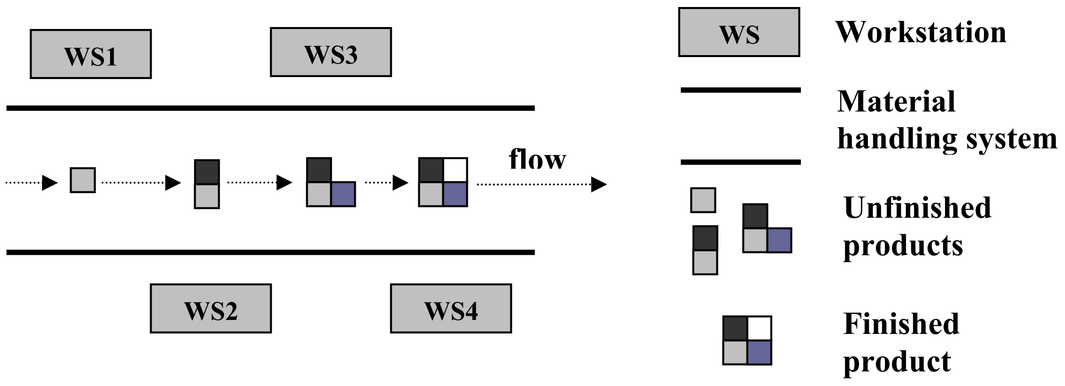


Figure 2.1 Concept of AL

2.2.1 Basic Concepts of Assembly Lines

Assembly is the process of collecting various parts together in order to create a finished product.

Assembly line is a flow-line production system composed of a sequence of workstations that are arranged along a material handling system. Unfinished and partially finished parts are consecutively launched down the system to create finished products, and are moved from one station to another.

Task is a small portion of the total work needed to be accomplished to assemble the product.

Task processing time (task time) is the time necessary for performing an operation (task).

Workstation (station) is a segment of assembly line in which one or more tasks are performed along the work flow by one or more workers.

Precedence relations are the task sequence in which order tasks must be performed.

Precedence diagram is a graphical representation of the sequence of tasks as defined by the precedence relations. Figure 2.2 shows an example of a precedence diagram, in which the nodes represent tasks and the arcs express the precedence relationships between the tasks. For example, task 4 can only be performed after the completion of tasks 1 and 2 (tasks 1 and 2 are direct predecessors of task 4), and its processing time is 3.

Cycle time is the time between the departures of two consecutive products from the line. In other words, it represents maximum amount of the work processed by each station. Cycle time can not be smaller than the largest processing time, and cycle time must not exceed the station time on the assembly line.

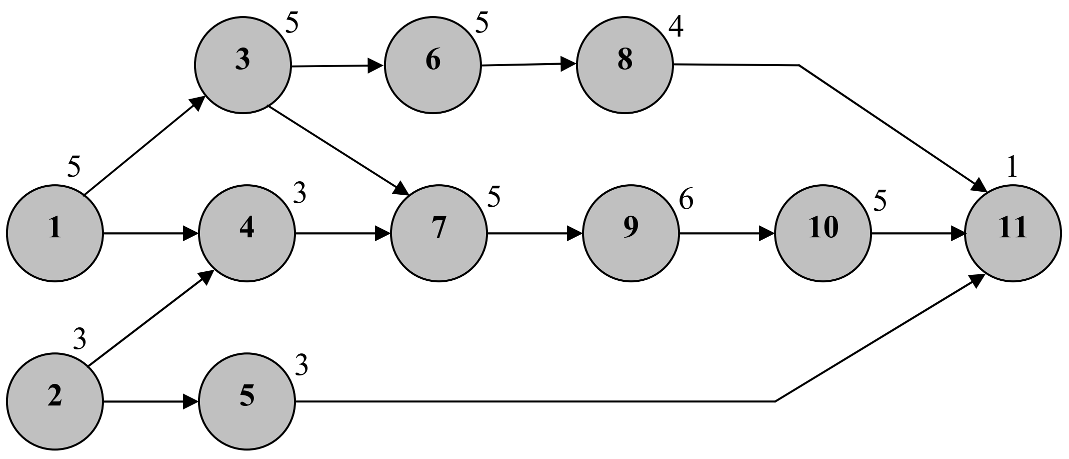


Figure 2.2 Precedence diagram

Workstation time (station time) is the total work content of a station, and it is also referred as station workload. In other words, it represents the sum of the times of assigned tasks in a particular workstation.

Workstation idle time is the positive difference between the cycle time and the workstation time.

2.2.2 Additional Characteristics of Assembly Lines

Assembly line systems show a great diversity due to very different conditions in industrial manufacturing. Assembly lines can be classified in a variety of additional technical or organizational aspects such as the number of products, line control, variability of task processing times, line layout, assignment restrictions, level of automation, type of stations, and etc. (Scholl, 1999; Baudin, 2002; Becker and Scholl, 2006; Rekiek and Delchambre, 2006; Boysen et al., 2008). Figure 2.3 illustrates main characteristics of assembly line balancing problems (Scholl, 1999). While continuous lines indicate that a particular combination of characteristics is typical, broken lines signify that it is unusual.

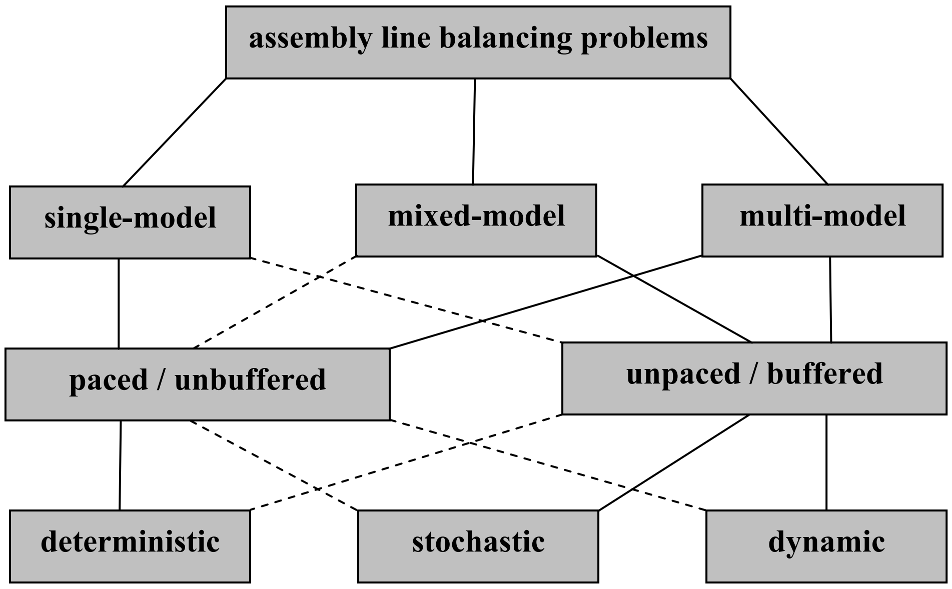


Figure 2.3 Classification of assembly line balancing problems

Some of most important properties of ALs can be explained as follows:

2.2.2.1 Number of Products

The number and variety of products assembled in the line can be categorized as single-model lines, mixed-model lines and multi-model lines (see Figure 2.4).

Single-model assembly lines; assembly lines are used to produce high-volume production of only one product.

Mixed-model assembly lines; assembly lines are used to produce simultaneously a set of different models of the same base product in an arbitrarily intermixed sequence (not in batches).

Multi-model assembly lines; assembly lines are used to produce batches of similar models with intermediate setup operations.

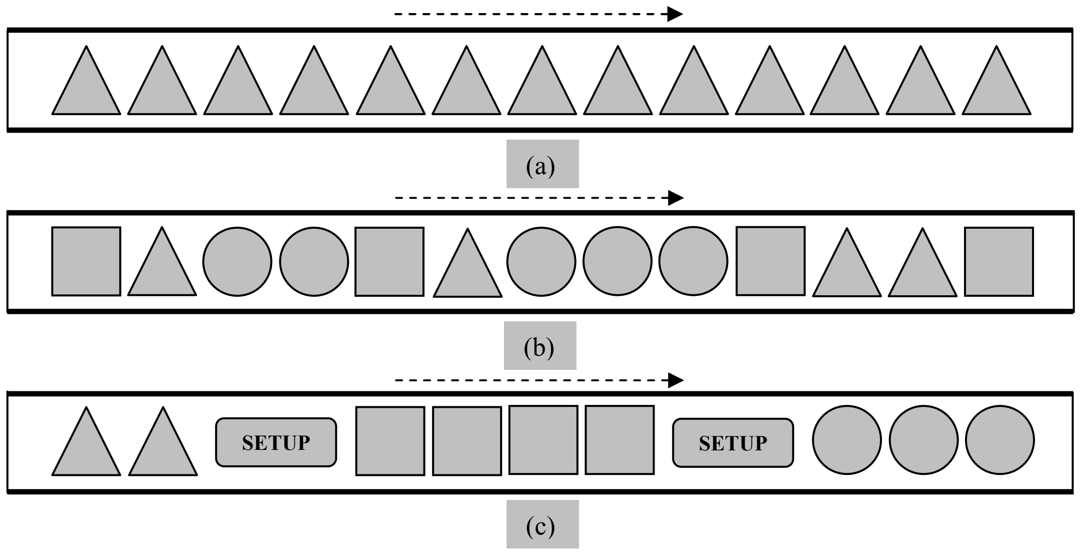


Figure 2.4 Assembly lines for (a) single-model, (b) mixed-model products, and (c) multi-model

2.2.2.2 Line Control

Line control can be categorized as paced assembly lines and unpaced assembly lines. In a paced assembly line, each workstation has a fixed amount of time to complete all the tasks which are assigned to it: the cycle time. When this time is elapsed the sub-assembly must be transferred to the next workstation, and the workstation receives a new sub-assembly from the previous workstation. Hence, these assembly lines have a fixed production rate equal to the reciprocal of the cycle time. Because tasks are indivisible work elements, cycle time can not be smaller than the largest task time. The absence of this fixed time can be referred as unpaced assembly lines. All workstations operate at an individual speed so that work pieces may have to wait before entering the next workstation and workstations may be idle when they have to wait for the next work piece. Allowing buffers between the workstations partially overcome the above mentioned difficulties. So, the ALBP is accompanied by the additional decision problem of positioning and dimensioning of buffers.

2.2.2.3 Variability of Task Times

A further important characteristic defining different versions of ALs is the variability of task times. The variability of task processing times depends on the nature of the tasks and operators.

Deterministic task time; in assembly lines, expected variance of the task times may remain sufficiently small, due to simple tasks or highly reliable equipment. Modern machines and robots are able to work permanently at a constant speed. In this case, task processing times are assumed to be deterministic.

Stochastic task time; in automated flow line-production systems, various production rates may be caused by machine breakdowns, the instability of worker's pace skill and motivation. To incorporate the processing time variability, operation times may be modified by adding the stochastic component.

Dynamic task time; in case of human workers, systematic reductions or successive improvements are possible due to learning effects of the production process. In this case, the task processing times are assumed to be dynamic.

2.2.2.4 Assignment Constraints

Several types of assignment constraints may restrict the possible assignments of tasks into workstations.

Task related constraints; in some situations, pairs of tasks must be assigned to same workstation or not, which are called positive or negative zoning constraints, respectively. Positive zoning constraints are related to the use of common equipment, tool or common processing conditions such as temperature, moisture, operator qualification level etc., so it is desirable that they must be assigned to the same workstation. In some cases, tasks are incompatible and must not be performed at the same workstation, which are called negative zoning constraints (e.g. milling and measuring operations, painting and drilling operations must not be performed at the same workstations).

Workstation related constraints; in some situations, special machines or tools requiring the execution of certain tasks are only available in one or a few workstations, and can not be moved another location.

Position related constraints; in some situations, tasks may need a certain position of the work pieces so that it may be neither possible nor economical to turn the work pieces too often (e.g., heavy items such as car, washing machines, etc.).

Operator related constraints; in some situations, tasks require different levels of skills, depending on their complexity. So, some operators must be assigned to the certain tasks.

2.2.2.5 Line Layout

Assembly lines can also be distinguished with regard to layout of the assembly line. Most important assembly lines encountered in industrial facilities may be explained as follows;

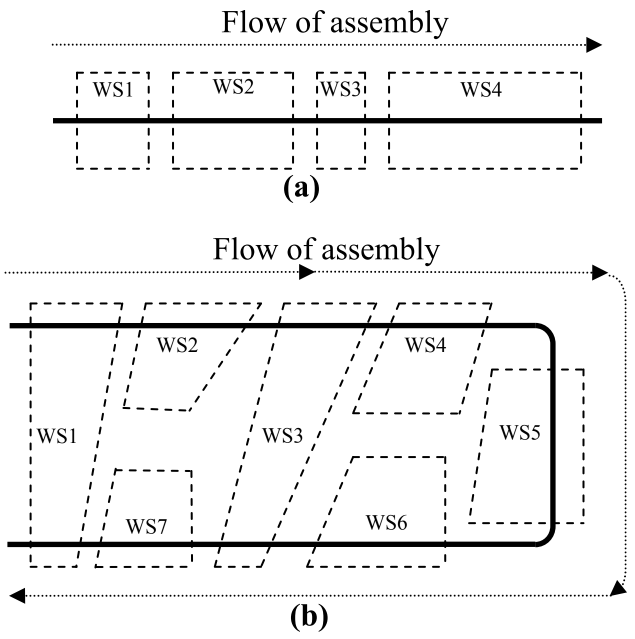
Traditional or Straight (serial) assembly lines: In traditional assembly lines, workstations are physically arranged along a conveyor belt serially, and operators perform tasks on a continuous portion of the line.

U-shaped lines: In U-shape assembly lines, the workstations are arranged along a rather narrow "U" so that during the same cycle two work pieces at different positions on the line can be handled simultaneously. This can result in better balance of workstation loads due to larger number of task-workstation combinations. The U-line assembly line balancing problem is introduced and modeled first by Miltenburg and Wijngaard (1994). Traditional lines may have several disadvantages. So, the companies have switched their lines from straight to U-shaped assembly lines since the just-in-time principles were introduced. A more detailed description of U-shaped assembly lines is given in Chapter four.

Parallel lines: The implementation of these lines allows increase in flexibility and decrease in failure sensitivity of the production system. Furthermore, the use of parallel lines allows the enlargement of cycle time which has several advantages such as the risk of production stoppage due to significant reduce in machine breakdowns; better line balances can often be obtained, because more combinations of tasks exist.

Two-side lines: It may be necessary to operate a two-sided line which consists of two connected serial lines in parallel for assembly heavy work pieces. Instead of single workstation, pairs of opposite workstations on either side of the line (left-hand side and right-hand side workstations) work in parallel, i.e., they work simultaneously at opposite sides of the same work pieces.

Feeder lines: In these lines, the main line fed by other lines where subassemblies are produced. Figure 2.5 illustrates some of the line layouts.



Flow of assembly

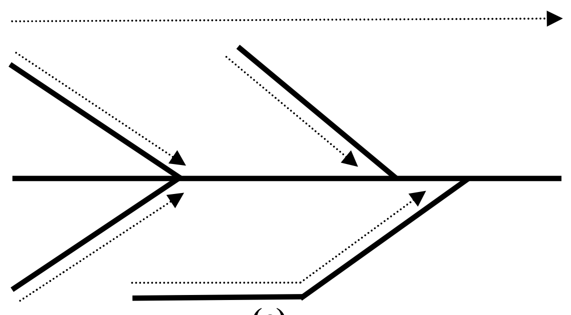


Figure 2.5 Line Layouts: (a) serial, (b) U-shaped, and (c) feeder lines

2.2.3 Performance Measures of Assembly Lines

The installation of an assembly line is a medium or long-term decision and usually requires large capital investments, hence designing and balancing the line is the most important issue in order to produce as efficiently as possible. Besides balancing a new system, a running one has to be re-balanced periodically or after changes in the production process or in the production program have taken place. Because of the long-term effect of balancing decisions, the objectives which are used have to be carefully chosen by considering the strategic goals of the enterprise (Becker and Scholl, 2006).

The most widely used criterions are related with the maximization of the capacity utilization which is measured by the line efficiency (the percentage of productive time in the line) (Ghosh and Gagnon, 1989). Among them are (i) the minimization of the number of workstations for a given cycle time, (ii) the minimization of cycle time for a given number of workstations and (iii) the minimization of the idle time of the line. Other capacity related criterions are as follows (Scholl, 1999): minimizing the flow time (throughput time, i.e. the time interval between launching a work piece down the line and removing the finished product from the line), equalizing the utilization levels of the stations, minimizing the balance delay time (i.e. sum of the idle times) and the balance delay (ratio) (percentage of idle times) over all stations, and minimizing the waiting times of work pieces.

The economical nature criteria deal with minimizing the total cost of the line, including long-term investment cost and short-term operating cost. Both investment and operating costs depend mainly on the cycle time and the number of workstations. The most important cost categories are as shown below (Scholl, 1999).

- machinery and tool costs,
- labor costs,
- materials costs,
- idle time costs,
- penalty costs for not satisfying the demand,
- incompletion costs,
- setup costs
- inventory costs

Besides capacity and cost related objectives, social goals such as job enrichment and job enlargement etc. may be important for assigning less monotonous tasks to an operator and for increasing the number of tasks performed by an operator.

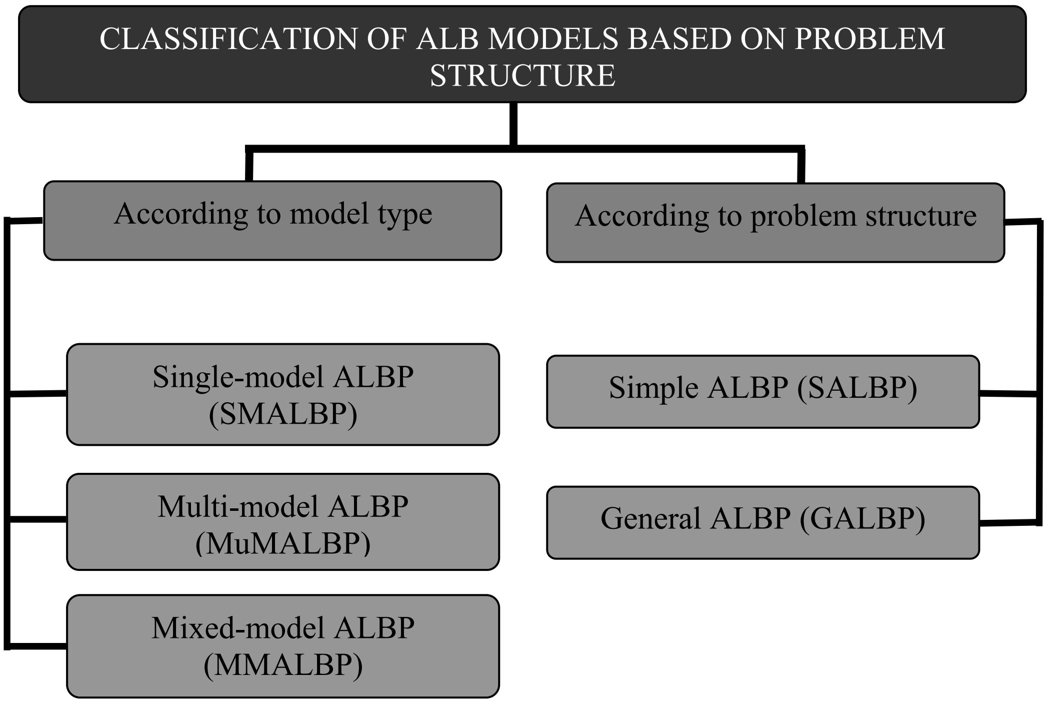
2.3 Assembly Line Balancing

Assembly line balancing is the problem of partitioning of tasks to workstations in such a way that some performance measures are maximized/minimized subject to precedence relationship among tasks (Erel and Sarin, 1998; Becker and Scholl, 2006).

The simple assembly line balancing problem (SALBP) was first mathematically formulated by Salveson (1955) and, since then, a massive body of academic literature has covered the balancing of assembly lines.

The basic problem described so far is called simple assembly line balancing problem (SALBP) in the literature (Baybars, 1986) and, since then, a few versions have been defined by varying problem structure and objective function.

Based on the model structure, ALBP can be classified into two groups as seen in Figure 2.6. This classification compiles the classification schemes of Baybars (1986), Scholl (1999) and Becker and Scholl (2006). The first group includes single-model assembly line balancing problem (SMALBP), multi-model assembly line balancing problem (MuMALBP), and mixed-model assembly line balancing problem (MMALBP); the second group includes simple assembly line balancing problem (SALBP) and general assembly line balancing problem (GALBP). The GALBP model includes all of the models that are not SALBP, such as balancing of mixed-model, parallel, u-shaped and two sided lines with stochastic processing times; thereby more realistic ALBP models can be formulated by GALBP (Gen et al., 2008).


CLASSIFICATION OF ALB MODELS BASED ON PROBLEM STRUCTURE
According to model type
According to problem structure
Single-model ALBP (SMALBP)
Multi-model ALBP (MuMALBP)
Mixed-model ALBP (MMALBP)
Simple ALBP (SALBP)
General ALBP (GALBP)
Figure 2.6 Classification of assembly line balancing models

SALBP has the following main characteristics (Scholl, 1999);

- Mass-production of one homogeneous product
- Given production process
- Paced line with fixed cycle time
- Deterministic and integral operation times
- No assignment restrictions besides the precedence constraints
- Serial layout, one-sided stations
- All stations are equally equipped with respect to machines and workers
- Fixed rate launching, launch interval equals to cycle time

According to objective function, well-known SMALBP versions are as follows (Baybars, 1986; Scholl, 1999);

SALBP-1 (Type-1) consists of assigning tasks to stations so that the number of stations is minimized for a given cycle time.

SALBP-2 (Type-2) aims at maximizing the production rate, or equivalently, minimizing the sum of the idle times for a given number of stations.

SALBP-F is a feasibility problem in which the feasible line balance whether exists or not for a given combination of number of stations and cycle time.

SALBP-E is the most general problem version maximizing the line efficiency thereby simultaneously minimizing cycle time and number of stations considering their interrelationship.

Partly, MMALBP relies on same basic assumptions of SALBP, such as, deterministic processing times, no assignment restrictions, serial line layout, fixed rate launching, etc.

Additional characteristics of MMALBP are as follows (Scholl, 1999):

- The assembly of each model requires performing a set of tasks which are connected by precedence relations (i.e., precedence graph for each model).
- A subset of tasks is common to all models; the precedence graphs of all models can be combined into a non-cyclical joint precedence graph.
- Tasks, which are common to several models, are performed by the same station but they may have different processing times (i.e., zero processing times indicate that the task is not required for the model).
- The total time available for the production is fixed and known (given by the number of shifts and the shift durations).
- The demands for all models (expected model mix) during the planning period are fixed and known.

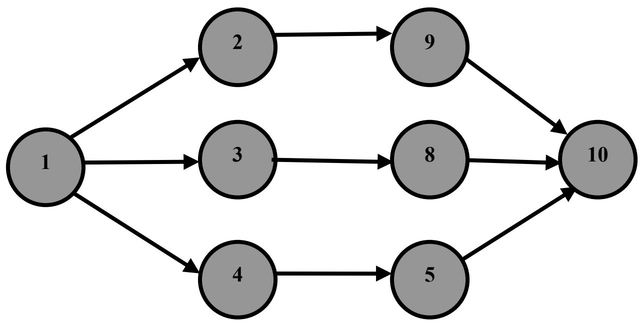
In Figure 2.7, precedence and joint precedence diagrams of two models can be seen.

According to the objective function, the MMALBP can be classified into four different types, (Scholl, 1999):

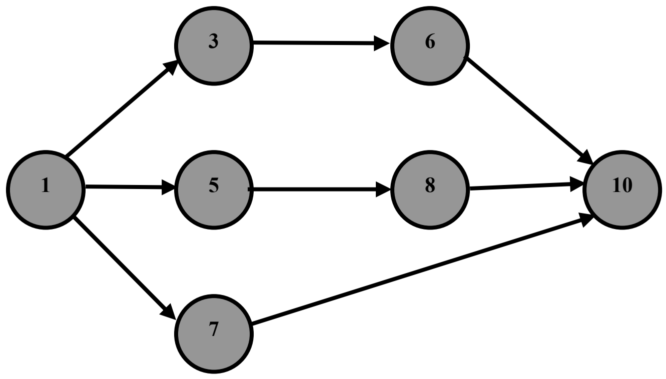
MMALBP-1 (Type-1): Minimizes the number of workstations, for a given cycle time.

MMALBP-2 (Type-2): Minimizes the cycle time, for a given number of workstations.

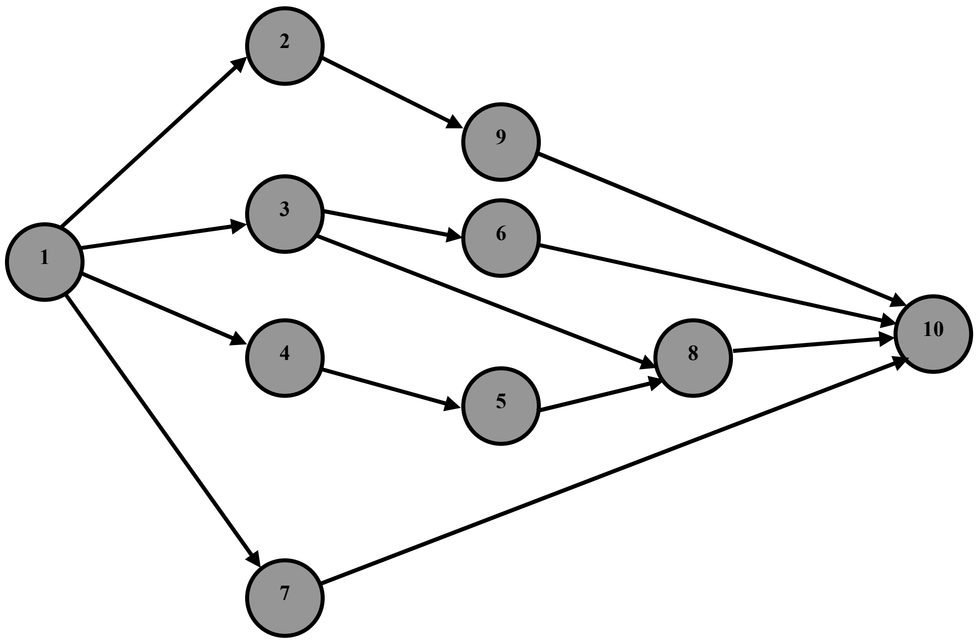
MMALBP-E: According to SALBP-E, the cycle time as well as the number of stations may vary in certain ranges. The objective is to maximize the line efficiency or, equivalently, to minimize the cycle time and the number of stations by considering their interrelationship.



(a)



(b)



(c)

Figure 2.7 Precedence diagrams of (a) model 1, (b) model 2 and (c) combined

MMALBP-F: By analogy with SALBP-F, it is a feasibility problem which is to establish whether a feasible line balance exists for a given combination of number of stations and cycle time.

Additionally, we can define three problem versions of U-line assembly line balancing problem (UALBP) regarding to SALBP version (Scholl, 1999):

UALBP-1 (Type-1): Given the cycle time, minimize the number of stations.

UALBP-2 (Type-2): Given the number of stations, minimize the cycle time.

UALBP-E: Maximize the line efficiency for cycle time and the number of stations which are variable.

In this thesis, the U-shaped mixed-model assembly line balancing Type-1 problem involving the minimization of the number of workstations for a given cycle time is studied.

2.4 Solution Approaches for Assembly Line Balancing Problems

The assembly line balancing problem was firstly formulated by Salveson (1955) and, since then, numerous procedures have been developed for solving the problem. ALBP falls into the NP hard class of combinatorial optimization problems (Karp, 1972). Therefore, the complex mathematical nature of the problem makes it difficult to solve (Erel and Gokcen, 1999). Classification of solution approaches for ALBP (Rekiek and Delchambre, 2006) is given in Figure 2.8.

For a comprehensive literature reviews on both exact and approximation methods for the different types of assembly line balancing problems, the readers can refer to Ghosh and Gagnon (1989) that presents a comprehensive review and analysis of the different methods for design, balancing and scheduling of assembly systems; Erel and Sarin (1998) that present a comprehensive review of the procedures for single-model and multi model assembly lines and by Becker and Scholl (2006) that present a survey on problems and methods for GALBP with features such as cost/profit oriented objectives, equipment selection/process alternatives, parallel workstations/tasks, U-shaped line layout, assignment restrictions, stochastic task processing times and mixed model assembly lines; Scholl and Becker (2006) present a review and analysis of exact and heuristic solution procedures for SALBP and lines; Rekiek and Delchambre (2006) focus on solutions methods for solving SALBP; and Batini et al. (2007) give a classification of the published papers between

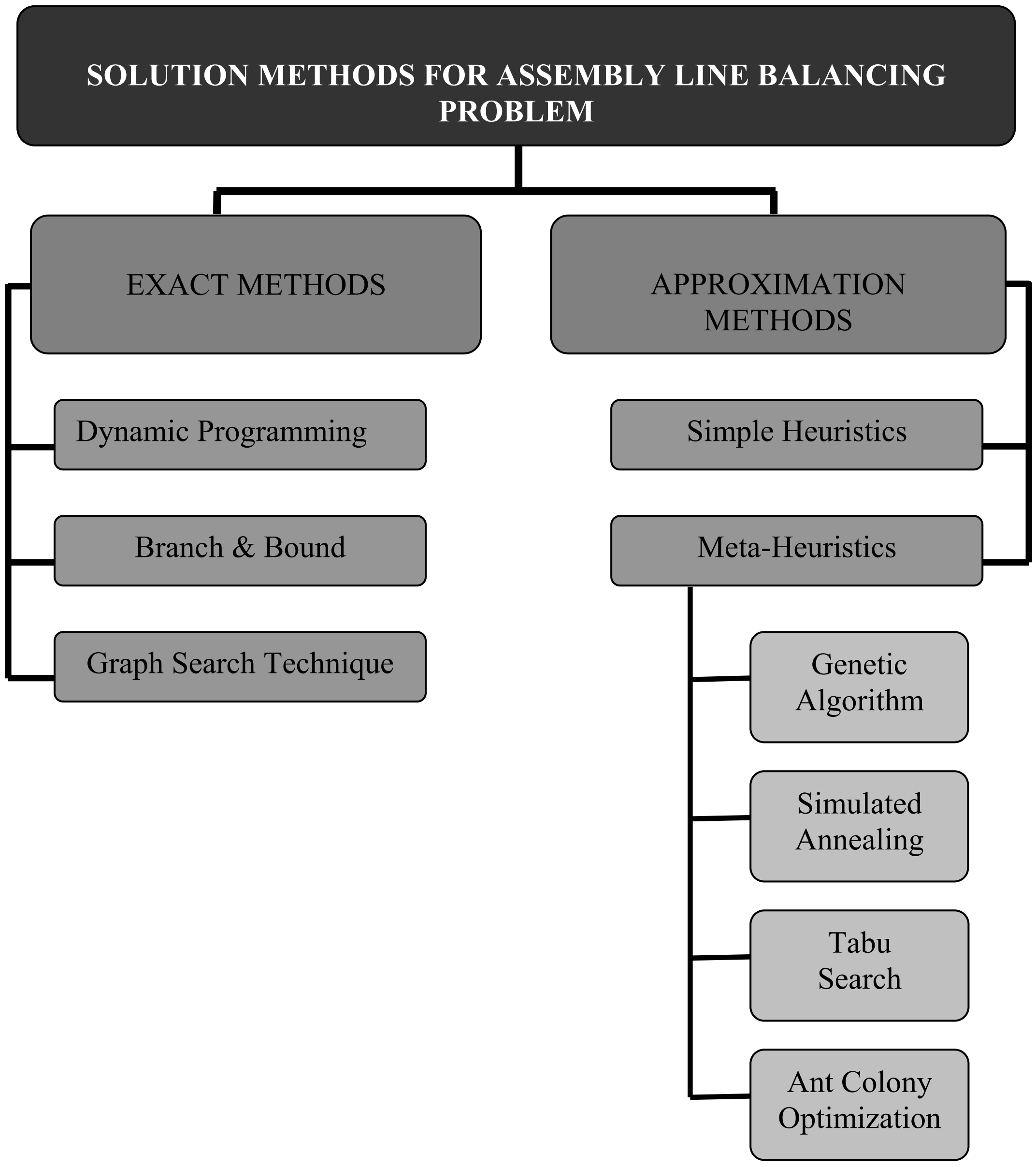


Figure 2.8 Classification of solution approaches for ALBP

the years 1989 and 2005 in relation to the adopted balancing method and the reference layout configuration taken into consideration.

2.4.1 Exact Methods

The optimum seeking methods, i.e., dynamic programming and branch & bound methods have been proposed to solve ALBP. Lower bounds are obtained by solving problems which are derived from the considered problem by omitting or relaxing constraints (Scholl, 1999).

2.4.1.1 Branch and Bound

The branch and bound method is a well-known general solution concept in combinatorial optimization. Branch and bound algorithms consist of two main components *branching (enumeration)* and *bounding*. During the branching process, the initial problem divided into sub-problems. By continuously developing such sub-problems, a multi-level enumeration tree (with sub-problems as nodes) is constructed. Generally, bounding is applied for reducing the size of enumeration trees. This is achieved by computing lower bounds on the number of stations, at least necessary for a feasible solution, in each node. Lower bounds are obtained by solving relaxations which are derived from the problem considered by omitting or relaxing constraints.

2.4.1.2 Dynamic Programming

Like branch and bound, dynamic programming is a general approach for many types of problems including most combinatorial optimization problems. A given problem is divided into sub-problems which are sequentially solved until the initial problem is finally solved.

2.4.1.3 Graph Search Technique

Johnson (1988) proposed a depth-first-search method called fast algorithm for balancing line effectively (FABLE). Sub-problems are constructed by adding an assignable task to the currently considered station k (starting with station 1). If no such task exists, the current station load is maximal and the consecutive stations $k+1$ are opened. In each of the n iterations ($i=1,..n$), one non-marked task with the largest process time (which has no predecessor or only marked predecessors) gets the number i and is marked. Whenever a station is opened, the task with the smallest number among the assignable tasks is added. Any further tasks in the station must have a larger number than the task assigned in the ancestor node. Then, the current branch is traced back by removing tasks assignments until an alternative branch can be followed.

2.4.2 Approximation Methods

Numerous research efforts have been directed for optimum seeking methods in order to obtain an optimal solution. However, none of these methods has proven to be of practical use for large problems due to their computational inefficiency and vast search space. So, instead of exact procedures that find optimal solutions for simplified problems, heuristic procedures are used to find good solutions for much more complex problems. These approaches can be divided into two categories, simple heuristics and meta-heuristics.

2.4.2.1 Simple Heuristics

None of the methods guarantees an optimal solution, but they are likely to result in good solutions. Among simple heuristic methods, the most notable ones are: Ranked Positional Weight Technique (RPWT) (Helgeson and Birnie, 1961), Kilbridge and Wester’s (1961), and Moodie and Young's (1965) heuristics. RPWT is the first heuristic proposed for solving ALBP.

2.4.2.2 Meta-Heuristics

Meta-heuristics are the natural extension of priority-based heuristics, as they start with an initial solution or population (predefined number of solutions) which are obtained through a heuristic or generated randomly. Meta-heuristics improve this initial solution or population. It has been shown that they provide effective approximate solutions for difficult NP-hard combinatorial optimization problems. In recent years, the usage of meta-heuristics for solving ALBPs became popular among researchers. Genetic Algorithm, Simulated Annealing, Tabu Search and Ant Colony Optimization are well known meta-heuristics for solving ALBPs.

2.5 Literature Review

The mathematical formulation of the ALBP for simple assembly lines was first stated by Salveson (1955) and, since then, extensive research has been done in this area. Comprehensive literature reviews on this subject were provided in Baybars (1986), Ghosh and Gagnon (1989), Erel and Sarin (1998), Scholl (1999). For traditional mixed model straight lines, line balancing was studied by few researchers, such as Thomopoulos (1970), Macaskill (1972), Askin and Zhou (1997), Gokcen and Erel (1997, 1998), McMullen and Frazier (1997, 1998), Erel and Gokcen (1999), Merengo et al. (1999), Kim et al. (2000a), Buckhin et al. (2002), Vilarinho and Simaria (2002, 2006), Simaria and Vilarinho (2004), Choi (2009). Model sequencing in the straight lines has been investigated by a number of researchers including Miltenburg and Sinnamon (1989, 1992, 1995), Miltenburg (1989), Yano and Rachamadugu (1991), Kim et al. (2000a), Duplaga and Bragg (1998), Merengo et al. (1999), McMullen and Frazier (2000), Karabati and Sayin (2003). Model sequencing in just-in-time (JIT) production systems has been addressed by Miltenburg (1989), Monden (1993) and McMullen (1998). Line balancing and model sequencing in the straight lines were solved sequentially by few researcher, such as Thomopolous (1967), Dar-el and Navidi (1981), Bard et al. (1992).

In case of U-line production systems, few researches have been carried out recently. Miltenburg and Wijngaard (1994), the first authors to study this problem, developed a dynamic programming exact procedure and a modified ranked positional weight technique (RPWT) heuristic being able to solve instances with up to 11 tasks. In order to address larger problems, they proposed a set of single-pass heuristic procedures being able to solve instances with up to 111 tasks. They also explained the differences between SALB and SULB.

Miltenburg (1998) developed dynamic programming model for solving U-line balancing problem. In his problem, more than one U-line assembly lines in one production line were considered. He found an optimal solution when individual U-lines did not have more than 22 tasks and did not have wide, sparse precedence graphs.

The problem of balancing a U-shaped mixed-model assembly line (U-MALBP) was first described by Sparling and Miltenburg (1998), and they proposed a four-stage approximate solution algorithm. They used the combined precedence diagram and the weighted average task processing times to create a single-model balancing problem, and by using a branch-and-bound algorithm, an optimal solution for this problem was obtained, called initial balance. Several unbalance measures regarding mixed-model nature of the original problem were defined and computed for the initial balance. Then, a smoothing algorithm was applied in order to reduce the unbalance. The objective of this smoothing algorithm was to minimize the absolute deviation of workloads (ADW) among workstations. This algorithm exchanges tasks between workstations so that the value of the selected unbalance measure decreases. An important aspect of this approach was that the sequence in which the models were launched in the U-shaped line must be known, as it directly influences the values of the unbalance measures. Although their study focuses on the minimization of the number of workstation, their algorithm mostly leads to infeasible solutions to the problem by means of cycle time restriction.

Ajenblit and Wainwright (1998) were pioneers in balancing the U-shaped SMALBP Type-1 using GAs. The authors dealt with two possible variations of this problem, minimizing the total idle time and balancing of workload among workstations, or a combination of both. They developed six different assignment algorithms to interpret a chromosome and assign tasks to workstations. These algorithms based on both dynamic programming and various heuristic algorithms, which were proposed in Miltenburg and Wijngaard's research (1994). In this study, the authors applied the proposed GA to 61 test problems. In comparison to previous researches, the proposed GA gave superior results in 11 cases, the same results in 42 problems, superior in 11 problems and worse in 1 problem.

The first integer programming formulation (IP) formulation of SULB was developed by Urban (1998). This formulation uses the phantom precedence diagram concept. A phantom precedence diagram was appended to the original precedence diagram so that assignments to the workstations could be made forward through the original diagram, backward through the phantom diagram, or simultaneously in both directions. The IP formulation managed to solve optimality problems with up to 45 tasks.

Scholl and Klein (1999) developed a branch-and-bound based heuristic called ULINO (U-Line optimizer), which was adapted from a previous algorithm, called SALOME, they had developed for balancing straight lines. The computational experience involved a large set of problems with up to 297 tasks and proved a good performance of the procedure, especially for the objective of minimizing the number of workstations.

The study of Kim et al. (2000b) was the first dealing simultaneously with the problems of balancing and sequencing mixed-model U-lines, as the line balance and the model sequence both influence the performance measure used by the authors: the absolute deviation of workloads (ADW). Combining these two problems results in a new problem, called mixed-model U-line balancing and sequencing (MMUL/BS). These authors proposed a new approach using an artificial intelligence search technique, called co-evolutionary algorithm, which maintains two sets of populations, one to represent solutions of the line balancing problem and the other to represent solutions of the model sequencing problem. Each individual in a population has a matching pair in the other population, and fitness (based on the absolute deviation of workloads) was computed for the pair of individuals. To generate new individuals, different genetic operators were defined for each of the populations. The proposed co-evolutionary algorithm aims at minimizing the ADW for a given number of workstations, and uses such a concept that the solution obtained from the MMUL/LB problem is input to the MMUL/MS problem. Computational experiments proved a good performance of the procedure when compared with that of the hierarchical approach and of two other co-evolutionary algorithms for the same set of test problems.

Erel et al. (2001) developed a simulated annealing (SA) based approach to solve the problem of assembly line-balancing problem a U-type configuration (SULB). The proposed algorithm employs an intelligent mechanism to search a large solution space. The SA procedure aims at achieving feasibility regarding cycle time constraints. The objective function used for the minimization of the maximum station time, thus eliminating the unfeasibility caused by the workstation exceeding the cycle time. They proposed a different way for building the initial solution. First, each task was assigned to a different workstation and then the number of workstations was reduced by combining two adjacent workstations. When the workload of the combined workstation exceeds cycle time, the initial solution was completed and the subsequent steps of the SA procedure were initialized. The performance of the algorithm was measured by solving a large number of benchmark problems available in the literature. The results of the computational experiments indicated that the proposed SA-based algorithm performs quite effectively. It also gave the optimal solution for most problem instances. Future research directions and a comprehensive bibliography were also provided here.

Miltenburg (2002) developed a genetic algorithm (GA) for solving the MMUL/BS, the balancing and sequencing problem, with fixed number of workstation. The model aims at minimizing the ADW and the deviation of part production quantities in a JIT environment to facilitate “level” production. Desired goal to achieve was the generation of level production schedules for other production facilities operating in JIT environment. It took into account the number of parts, from each of the different production facilities, each model required to be assembled. The proposed GA was found to offer good solutions. Detailed information was given concerning the performance of the proposed GA. Average computation times per instance were found to be 130s when the proposed GA employed two point crossovers, 300s when the proposed GA involved cycle crossover and 300s when the proposed GA included randomly generated solutions.

Aase et al. (2003) proposed a set of branch-and-bound procedures, called U-OPT, with different design elements (branching strategies, fathoming criteria, etc.) to solve the U-ALBP. They showed that design elements should be included in optimization procedures or algorithms, including branch-and-bound procedures, for solving the U-shaped assembly line-balancing problem. New solution procedures were proposed and compared experimentally with several existing procedures using a variety of problem sets from the literature. Significant improvements over the existing methods were reported by the authors when solving problem instances of reasonable application size for U-shaped layouts (problems with up to 50 tasks).

Guerriero and Miltenburg (2003) developed a mathematical model and recursive algorithms to solve the U-ALBP (Type-1) with stochastic task processing times. An equivalent shortest path network was also presented. 558 instances were solved by the first algorithm, and the largest 198 instances were solved again by the second algorithm. Their study suggested that the algorithms were able to solve most instances of practical size, where practical size seemed to be 25 or fewer tasks and precedence order strengths of 0.2 or more. So, Computational experiments showed that the algorithms were able to solve problems of practical size.

Aase et al. (2004) addressed the impact on labor productivity. The purpose of this research was to confirm empirically that U-shaped assembly lines improve labor productivity. Results indicated that labor productivity would improve significantly under certain conditions when switching from a straight-line layout to a U-shaped layout but not in all cases. The research also revealed some limitations of such a layout change when factors such as the number of tasks and cycle times were varied.

Martinez and Duff (2004) addressed the U-shaped SMALBP Type-1. They first solved this problem using 10 heuristic rules adapted from a simple line balancing problem, such as maximum ranked positional weight, maximum total number of follower tasks or precedence tasks, and maximum processing time, and compared these heuristic solutions with the optimal solutions obtained from previous researches. Thereafter, they modified the Ponnambalam et al.'s GA (2000) and inserted the solutions obtained using these heuristic rules to the initial population. They illustrated the proposed GA using the Jackson's problem (1956). The results showed that the addition of a GA can improve the current solution.

Gokcen et al. (2005) presented a shortest route formulation for simple U-type assembly line balancing (SULB) problem and illustrated on a numerical example. This model was based on the shortest route model developed by Gutjahr and Nemhauser (1964) for the traditional single model assembly line balancing problem. They noted that future research directions about the developed model could also be used as a framework to develop effective heuristic procedures for solving a simple U-type line-balancing problem.

Erel et al. (2005) presented a beam search-based method for the stochastic assembly line balancing problem in U-lines. The proposed method was the first heuristic for the stochastic U-type problem with the total expected cost criterion. The proposed method minimizes expected total cost comprised of total labor cost and expected total incompletion cost. The performance of the proposed method was measured on various test problems. The results of the computational experiments indicated that the average performance of the proposed method was better than the best-known heuristic in the literature for the traditional straight-line problem. Future research directions and the related bibliography were also provided in this paper.

A goal programming approach to simultaneously consider several conflicting objectives was presented by Gokcen and Agpak (2006). The model was based on the integer programming formulation developed by Urban (1998) for the ULB problem and the goal model of Deckro and Rangachari (1990) that developed for the traditional single model assembly line balancing (ALB) problem. The proposed model, the first multi-criteria decision making approach to the U-line version, provides increased flexibility to the decision maker since several conflicting goals can be simultaneously considered. No comparison with other algorithms was provided and the computational experience was only dedicated to the study of the multi-criteria version of the problem.

Kim et al. (2006) proposed a new evolutionary approach to deal with both balancing and sequencing problems in mixed-model U-shaped lines with fixed number of workstation. A new genetic approach, called endosymbiotic evolutionary algorithm, was proposed for solving the two problems of line balancing and model sequencing at the same time. The algorithm imitates the natural evolution process of endosymbionts that is an extension of existing cooperative or symbiotic evolutionary algorithm. The distinguishing feature of the proposed algorithm is that it maintains endosymbionts being a combination of an individual and its symbiotic partner. The existence of endosymbionts can accelerate the speed that individuals converge to good solutions. This enhanced capability of exploitation together with the parallel search capability of traditional symbiotic algorithms results in finding better quality solutions than existing hierarchical approaches and symbiotic algorithms. A set of experiments were carried out, and the results were reported.

Urban and Chiang (2006) proposed an optimal piecewise-linear program for the U-line balancing problem with stochastic task times**.** This paper examined the U-line balancing problem with stochastic task times. A chance-constrained, piecewise-linear, integer program was formulated for finding the optimal solution. Various approaches used to identify a tight lower bound were also presented. Computational results showed that the proposed method was able to solve problems of practical size.

Kara et al. (2007a) proposed a simulated annealing algorithm based approach for simultaneously solving the balancing and sequencing problems of mixed-model U-lines. The primary goal of the proposed approach was to minimize the number of workstations required on the line (Type I). To meet this aim, the proposed approach uses such a methodology that enables the minimization of the absolute deviation of workloads among workstations as well. In terms of minimizing the number of workstations required on the mixed-model U-line, as well as minimizing the absolute deviation of workloads among workstations, the proposed approach was the first method in the literature dealing with the balancing and sequencing problems of mixed-model U-lines at the same time. The newly developed neighborhood generation method was inserted into the simulated annealing (SA) algorithm. Problem illustrated on a numerical example.

Agpak and Gokcen (2007) developed four different new models of chance-constrained binary integer programming models for the stochastic traditional and U-type line balancing (ULB) problem. In this study, these models have been solved for several test problems well-known in the literature and the results have been compared with respect to the number of stations.

Toklu and Ozcan (2007) presented a fuzzy goal programming model for the simple U-line balancing (SULB) problem with multiple objectives. The proposed model was the first fuzzy multi-objective decision-making approach to the SULB problem with multiple objectives which aims at simultaneously optimizing several conflicting goals. The proposed model was illustrated using an example. A computational study was conducted by solving a large number of test problems to investigate the relationship between the fuzzy goals and to compare them with the goal programming model proposed by Gokcen and Agpak (2006). The results of the computational experiments indicated that the proposed model was more realistic than existing models for the SULB problem with multiple objectives and also gave increased flexibility for the decision-makers to determine different alternatives.

Baykasoglu and Ozbakir (2007) proposed a new multiple-rule-based genetic algorithm (GA) for balancing U-type assembly lines with stochastic task times. The proposed algorithm integrates the COMSOAL method, task assignment heuristics, and a GA. The performance of the proposed algorithm was compared with the optimal solutions found by Urban and Chiang (2006). The proposed algorithm found optimal solutions for all problems, except one case, within considerably shorter CPU times than the existing results. It was concluded that the proposed GA was able to solve problems of practical size with reasonable CPU times.

Kara et al. (2007b) presented a multi-objective simulated annealing algorithm based approach for balancing and sequencing mixed-model U-lines to minimize simultaneously the absolute deviations of workloads across workstations, part usage rate, and cost of setups. To increase the performance of the proposed algorithm, a newly developed neighborhood generation method was also employed. Solution methodology was illustrated using an example; and a two-stage comprehensive experimental study was conducted to determine the effective values of algorithm parameters and investigate the relationships between performance measures. Results showed that the proposed approach was more realistic than the limited number of existing methodologies. The proposed approach was also extended for considering the stochastic completion times of tasks.

Boysen and Fliedner (2008) proposed a versatile algorithm for assembly line balancing. The proposed algorithm consists of two staged graph-algorithm, which was designed to solve line balancing problems including relevant practice constraints (GALBP), such as parallel work stations and tasks, cost synergies, processing alternatives, zoning restrictions, stochastic processing times or U-shaped assembly lines. Unlike former procedures, the presented approach can be easily modified to incorporate all of the named extensions. It is not only possible to select and solve single classes of constraints, but rather any combination of them with just slight modifications.

Hwang et al. (2008) presented a multi-objective genetic algorithm (moGA) using the priority-based coding method to solve the U-shaped assembly line balancing problem (UALBP). They considered both the traditional straight line system and the U-shaped assembly line system, thus as an unbiased examination of line efficiency. Considered performance criteria are the number of workstations (the line efficiency) and the variation of workload. Several well-known test problems considered by Talbot et al. (1986) were solved by using proposed multi-objective genetic algorithm. The results of experiments showed that the proposed model produced as good or even better line efficiency of workstation integration and improved the variation of workload.

Sabuncuoglu et al. (2009) proposed ant colony algorithms to solve the single-model U-type assembly line balancing problem. The problem considered in this study is a single model, deterministic U-line balancing problem. Their objective was to find a design with the minimum number of stations subject to the cycle time and precedence relations constraints. They conducted an extensive experimental study in which the performance of the proposed algorithm was tested by using the benchmark problems in the literature, and was compared against best known algorithms reported in the literature. They used two data sets: Talbot et al. (1986) with 64 instances of problem sizes ranging from 8 to 111 tasks and Scholl (1993) with 168 instances ranging from 25 to 297 tasks. The results indicated that the proposed algorithms display very competitive performance against them.

Hwang and Katayama (2009) proposed a new evolutionary approach to deal with workload balancing problems in mixed-model U-shaped lines without job sequence so that all models are produced by same quantity. Their paper was an extension of the priority-based genetic algorithm (PGA), and designs an amelioration structure with a genetic algorithm (ASGA) to improve workload balance on MMAL production systems. They considered both the traditional straight line system and the U-shaped assembly line; and the performance criteria considered were the number of workstations (the line efficiency) and the variation of workload, simultaneously.

Computational experiments were performed based on three well-known test problems.

Kara and Tekin (2009) presented a mixed integer programming formulation for optimal balancing of mixed-model U-lines. The proposed approach minimizes the number of workstations required on the line for a given model sequence. They also presented the comsoal algorithm based heuristic method. They solved two methods up to 10-task, 20-task and 30-task problem instances. They reported that most of the 10-task problem instances were solved optimally, the optimality of almost none of 20-task and 30-task problem instances was not guarantied or not found, and in addition, feasible solutions were found for most of 20-task problems but feasible solutions could be obtained for a few of 30-task problems.

The literature review is summarized as shown in Table 2.1. This table contains the published papers, which address the U-line assembly line balancing problem in chronological order.

Our conclusions about this review are listed below:

- 8 out of 28 articles surveyed, studied the mixed-model U-shape line balancing problem. The other 19 articles surveyed, focused on the simple U-shape line balancing problem. Only Aase et al. (2004) addressed the benefits of U-shape production lines on labor productivity.

- Only one article (Kara et al., 2007a) dealt with the balancing and sequencing problem of mixed-model U-lines simultaneously to minimize the number of workstations (Type 1). The other five articles that focused on mixed-model U-shape line balancing problem tried to solve model sequencing and line balancing problem sequentially by considering the fixed number of workstation or the fixed model sequence.

- None of the articles focusing on the mixed-model U-shape line balancing problem has considered parallel workstations and zoning restrictions simultaneously.

- Six articles that focused on mixed-model U-shape line balancing problem used the absolute deviation of workloads (ADW) among workstations as performance measure (Sparling and Miltenburg (1998), Miltenburg (2002), Kim et al. (2000b), Kim et al. (2006), Kara et al. (2007a), Kara et al. (2007b)).

- Only one article (Kara et al., 2007b) that focused on mixed-model U-shape line balancing problem dealt with stochastic and all the others dealt with deterministic processing times.

Table 2.1 Evolution of the solution approaches for U-shape line

PUBLICATIONS	CHARACTERISTICS	METHODOLOGY
Miltenburg and Wijngaard (1994)	Single model, deterministic, type 1	dynamic programming & (RPWT) heuristic
Miltenburg (1998)	facility design, multiple U-line	dynamic programming
Sparling and Miltenburg (1998)	mixed model, deterministic , fixed number of station, adjusted task time, sequencing, horizontal balancing, workpace transportation	four-stage approximate solution algorithm
Ajenblit and Wainwright (1998)	single model, deterministic, type 1, vertical balancing	genetic algorithm
Urban (1998)	Single model, deterministic, type 1	integer programming
Scholl and Klein (1999)	single model, deterministic, maximize the line efficiency	branch-and-bound based heuristic (ULINO)
Kim et al. (2000b)	mixed model, deterministic , fixed number of station, sequencing, vertical balancing	artificial intelligence search technique (co-evolutionary algorithm)
Erel et al. (2001)	Single model, deterministic, type 1	simulated annealing based approach
Miltenburg (2002)	mixed model, deterministic, sequencing, horizontal balancing, vertical balancing	genetic algorithm
Aase et al. (2003)	Single model, deterministic, type 1	branch-and-bound procedures (U-OPT)
Guerriero and Miltenburg (2003)	single model, stochastic, type 1	mathematical model & recursive algorithm
Aase et al. (2004)	impacts on labor productivity	An experimental study
Martinez and Duff (2004)	Single model, deterministic, type 1	genetic algorithm with 10 heuristic rules
Gökçen et al. (2005)	Single model, deterministic, type 1	shortest route formulation

Table 2.1 (cont) Evolution of the solution approaches for U-shape line

Erel et al. (2005)	single model, stochastic, cost minimization	beam search-based heuristic
Gökçen and Ağpak (2006)	single model, deterministic, multi-criteria decision making	integer programming based goal programming
Kim et al. (2006)	mixed model, deterministic , fixed number of station, sequencing, vertical balancing	endosymbiotic evolutionary algorithm
Urban and Chiang (2006)	single model, stochastic, type 1	optimal piecewise-linear program
Kara et al. (2007a)	mixed model, deterministic, simultaneously line balancing/ model sequencing, type 1	simulated annealing algorithm based approach
Ağpak and Gökcen (2007)	single model, stochastic, multi-criteria decision making	four different chance-constrained binary integer programming model
Toklu and Özcan (2007)	single model, fuzzy time, multi-criteria decision making	fuzzy goal programming model
Baykasoğlu and Özbakir (2007)	single model, stochastic, type 1	multiple-rule-based genetic algorithm
Kara et al. (2007b)	mixed model, deterministic, stochastic, fixed number of station, type 1, sequencing, vertical balancing, multi-objective	multi-objective simulated annealing algorithm based approach
Boysen and Fliedner (2008)	single model, stochastic, profit maximization, parallel work stations and tasks, processing alternatives, zoning restrictions	versatile algorithm
Hwang et al. (2008)	single model, deterministic, type 1, vertical balancing	multi-objective genetic algorithm
Sabuncuoğlu et al. (2009)	Single model, deterministic, type 1	ant colony algorithm
Hwang and Katamaya (2009)	mixed model, deterministic , type 1, fixed model sequence, vertical balancing	genetic algorithm
Kara and Tekin (2009)	mixed model, adjusted task times, deterministic , type 1, given model sequencing, vertical balancing	mixed integer programming, comsoal algorithm based heuristic

CHAPTER THREE
BACKGROUND INFORMATION FOR SOLUTION METHODS: GENETIC ALGORITHM AND SIMULATED ANNEALING

3.1 Introduction

This thesis proposes Priority-Based Genetic Algorithm (PGA) that uses newly developed Simulated Annealing (SA) based fitness evaluation approach to solve simultaneously MMUL/BS problems. To gain a more comprehensive understanding, these two algorithms are explained in detail. The chapter is organized as follows. In Section 3.2, a brief introduction of basic concepts for GA is presented and in Section 3.3, a brief introduction of basic concepts for SA is presented.

3.2 Genetic Algorithms

Since 1950's, several evolutionary computation methodologies have emerged and have gained popularity. These include evolutionary programming, evolution strategy, genetic programming and genetic algorithm. Genetic Algorithm (GA) was firstly introduced by Holland (1975) and then, Genetic Algorithm has been applied to various types of problems. Genetic Algorithm is a stochastic search technique based on the process of natural selection and genetics. GA does not operate directly on the solution space. This requires a mapping mechanism between the solution space and the search space. Solutions are coded in strings, over a finite alphabet, called chromosomes or individuals. An encoding is selected in a way that each solution in the search space is represented by one *chromosome*. Each chromosome is then decoded according to a user defined mapping function, enabling the computation of the corresponding fitness value, which reflects the quality of the solution represented by the chromosome. The process of producing a phenotype from a genotype is as shown in Figure 3.1.

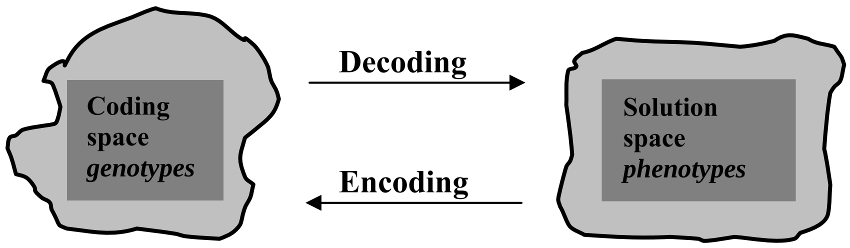


Figure 3.1 Mapping between solution space and search space (Rekiek and Delchambre, 2006)

In the original implementation of GA by Holland, each design variable is represented by a binary digit (see Figure 3.2) comprised of 1's and 0's. In later implementations integers or real-valued continuous values etc. have been introduced. Comprehensive literature review on chromosome representation scheme for assembly line balancing problems is provided in Scholl and Becker (2006).

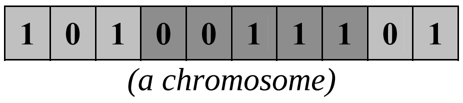


Figure 3.2 The binary encoding scheme

In general, a GA has seven basic components; coding of solutions, population, fitness function, selection scheme, genetic operators, i.e., crossover and mutation, survival scheme, termination criteria. Figure 3.3 illustrates the general working principle of GAs.

GA operates with a collection of chromosomes, called a *population*. Genetic Algorithm then starts with initialization which is done by random generation, so it starts with large search space to make sure that it does not become stuck in a local suboptimal point. Indeed, most solutions are largely different and belong to different areas of the search space. Over time, the population begins to converge, with the separate individuals resembling each other more and more. The GA narrows its search in the solution space and reduces the changes made by evolution until eventually the population converges to a single solution (Rekiek and Delchambre, 2006). On one side, if population size is too small, the search space will not be sufficient and will lead the search to premature coverage. On the other side, if it is too big, the search will be inefficient and the solution will not be found within a reasonable computation time (Sastry and Goldberg, 2005). Choosing an appropriate population size is always a trade-off between solution quality and execution time.

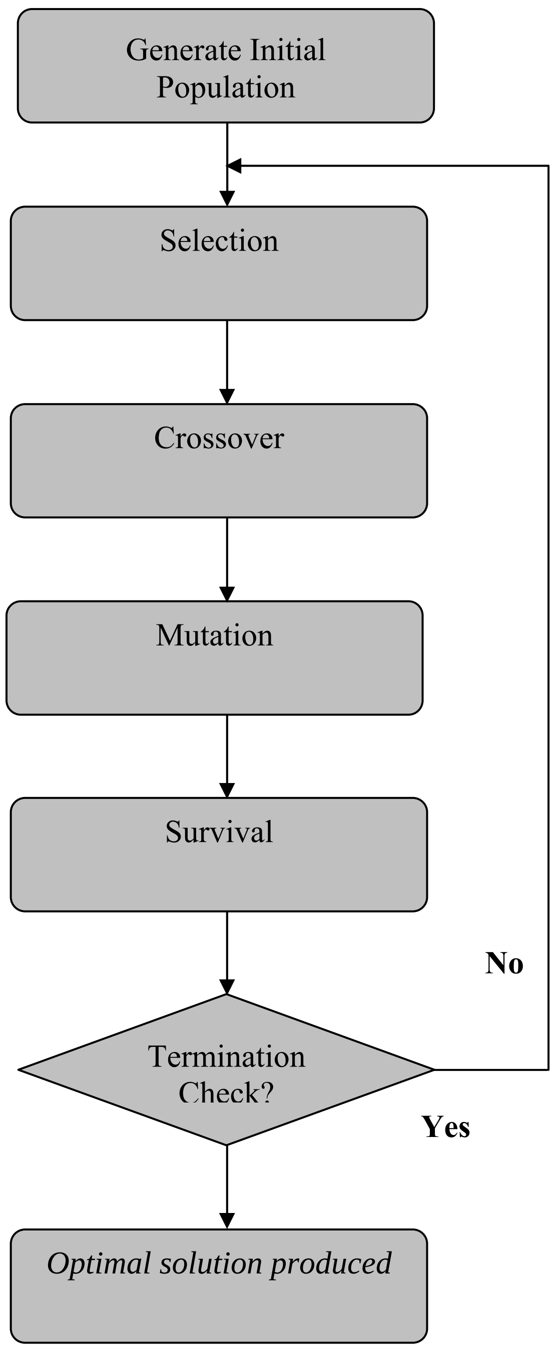


Figure 3.3 Main steps of a generalized genetic algorithm (Grupe and Jooste, 2004)

Fitness function in GA is the value of the objective function for its phenotype. Each individual represents a potential solution to a problem. The fitness function assigns a real number as a measure of fitness to each solution.

Selection is the "survival of the fittest" operator in a genetic algorithm. This operator determines which designs from the population will survive to form the 'parents' of the next generation. The selection operator is the mechanism so that it establishes which individuals are best adapted to the fitness landscape and should have their genes advanced to future generations. Individuals that are more fit to the design 'environment' will be more likely to survive and pass on their traits. This procedure is analogous to natural selection as described by Darwin. Using the fitness values, the selection scheme is executed to choose the individuals from a population for breeding offspring. Individuals in the original population are selected for reproduction. Two popular selection methods are the roulette wheel and tournament. The roulette wheel gives individuals a chance of selection which is equal to their fitness relative to the population. Tournament selection randomly pits k individuals (k >= 2) against each other, with the winner contributing to the next generation. An additional method often used is random selection, which is completely arbitrary.

The genetic operations mimic the process of heredity of genes to create new offspring at each generation. GA uses two operators to generate new solutions from existing ones: *crossover* and *mutation. Crossover* is a process of breeding new offspring by the selected individuals from the selection operator. These individuals are combined into pairs to exchange genetic information and produce new individuals. A two-point crossover exchanges all genes between the cut-points, which are randomly determined in general (see Figure 3.4). The aim of crossover is to transmit good characteristics from parents to offspring.

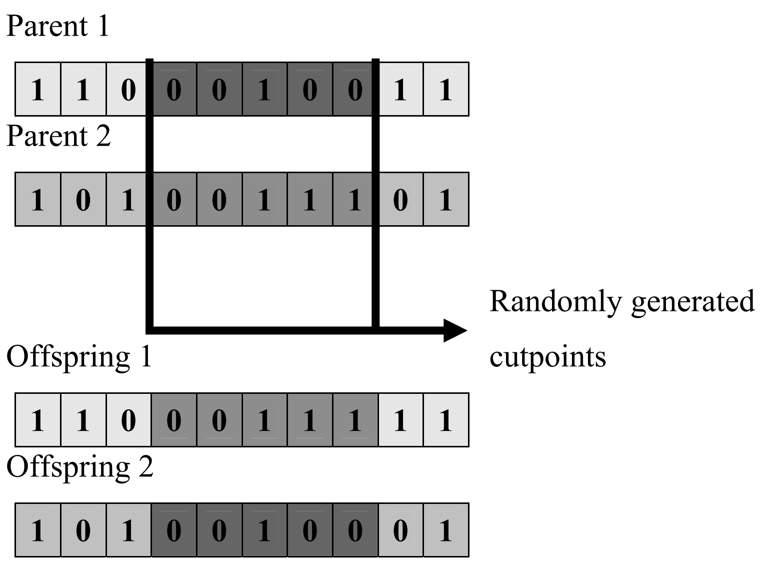


Figure 3.4 Two point crossover

Mutation is a background operator which produces spontaneous random changes in various chromosomes. Mutation represents new discovery in the new search space. Figure 3.5 shows the simplest mutation, which is performed by changing the value of a randomly selected gene from 0 to 1 (or from 1 to 0) in a binary string. In GAs, mutation serves the crucial role of either (a) replacing the genes lost from the population during the selection process so that they can be tried in a new context or (b) providing the genes that were not present in the initial population (Gen and Cheng, 1997).

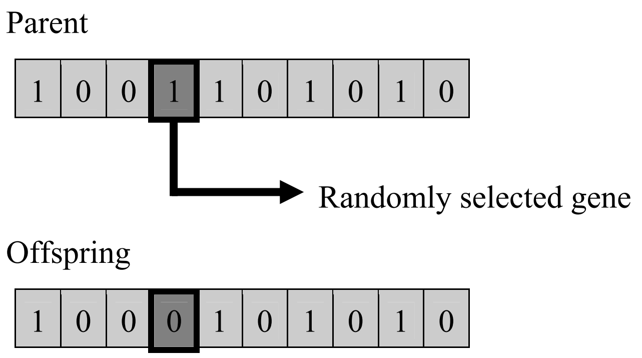


Figure 3.5 Mutation

A *replacement (survival)* strategy is necessary to determine which individuals stay in the population and which are replaced by offspring. The most common replacement approach is *elitism*, which allows the best chromosome in each generation to survive in the next generation, thus guaranteeing that the final population contains the best solution ever found.

For *termination criteria*, the various stopping condition are listed as follows (Sivanandam and Deepa, 2008);

Maximum generations: The genetic algorithm stops when the specified numbers of generations has evolved.

Elapsed time: The genetic process will end when a specified time has elapsed.
Note: If the maximum number of generation has been reached before the specified time has elapsed, the process will end.

No change in fitness: The genetic process will end if there is no change in the population's best fitness for a specified number of generations.
Note: If the maximum number of generation has been reached before the specified number of generation with no changes has been reached, the process will end.

Stall generations: The algorithm stops if there is no improvement in the objective function for a sequence of consecutive generations of length Stall generations.

Stall time limit: The algorithm stops if there is no improvement in the objective function during an interval of time in seconds equal to stall time limit.

The procedure of a generic GA (Goldberg, 1989) is given as follows:

Step1: *Set $t = 1$. Randomly generate N solutions to form the first population, P_1 and evaluate the fitness of solutions in P_1.*

Step2: *Crossover: Generate an offspring population Q_t as follows:*

2.1. *Choose two solutions x and y from P_t based on the fitness values.*

2.2. *Using a crossover operator, generate offspring and add them to Q_t.*

Step3: *Mutation: Mutate each solution $x \in Q_t$ with a predefined mutation rate.*

Step4: *Fitness assignment: Evaluate and assign a fitness value to each solution $x \in Q_t$ based on its objective function value and infeasibility.*

Step5: *Selection: Select N solutions from Q_t based on their fitness and copy them to $P_t + 1$.*

Step6: *If the termination criterion is satisfied, terminate the search and return to the current population, else, set $t = t + 1$ go to Step 2.*

3.3 Simulated Annealing Algorithms

The SA algorithm was introduced by Kirkpatrick et al. (1983) to solve NP-hard combinatorial optimization problems, by using the analogy with the simulation of the physical annealing of solids, in order to minimize/maximize the value of an objective function. The simulated annealing meta-heuristic is based on the analogy with the thermodynamic annealing process. Thermodynamic annealing is a method in metallurgy to reduce the defects in a metal, alloy, or other material by heating them to a very high temperature and then having a controlled cooling. This causes molecules with high-energy state to move randomly in their neighborhood to find a configuration with lower energy state than the current energy state. The result of this process is to have an ordered crystalline structure. If the metal is cooled too slowly or too fast, defects will be formed in the metal, which in our case represents local minimum or maximum. Similarly, in simulated annealing (SA) heuristic it is important to have a proper cooling schedule. The procedure for the SA process is defined as below:

```
Create a random initial solution $S_0$
Determine initial temperature $T_0$, crystallization temperature $T_{cry}$, iteration length at
each level of energy state $IT$ and cooling rate $q$
$T_c = T_0$, $S = S_0$
Repeat
For ( i = 1 to IT )) {
Generate a random solution from $S$ to $S'$
$\Delta = evaluate(S') - evaluate(S)$
If ( $\Delta \le 0$ ) then $S = S'$
```

```
Else {
If (Perform probability ($\exp(-\Delta / T_c)$) that S' is still accepted solution)
$S = S'$
} End Else
} End For
Reduce temperature $T_c = T_c \times q$
Until ($T_c \geq T_{cry}$)
Optimal Solution = $S$
```

It starts from an initial solution to the problem, S_0 and a control parameter, T_c, which is set to an initial temperature value, T_0. During the algorithm, the value of T_c is systematically decreased according to an annealing schedule as shown in Figure 3.6. In this schedule the following issues are defined: a temperature reduction function, q, and the length of each temperature level, IT, that determines the number of solutions generated at a certain temperature level.

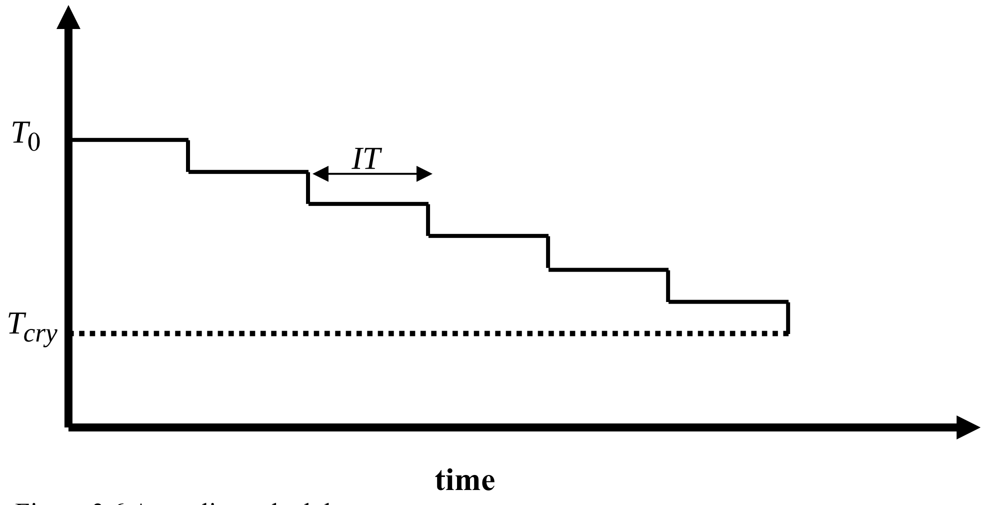


Figure 3.6 Annealing schedule

For each iteration, neighboring solutions, S', the current solution are generated and the value of the objective function is calculated. If the value is better than the current solution, the neighboring solution becomes the new current solution. On the other hand, if the neighboring solution provides an objective function value inferior to that of the current solution, the neighboring solution may still become the current solution according to certain acceptance probability. The acceptance probability p is computed according to the criterion established by Metropolis (Metropolis et al., 1953) as follows:

$$p = \exp(-\Delta / T_c)$$

where; Δ = change in the objective function.

A random number between zero and one is generated, if the random number is smaller than p the solution is accepted. This strategy prevents the algorithm from getting trap in a local optimum.

CHAPTER FOUR
PROPOSED GENETIC ALGORITHM BASED APPROACH FOR SIMULTANEOUSLY SOLVING U-SHAPE MIXED-MODEL ASSEMBLY LINE BALANCING AND SEQUENCING PROBLEM

4.1 Chapter Introduction

In chapter four, simultaneous solution of balancing/sequencing (MMUL/BS) problems in mixed model U-shaped assembly lines are addressed, and this chapter is divided into 3 sub-titles in general.

Firstly, general characteristics of U-shaped production lines are presented, and differences of mixed-model U-shaped assembly line from other lines are explained in detail.

And then, the problem is presented with notations and equations for being able to understand better the general characteristics of the problem. Fitness function aims at minimizing the number of stations as primary objective and workload balancing between-within workstations at the end of all cycles as secondary objective is mathematically presented, and our proposed solution method based on genetic algorithm for the solution of the problem is introduced. In our solution method based on priority based-genetic algorithm, simulated annealing based fitness evaluation that we developed for carrying out fitness assessments is improved. Experimental design is established in order to ensure the execution of our proposed algorithm with more efficient parameters. These new procedures are illustrated with a numerical example and its performance is tested through a set of test problems with the generated minimum part sets (MPS).

Finally, the problem is expanded in a manner comprising parallel station and zoning constraints. This new case is showed by notations – equations. Our fitness function is mathematically expressed in a manner comprising this new case.

Moreover, these procedures are illustrated with a numerical example and its performance is tested through a set of test problems with the generated MPS.

4.2 Characteristics of U-shaped Assembly Lines

In recent years, many manufacturers have adopted a Just-in-Time (JIT) approach for manufacturing, finding that it improves their productivity, profits, and product quality. Straight assembly lines have been an important part of traditional mass production while U-shaped assembly lines have been emerged as a consequence of continuous improvement and cost reduction efforts of just-in-time (JIT) production (Monden, 1993). One of the important changes resulting from JIT implementation is the replacement of the traditional straight lines with U-shaped production lines. The reason for this is that JIT use of multi-skilled workers and efficient facility layouts, so many companies are rearranging their traditional straight assembly lines into a U-shaped layout (Scholl and Klein, 1999, Aase et al., 2004). The UALB is more complex than the SALB because tasks can be assigned by moving forward, backward, or simultaneously in both directions through the precedence diagram (Scholl and Klein, 1999).

Miltenburg and Wijngaard (1994) and Cheng et al. (2000) summarized the major benefits and factors of U-shaped assembly lines and explain its popularity among JIT practitioners. The major benefits and factors of U-lines are as followings:

- *Volume Flexibility:* As a consequence of just-in-time principles, the output from a U-line may need to be adjusted from time to time for matching the rate at which the produced parts are consumed by subsequent operations. The production rate as required in this situation on U-line can be adjusted by adding or removing workers. This level of volume flexibility is harder to obtain with a straight line because of the fact that rebalancing is more difficult on a traditional "straight line" with its narrowly trained operators.
- *Operator Flexibility:* Since they rotate through many stations in the U-line each day, it is easier for one operator to oversee several work centers. Hence, operators are involved in different parts of the assembly process; they can

easily enlarge their skills. So, they can respond to the problems quickly. Also, the acquisition of multiple skills leads to higher motivation, improved product quality and increased flexibility.

- *Number of Workstations:* The number of workstations required on a U-line is never more than, and is sometimes less than, that required on a straight line. The reason of this is that there are more possibilities for grouping tasks into workstations on a U-line.
- *Material Handling:* A U-line eliminates the need for special material-handling equipment such as conveyors and special material-handling operators. Instead, production operators move products from machine to machine.
- *Visibility and Teamwork:* The compact size of a U-line improves visibility and communication. This enhances teamwork, gives a greater sense of belonging, and increases responsibility and ownership compared to a straight line, where operators are spread out along a long line and may be separated by walls of inventory.
- *Rework:* A tenet of the total quality management (TQM) is quality at the source, which calls for correcting quality problems as soon as possible after they occur by returning a defective product to the station where it was produced. In a U-line, the distance to return the defective product is short, making it easier to follow this tenet. This is in contrast to the traditional policy of sending the defective product to a separate rework area.

Miltenburg (2001) presented a review of the theory and practice on U-shaped production lines on a set of US and Japanese companies which changed their straight lines to U-lines. The results showed that the adoption of U-shaped lines leads to remarkable benefits: productivity improvement of 76%, decreasing of work-in-process inventory 86%, decreasing of lead time 75% and dropping of defective rates 83%, on average.

4.3 Problem Statement of the MMUL/BS

The problem of balancing a U-shaped assembly line to produce a set of models of a product is the mixed-model U-ALBP (U-MALBP), and it was first described by Sparling and Miltenburg (1998). The key difference between the single model assembly line balancing problem and the mixed-model assembly line balancing problem is that more than one product models are produced on mixed-model assembly lines while only one product model is produced on single model assembly lines. An additional and very important issue of mixed-model U-lines, when compared to single-model U-lines, is that a workstation may perform its tasks in the same cycle in two different models, one at each leg of the line. A successful implementation of a mixed-model U-line requires solutions for two important problems, called mixed-model U-line line balancing (MMUL/LB) and mixed-model U-line model sequencing (MMUL/MS). These two problems are tightly interrelated with each other and can not be set independently (Sparling and Miltenburg, 1998, Kim et al., 2000b and Kara et al., 2007a). The balancing problem, MMUL/LB, is the assigning tasks to an ordered sequence of workstations on the mixed-model U-line in such a way that some performance measures are optimized. The sequencing problem, MMUL/MS, is the determining the production sequence of models produced on the mixed-model U-line. It is NP-hard. For example, a small U-line of 8 tasks producing 2 models with demands of 6 and 4 units each could have more than $8!(6+4)!/(6!4!)=8.5\times10^{6}$ solutions. Even the problem of determining the number of feasible solutions is NP-hard (Miltenburg, 2002).

Miltenburg (2002) presented two observations about the reason why the balancing and sequencing problem can not be set independently when JIT principles are being used are as in the followings:

- The sequence in which different models are produced cannot be set independently of the line balance (i.e., the assignment of tasks to stations). Different models require different tasks and the same tasks have different completion times for different models. *On a U-line two different models may* *be worked on in the same station in the same cycle. On a straight line, only one model is worked on in each station in each cycle.*

- The sequence in which the different models are produced on a U-line cannot be set independently of the schedules of other lines and production facilities when JIT principles are being used. JIT uses a *pull* rather than a *push* system of production control, which means that model sequence at the U-shaped mixed-model final assembly line sets the schedules at the other production facilities. Most often, JIT requires these latter schedules to be "level", and this imposes additional constraints on model sequence.

Figure 4.1 illustrates the Mixed-Model Production on a U-Shaped Assembly Line. The line produces three models in the sequence ABC. At the first cycle; the operator of workstation 1 (w-1) performs tasks 1 and 2 on model C at the front of the line and then crosses to the back to complete tasks 13 and 14 on model A, the operator of workstation 2 (w-2) performs tasks 3 and 4 on model B at the front of the line and then crosses to the back to complete task 12 on model B, the operator of workstation 3 (w-3) performs tasks 5,6,7 and 8 on model A and lastly the operator of workstation 4 (w-4) performs tasks 9,10 and 11 on model C. This line consists of 4 workstations and 6 regions where the models are processed at workstations.

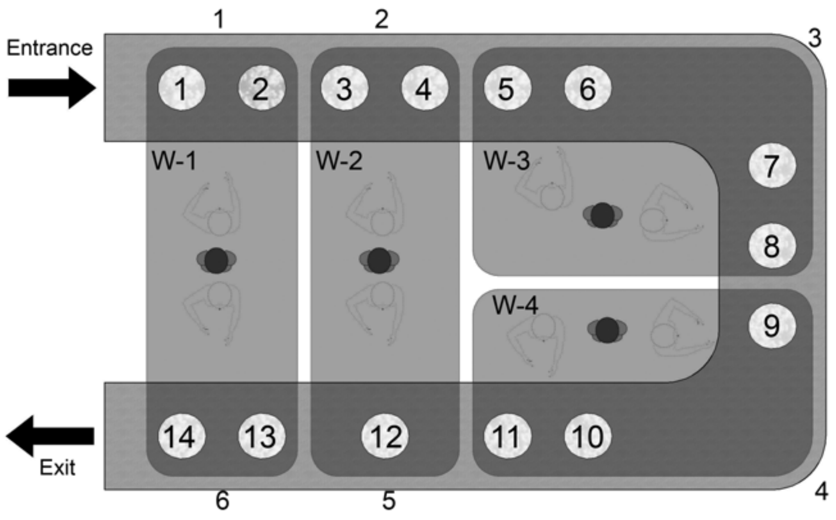


Figure 4.1 Mixed-model productions on a U-shaped assembly line

4.3.1 Model Assumptions

The U-lines considered in this study operates under the following assumptions:

- Product models having similar production attributions are produced on the same U-shaped production lines.
- Zoning restrictions and parallel workstations are not allowed.
- The travel times of operators and setup times are ignored.
- Precedence diagrams of different models are known, and a combined precedence diagrams is employed (Macaskill 1972).
- The completion times of tasks may differ from one model to another and can be equal to zero. Common tasks among different models exist.
- Task completion times are deterministic and independent from each others.
- Paced assembly line considered and no work-in-process is allowed.
- Minimum Part Set (MPS) principle is used (Bard et al., 1992, Merengo et al., 1999, Kara et al., 2007a, Kim et al., 2000b).
- Equally equipped stations and fixed rate launching are considered.

4.3.2 Notation and Equations

The following notation and equations will be used to describe the problem characteristics:

N	Total numbers of tasks are performed in a set of workstations ($i = 1,2,...,N$)
K	Number of workstations utilized on the mixed-model U-line ($k = 1,2,...,K$)
P	The planning horizon has a fixed length
M	Number of different models produced on the MMUL ($m = 1,2,...,M$)
X_k	Set of tasks assigned to workstation k
T	Set of tasks performed on the mixed-model U-line
PR	Set of merged precedence constraints for all models
MS	Model sequence of the mixed-model U-line

D The vector that represents the total demand for each model $D = \{D_1, D_2, ..., D_M\}$

cd The greatest common divisor of the elements of D

d_m Over the planning horizon, the forecast demand for model m for one MPS

$$d_m = D_m / cd \text{ , } (m = 1,2,...,M) \tag{1}$$

MPS Minimum part set, calculated by dividing the total demands of the models by the greatest common divisor of these demands $MPS = \{d_1, d_2, ..., d_m\}$

R The length of the model sequence for one MPS

$$R = \sum_{m=1}^{M} d_m \tag{2}$$

R Also represent the number of the possible cycle ($r = 1,2,..., R$)

C Cycle time $C = P / R$ (3)

t_{im} The required time to perform task i on the model m

IP_i Set of the immediate predecessors of task i

CS_r Depicts the appearance of the models in model points at the cycle r

S_{kr} Idle time of workstation k at the cycle r

KI_k Total idle time of all cycles in the workstation k

RI_r Total idle time of all workstations at the cycle r

$C_{\min}$ Theoretical minimum cycle time (Kim et al., 2000b)

$$C_{\min} = (1/(K \times R))\sum_{i=1}^{N}\sum_{m=1}^{M} d_m t_{im} \tag{4}$$

For a given line balance and model sequence:

XF_k Set of tasks in workstation k located on the front of the U-line

XB_k Set of tasks in workstation k located on the back of the U-line

f_k^r Model produced on the front of station k at the cycle r in the sequence

b_k^r Model produced on the back of station k at the cycle r in the sequence

W_{kr} Amount of work assigned to station k at the cycle r

$$W_{kr} = \sum\nolimits_{i \in XF_k} t_{if_k^r} + \sum\nolimits_{i \in XB_k} t_{ib_k^r} \quad (k = 1,2,...,K), (r = 1,2,...,R) \tag{5}$$

The following mathematical model is only used as a means to formally describe the problem, as its high complexity makes it impossible to be solved to optimality. The model is based on the model of Sparling and Miltenburg (1998) developed for MMUL/LB. The model was modified and expanded to include model sequencing by Kara et al. (2007a) and is shown below.

$$\bigcup_{k=1}^{K} X_k = T \tag{6}$$

$$X_u \bigcap_{u \neq v} X_v = 0 \tag{7}$$

$$W_{kr} \leq C \quad (k = 1,2,...,K), (r = 1,2,...,R) \tag{8}$$

For every task $y \in T$: (9)

either: if $(i, y) \in PR$, $i \in X_u$, $y \in X_v$, then $u \leq v$ for all i

or : if $(y, z) \in PR$, $y \in X_v$, $z \in X_x$, then $x \leq v$ for all z

$$\bigcup_{r=1}^{R} f_k^r \equiv MPS \quad (k = 1,2,...,K) \tag{10}$$

$$\bigcup_{r=1}^{R} b_k^r \equiv MPS \quad (k = 1,2,...,K) \tag{11}$$

The constraint in Eq. 6 ensures that all tasks are assigned to a workstation. The constraint in Eq. 7 ensures that each task is assigned to only one workstation. By the constraint in Eq. 8, each workstation capacity does not exceeded the predetermined cycle time, C. The constraint in Eq. 9 ensures that a task can be assigned to a workstation if either all its predecessors or all its successors have been assigned to the same or to an earlier workstation and the last two constraints (Eq. 10 and Eq. 11) ensure that all workstations are visited by all models for which the demands are met.

4.3.3 New Objective Function

A new task can be assigned to a workstation as long as the sum of total processing time does not exceed the cycle time at each cycle. In a U-shape production system, the workload of a workstation will depend on the models performed on the front and the back of the line at each cycle. Then, finding the best model sequence (the sequence in which the models are launched to the line) is more important in order to allow a good workload balance. Due to the mixed-model nature of the problem, in the MMUL/BS, this issue becomes even more important (models may be located to different locations (back front or in front of the line) at each cycle by satisfying the assignment constraints and the given model sequence (MS)).

The main goal of ALBP of type I is to minimize the number of workstations for a given cycle time. In addition to main goal, different performance measures must also be taken into consideration. Vilarinho and Simaria (2002) introduced a performance measure that provides equally distribution of the workload between and within workstations for the straight mixed-model assembly line balancing. In this study, a new fitness function based on Vilarinho and Simaria's (2002) objective function is developed to the U-shape mixed-model assembly line balancing for aiming at minimizing the number of workstations as primary goal and smoothing the workload between - within workstations at the end of all cycles as secondary goal.

The new fitness function is explained by following equations:

$$S_{kr} = (C - W_{kr}) \ (k = 1,2,...,K), (r = 1,2,...,R) \tag{12}$$

$$KI_k = \sum_{r=1}^{R} S_{kr} \ (k = 1,2,...,K) \tag{13}$$

$$C_b = \frac{R}{K(R-1)} \sum_{k=1}^{K} \sum_{r=1}^{R} \left(\frac{S_{kr}}{KI_k} - \frac{1}{R} \right)^2 \tag{14}$$

$$RI_r = \sum_{k=1}^{K} S_{kr} \quad (r = 1,2,...,R) \tag{15}$$

$$C_w = \frac{K}{R(K-1)} \sum_{r=1}^{R} \sum_{k=1}^{K} \left(\frac{S_{kr}}{RI_r} - \frac{1}{K} \right)^2 \tag{16}$$

$$\min \mathrm{Z} = \mathrm{K} + C_b + C_w \tag{17}$$

S_{kr} (Eq. 12) represents the idle time of workstation k at the cycle r. The idle time of a workstation is the difference between the capacity of the workstation and its workload.

KI_k (Eq. 13) represents the total idle time at the end of all cycles in the workstation k.

In the objective function (Eq. 17), C_b (Eq. 14) aims at smoothing the workload of workstations between the cycles, i.e., the idle time is distributed across all cycles as equally as possible for any workstation. The value of function C_b varies between a maximum of 1, when the total idle time of a workstation at the end of all cycles equal to only one cycle's idle time, and a minimum of 0, when the idle times of a workstation at the each cycle are equal to each other.

RI_r (Eq. 15) represents the total idle time of all workstations at the cycle r.

In the objective function (Eq. 17), C_w (Eq. 16) aims at workload balance of all workstations within any cycles, i.e., the idle time is distributed across all workstations as equally as possible at any cycle. The value of function C_w varies between a maximum of 1, when the total idle times of all workstations at any cycle equal to only one workstation's idle time, and a minimum of 0, when the idle times of each workstation at any cycle are equal to each other.

The first term (K) of the fitness function (Eq. 17) is to minimize the number of the workstations required on the line. The second term (C_b) is to smooth the workloads of workstations between the cycles. The third term (C_w) is to smooth the workloads of workstations within the cycles. The second and the third terms are within the value range [0, 1]. So, the model minimizes the number of workstations, before the secondary goal becomes active. The proposed performance measure may vary depending on the balance and the model sequence.

Note:

If KI_k equals to 0 , $\sum_{r=1}^{R}\left(\frac{S_{kr}}{KI_k}-\frac{1}{R}\right)^2$ will be equal to 0.

If RI_r equals to 0, $\sum_{k=1}^{K}\left(\frac{S_{kr}}{RI_r}-\frac{1}{K}\right)^2$ will be equal to 0.

4.3.4 Proposed GA-Based Approach

The workloads of workstations in MMULs depend on various factors; therefore development of solution procedures for MMUL is more complex than that of other types of line balancing (Kara and Tekin 2009). The NP-hard nature of this problem expedites the development of computer-aided effective heuristic solution procedures. Nowadays, evolutionary approaches have been developed for solving this problem.

The GA combines the exploitation of past results with new areas of space search exploration. Using survival of the fittest techniques, combined with a structured yet randomized information exchange, a GA can mimic some of the innovative attributes of a human search (Hwang et al.2007).

Hwang et al. (2007) have studied the assembly line balancing problem with genetic algorithms and have used the priority-based encoding method to treat the precedence constraints efficiently, but they only considered single model assembly lines and more recently, Hwang and Katamaya (2009) have studied the mixed-model U- line balancing problem without a job sequence.

In this thesis, we used the priority-based genetic algorithm (PGA) (Gen and Cheng 2000) for simultaneously solving the balancing and sequencing problems of mixed-model U-lines. The simulated annealing algorithm (SA) based fitness evaluation approach is developed and inserted into PGA to evaluate the fitness value of the task sequences TS that will discussed in Chapter 4.3.5.

The newly developed Simulated Annealing Algorithm based fitness evaluation approach is the most significant property of proposed algorithm. This enables us to consider the line balancing/model sequencing problems of mixed-model U-lines simultaneously.

In this new approach, our fitness function developed is capable of comparing the performances of the task sequences that have been used. As mentioned before, this new fitness function minimizes the number of stations as primary goal, and provides the workload balance within station and between stations by considering all possible cycles.

Genetic algorithms and its main concepts have been previously characterized in Chapter 3. The main steps of the proposed genetic algorithm are presented in Figure 4.2.

The general specifications of proposed GA approach are summarized in the following subsections.

4.3.4.1 Selected Chromosome Representation and Initialization of Population

We used the priority-based encoding method (Gen and Cheng 2000) for working effectively with precedence constraints. The position of a gene was used to represent a task node, and the value of the gene was used to represent the priority of the task node for constructing a task sequence among candidates (Hwang et al., 2008).

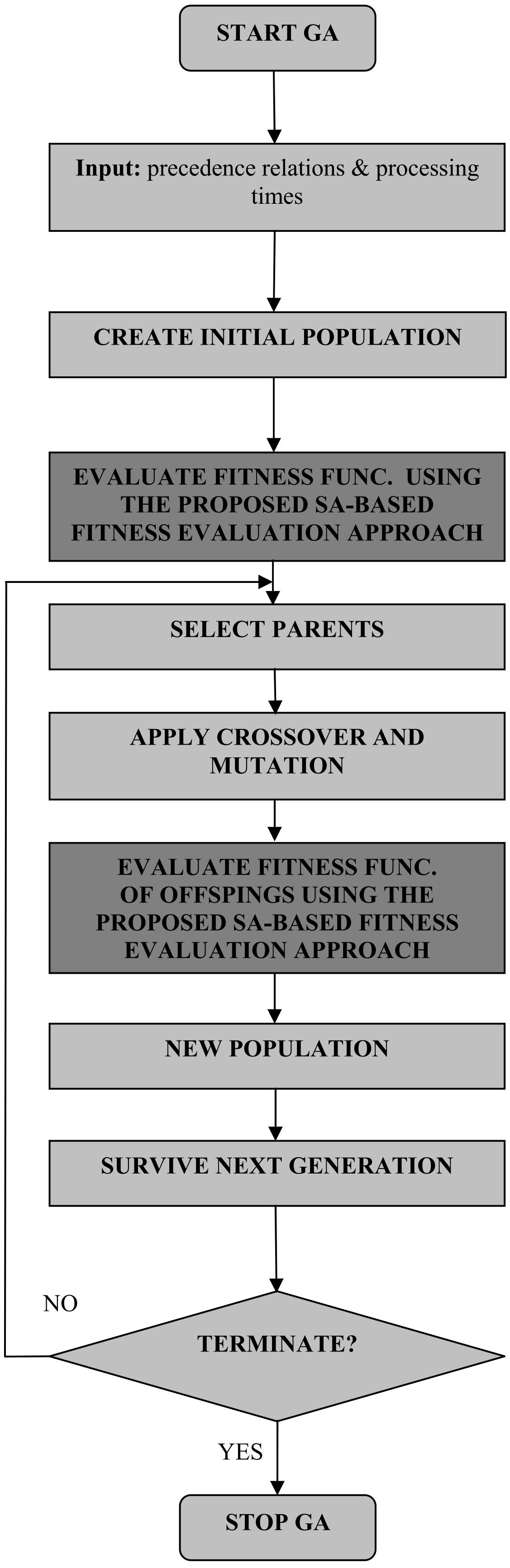

START GA
Input: precedence relations & processing times
CREATE INITIAL POPULATION
EVALUATE FITNESS FUNC. USING THE PROPOSED SA-BASED FITNESS EVALUATION APPROACH
SELECT PARENTS
APPLY CROSSOVER AND MUTATION
EVALUATE FITNESS FUNC. OF OFFSPINGS USING THE PROPOSED SA-BASED FITNESS EVALUATION APPROACH
NEW POPULATION
SURVIVE NEXT GENERATION
NO
TERMINATE?
YES
STOP GA

Figure 4.2 General structure of the proposed approach

The initial chromosome is generated randomly as shown in procedure 1 (Figure 4.3). Each chromosome position is called a gene. Each gene will use the priority of nodes in an assembly network. This encoding method easily verifies any permutation of the encoding to correspond to the sequences, so that most existing genetic operators can easily be applied to the encoding (Hwang et al., 2008).

Each chromosome is created by using the priority-based encoding method, and then the priority-based decoding (Hwang et al., 2008) procedure 2 ensuring the use of precedence relations to obtain feasible chromosomes (Figure 4.4). The priorities of eligible nodes with the highest priority are placed into the task sequence (TS). So, infeasible solutions violating the precedence constraints are not allowed.

```
Procedure 1: priority-based encoding (initial chromosome)
Input: number of tasks $N$
Output: chromosome $[v(i)]$
begin
for $i = 1$ to $N$
$v(i) \leftarrow (1, N)$;
for $i = 1$ to $[N/2]$
$i \leftarrow random(1, N)$;
$k \leftarrow random(1, N)$;
if $i \neq k$ then
swap $\{v(i), v(k)\}$;
output the chromosome $[v(i)]$
```

Step 0: Input the priority number

task ID i:	1	2	3	4	5	6	7	8	9	10
priority $v(i)$:	1	2	3	4	5	6	7	8	9	10

Step 1: Swapping two nodes randomly

task ID i:	1	2	3	4	5	6	7	8	9	10
priority $v(i)$:	6	2	3	4	5	1	7	8	9	10

Step 2: Output priority based-chromosome

task ID i:	1	2	3	4	5	6	7	8	9	10
priority $v(i)$:	6	5	3	4	9	1	10	8	7	2

Figure 4.3 Priority-based encoding procedures

```
**Procedure 2: priority-based decoding**
**Input:** number of tasks $N$, chromosome $v(.)$, the set of task nodes
**Output:** task sequence $TS$
**Begin**
$\bar{N} \leftarrow \phi, TS \leftarrow \phi$;
$N \leftarrow 0, i \leftarrow 0$;
**while** $(i \le N)$ **do**
    $\bar{N} \leftarrow suc(i)$;
    $\bar{N} \leftarrow pre(i)$;
    $i^* \leftarrow \arg\max\{v(i) \mid i \in \bar{N}\}$;
    $\bar{N} \leftarrow \bar{N} \setminus i^*$;
    $TS \leftarrow TS \cup i^*$;
    $i \leftarrow i^*$;
**end**
**output** task sequence $TS$
```

Figure 4.4 Priority-based decoding procedures

task ID i :	1	2	3	4	5	6	7	8	9	10
priority $v(i)$:	6	5	3	4	9	1	10	8	7	2

Figure 4.5 An example output of priority-based encoding method

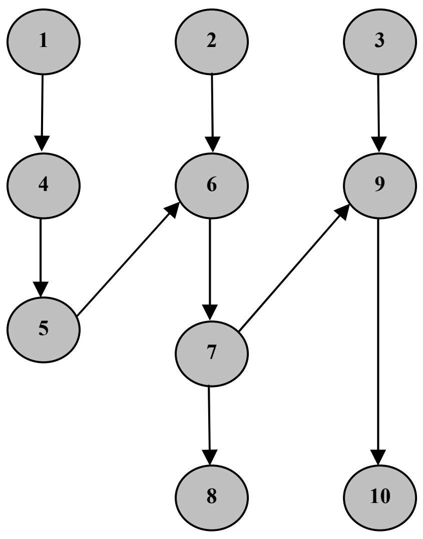


Figure 4.6 Combined precedence diagram (Kara et al., 2007a)

In this encoding method, the position of a gene is used to represent a task node, and the value of the gene is used to represent the priority of the task node for constructing a task sequence among candidates. An example output of this encoding method is given in Figure 4.5. Then, the generated chromosomes are converted to the feasible task sequences by using priority-based decoding method. In Figure 4.6, the nodes eligible to be assigned to the first position of task sequence are 1, 2, 3, 8 and 10. The priorities of nodes are 6, 5, 3, 8 and 2 respectively, so the task 8 having the highest priority is placed into the task sequence TS. For the second position the possible nodes are 1, 2, 3 and 10 which have priorities 6, 5, 3 and 2 respectively. So, task 1 is placed next into the task sequence TS. These steps are repeated until we obtain a complete task sequence TS = {8, 1, 2, 4, 5, 3, 10, 9, 7, 6}, and these procedures continue as long as the initial population is generated.

4.3.4.2 Selected Selection Scheme

Roulette wheel selection scheme (Holland, 1975) is used, which is a method for reproducing a new generation proportional to the fitness of each individual. In this procedure, the fitness values of the members scales within the population so that the sum of the rescaled fitness values equals to 1. To select a parent, a uniform random number within the interval (0, 1) is generated firstly (wheel is spun), and then the member whose cumulative rescaled fitness value is greater than the generated number is selected as parent.

4.3.4.3 Selected Genetic Operators

There are two kinds of genetic operators, i.e., crossover and mutation. Crossover is the operation by which two individuals in the current population create offspring for the next population. Mutation operator is used to change randomly the value of single genes within chromosomes. We used weight mapping crossover (WMX) (Hwang et al. 2006) and swap mutation.

4.3.4.3.1 Crossover Operator. Here the position-based *crossover* operator can be viewed as a two-point crossover of a real number string by weight mapping crossover (WMX) that we used and a remapping by order of different real number strings is shown in Figure 4.7 (Hwang et al. 2006).

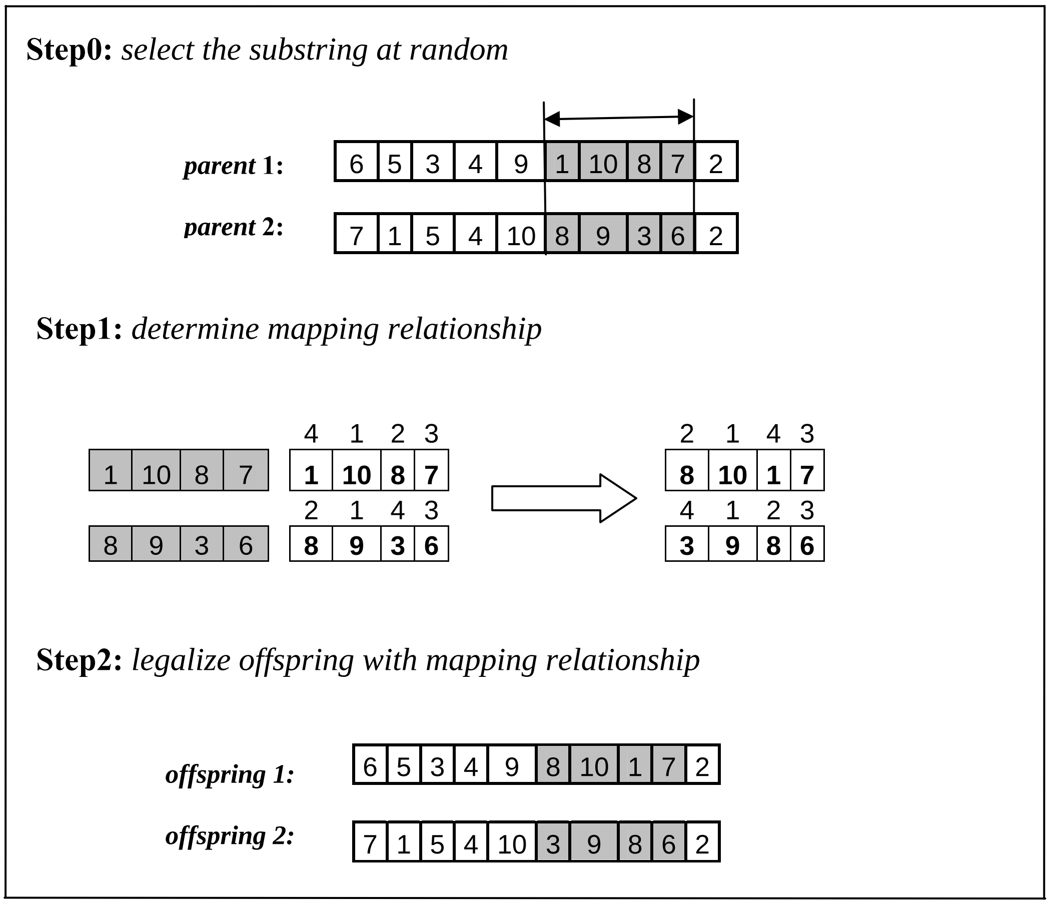


Figure 4.7 Two point-based weight mapping crossover (WMX)

4.3.4.3.2 Mutation Operator. We used the swap mutation operator, in which two positions are selected randomly, and their contents are swapped (see Figure 4.8).

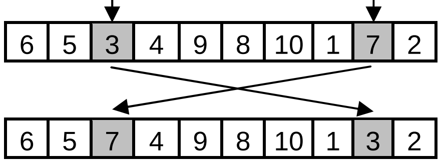


Figure 4.8 Swap mutation operator

4.3.4.4 Selected Survival Scheme

Survival is an essential process in GAs that removes individuals with a low fitness and drives the population towards better solutions. The transfer of the best solution of the previous population to the next one is carried out for continuous survival of the best solution.

4.3.4.5 Selected Termination Criteria

The genetic algorithm stops when the specified numbers of generations have evolved ($GN_{\max}$).

4.3.5 The Proposed Simulated Annealing Algorithm Based Fitness Evaluation Approach

In general, the assignment of tasks to workstations benefits from a single-pass decision rule procedure using a feasible task sequence as a basis. Tasks are listed in a sequence, and then an attempt is made for assigning them to workstations in that sequence (Hwang and Katamaya 2009). If total time of the tasks previously assigned to a workstation and new task to be assigned is smaller than cycle time or equal to cycle time, the said task can be assigned to this station. When the cycle time is exceeded, a new workstation can be opened and this new task is assigned to the new workstation. This procedure continues until the assignment of all tasks in task sequence to workstation is completed.

This simple procedure is not valid for being able to solve mixed-model U-lines under varying model sequence. Because u-shaped lines may include both traditional and crossover workstations as a general characteristic, it cannot be known which task will take which model time.

In order to optimize a certain performance measure, it is also very difficult to make these assignments by considering all different cases that can be formed under fixed model sequence and fixed number of stations only if at least one task is assigned to each workstation. Especially, as the number of tasks increases, the number of cases that must be taken into consideration increases exponentially. Furthermore, none of these different cases may give a feasible solution.

Suppose that we will assign a feasible task sequence (TS={1,8,2,3,4,6,5,7}) obtained from the precedence diagram in Figure 4.9 and consisting of 8 tasks (N=8) to 3 stations (K=3). Twenty-one ($(C(N-1, K-1)$) different cases are formed in a manner that at least one task will be assigned to each station. In order to optimize the desired performance measure, we must take each of these different cases into consideration. Cases to be evaluated for this example are presented in Table 4.1.

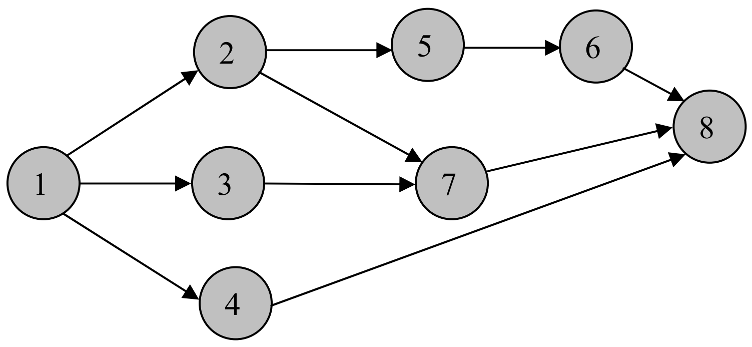


Figure 4.9 Precedence diagram

Table 4.1 Cases to be evaluated for being able to assign the tasks in feasible task sequence in the example to 3 stations

C. No	Combinations	C. No	Combinations
1	1 | 8 | 2 3 4 6 5 7	12	1 8 2 | 3 | 4 6 5 7
2	1 | 8 2 | 3 4 6 5 7	13	1 8 2 | 3 4 | 6 5 7
3	1 | 8 2 3 | 4 6 5 7	14	1 8 2 | 3 4 6 | 5 7
4	1 | 8 2 3 4 | 6 5 7	15	1 8 2 | 3 4 6 5 | 7
5	1 | 8 2 3 4 6 | 5 7	16	1 8 2 3 | 4 | 6 5 7
6	1 | 8 2 3 4 6 5 | 7	17	1 8 2 3 | 4 6 | 5 7
7	1 8 | 2 | 3 4 6 5 7	18	1 8 2 3 | 4 6 5 | 7
8	1 8 | 2 3 | 4 6 5 7	19	1 8 2 3 4 | 6 | 5 7
9	1 8 | 2 3 4 | 6 5 7	20	1 8 2 3 4 | 6 5 | 7
10	1 8 | 2 3 4 6 | 5 7	21	1 8 2 3 4 6 | 5 | 7
11	1 8 | 2 3 4 6 5 | 7		

Suppose that 2 models processed in U-line are with MPS={2,1} and the cycle time is 12. Task times of the models are as shown in Table 4.2. If we assess the cases 10 and 16 in Table 4.1 by taking AAB as model sequence constant;

Table 4.2 Task times for two models

Task	Model A	Model B
1	5	3
2	1	5
3	6	0
4	0	1
5	2	6
6	0	4
7	4	2
8	3	5

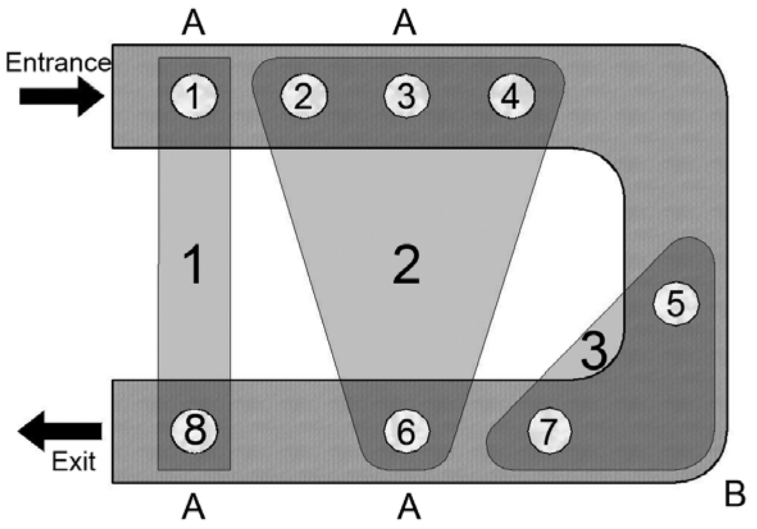


Figure 4.10 U-line structure of the case 10

Table 4.3 Workloads of workstations for the case 10

Workstation	1			2			3		
Tasks assigned {front}, {back}	{1}, {8}			{2, 3, 4}, {6}			{5, 7}, {ϕ}		
Models processed [front, back]	A, A	B, A	A, B	A, A	A, B	B, A	A	A	B
Total workload	8	6	10	7	11	6	6	6	8

U-line configuration structure of the case 10 and workloads of the workstations are shown in Figure 4.10 and Table 4.3 respectively. As it can be seen, the workload of any station for the case 10 did not exceed the cycle time.

If we consider the case 16, as shown in Table 4.4, 1st workstation exceeded the cycle time at all cycles (15,15,13). U-line configuration structure of the case 16 is shown in Figure 4.11.

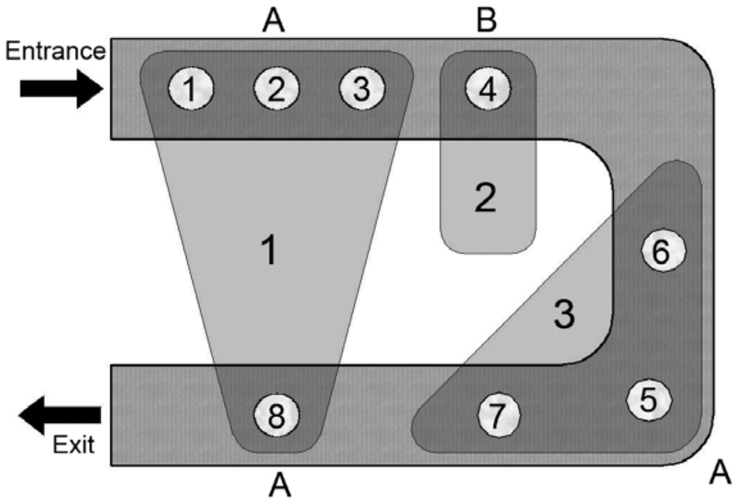


Figure 4.11 U-line structure of the case 16

Table 4.4 Workloads of workstations for the case 16

Workstation	1	2	3
Tasks assigned {front}, {back}	{1, 2, 3}, {8}	{4}, $\{\phi\}$	{6, 5, 7}, $\{\phi\}$
Models processed [front, back]	A, A A, A B, B	B A A	A B A
Total workload	15 15 13	1 0 0	6 12 6

If noticed, the number of the zones processed for the case 10 (zones where the tasks are processed in the line) is 5, and 4 for the case 16. We will call these zones as model point. The appearance of the models processed at 1st cycle from the beginning point of the line to the endpoint at model points will be {A,A,B,A,A} for the case 10, and {A,B,A,A} for the case 16. We will call these appearances formed at cycle k as cycle sequence (CS_r).

The number of all cycles to be formed until the completion of the production of all products (R) is equal to the length of the model sequence (MS). Because the model sequence is 3 in the given example, so the number of cycles is 3.

As mentioned before, the best line configuration can be obtained by considering all cases under a fixed model sequence and the number of fixed stations. However, there may also be cases where any feasible solution can not be found for desired number of stations. For example, if the number of stations is taken as 1 for the above mentioned example, there is only one case to be considered, and all tasks are assigned to one station, but no feasible solution can be obtained.

While evaluating a task sequence by a given minimum part set (MPS), if we consider all different model sequences to be formed rather than under a fixed model sequence assumption, the permutation $\left(\frac{(\sum_{m=1}^{M} d_m)!}{(d_1!d_2!...d_M!)}\right)$ is formed. In this given example, there are 3 permutations to be formed (3!/2!1!) because MPS={2,1}. If we consider all model sequences that may be formed in order to optimize a certain performance,

$$(C(N-1,K-1)) \times \left(\frac{(\sum_{m=1}^{M} d_m)!}{(d_1!d_2!...d_M!)}\right)$$

different cases are formed that we must consider for evaluating a tasks sequence under the number of fixed stations.

In case where model sequence is not accepted as fixed at MMULs, a procedure that calculate a lower bound has not be developed yet. Therefore, the number of stations must be increased one by one beginning from the station number one until the feasible solution is found in order to find minimum station number (type 1) and all obtained cases must be assessed (including number of stations giving the feasible solution). In this case, the number of all cases that must be evaluated for N pieces of tasks until finding a feasible solution (including the station giving the feasible solution) in a given task sequence (under variable model sequence assumption) is given at following equation:

$$(C(N-1,0)+C(N-1,1)+...+C(N-1,K-1)) \times \left(\frac{(\sum_{m=1}^{M} d_m)!}{(d_1!d_2!...d_M!)}\right)$$

As it is seen, evaluation of all cases to be formed for only one task sequence is a difficult and very time consuming work, and shows NP-hard structure.

As indicated by Jin et al. (2002), one essential difficulty in employing evolutionary algorithms in some applications is the huge time consumption due to the high complexity of performance analyses for fitness evaluation and the large number of evaluations needed in the evolutionary optimization. Various efficiency-enhancement techniques have been developed in order to facilitate the solution of large-scale complex problems, and further enhance the performance of GAs. One such class of efficiency-enhancement technique is called evaluation relaxation. In evaluation relaxation, an accurate, but computationally expensive fitness evaluation is replaced with a less accurate, but computationally inexpensive fitness estimate (Goldberg, 2002).

In this thesis, simulated annealing based fitness evaluation approach is developed in order to facilitate fitness function calculation in PGA and to perform it in an effective manner. Simulated annealing algorithm is chosen, among other meta-heuristics, mainly because of its flexibility to respond to modifications in the objective functions or in the problem constraints. When these changes occur, the basic simulated annealing program remains unchanged. In addition, the ALBP solutions and neighborhood structures can be easily defined by using simulated annealing.

Kara et al. (2007a) developed a neighborhood generation mechanism by changing the model sequence or the line balancing. In this study, we modified this neighborhood generation mechanism. Our modified neighborhood generation mechanism can make station assignments by randomly dividing the task in a manner at least one task will be assigned to each station under a certain number of stations or change the model sequence. We inserted this modified neighborhood generation mechanism into our simulated annealing algorithm based fitness evaluation approach.

The proposed SA-based approach is described by the following steps:

Step0. *Specify the problem parameters (SA algorithm parameters (T_0, T_{cry}, q, IT), a task sequence (TS) from GA and a feasible model sequence (MS_0)); Set E_c =a very high value, E_{best} =a very high value, $con = 0$, K =a user-defined station number. Then, go to Step 1.*

Step1. *Generate an initial line balance (LB_0) by using the line balancing procedure (see Chapter 4.3.5.1) with the current K; Set $T_C = T_0$, $n = 1$, $LB_C = LB_0$ and $MS_C = MS_0$. Then, go to Step 2.*

Step2. *Generate a neighbor (a new LB_n or MS_n can be generated changing LB_C or MS_C) by using the neighborhood generating procedure (see Chapter 4.3.5.2) and go to Step 3.*

Step3. *Check the workstations time feasibility (The each station time at each cycle is checked with generated combination of LB_n , MS_n and the given cycle time (C)) by using the checking feasibility of workstation time procedure (see Chapter 4.3.5.3) and then, assign $f = 1$ to feasible case and $f = 0$ to infeasible case. Then, go to Step 4.*

Step4. *If f equal to 1, then go to Step 5. Otherwise; go to Step 9.*

Step5. *Set; $con = con + 1$. Calculate; the cost of this neighbor solution (E_n) and then calculate the difference between the cost of neighbor solution (E_n) and the cost of current solution (E_c) by the following equation: $\Delta = E_n - E_C$ and then, go to Step 6.*

Step6. *If $\Delta \leq 0$, then accept the sequence (MS_n) as a new sequence (MS_C), set $E_c = E_n$, $LB_C = LB_n$, $MS_C = MS_n$ and then, go to Step 7. Otherwise; go to Step 8.*

Step7. *If $E_c < E_{best}$;set $E_{best} = E_C$, $LB_{best} = LB_C$, $MS_{best} = MS_C$. Otherwise; E_{best}, LB_{best} and MS_{best} are not change. Go to Step 9.*

Step8. *If* $\Delta > 0$*, then accept the neighbor solution as the current solution with the probability of* $\exp(-\Delta / T_c)$ *and set* $E_{best} = E_c, LB_{best} = LB_c$, $MS_{best} = MS_c$*. Otherwise;* E_c*,* LB_c*,* MS_c *are not change. Then, go to Step 9.*

Step9. *If* $n = IT$*, then g o to Step 10. Otherwise; set* $n = n + 1$ *and go to Step 2.*

Step10. *Set;* $T_c = T_c \times q$*,* $n = 1$ *and then go to Step 11.*

Step11. *If* $T_c \geq T_{cry}$*, then go to Step 2. Otherwise; go to Step 12.*

Step12. *If* $con \geq 1$*, then Stop. Otherwise; set* $K = K + 1$ *and go to Step 1.*

The flowchart of the proposed SA-based fitness evaluation approach is given in Figure 4.12.

Developed simulated annealing algorithm based fitness evaluation approach uses user defined station number, a feasible model sequence given by user (MS_0-initial model sequence), and SA algorithm parameters (T_{cry}, T_0, q, IT) suitable for the problem. At the beginning, a high value is assigned to the current cost function (E_c *=a very high value)*, a high value for the best cost function (E_{best} *=a very high value)*, and the value zero ($con = 0$) to the variable *con* which is used for controlling whether a feasible solution is found under current station number of the algorithm. Initial solution for line balancing (LB_0) is generated by using line balancing part of the modified neighborhood generation mechanism (with the current number of stations). This line balancing and feasible model sequence given by user form current line configuration ($LB_c = LB_0$ *and* $MS_c = MS_0$*)*. Current temperature is equalized to initial temperature ($T_C = T_0$*)*, and iteration number at length of each temperature level is equalized to 1 ($n = 1$). Feasible solution is sought after the assignments of the said variable and parameter by generating new neighborhoods from the current line configuration (LB_c *and* MS_c*)* for the addressed task sequence (TS). New neighborhoods generated for line balancing are generated depending on the probability p_1. New neighborhoods formed for line balancing (LB_n*)* are generated randomly completely independently from the current line balancing,

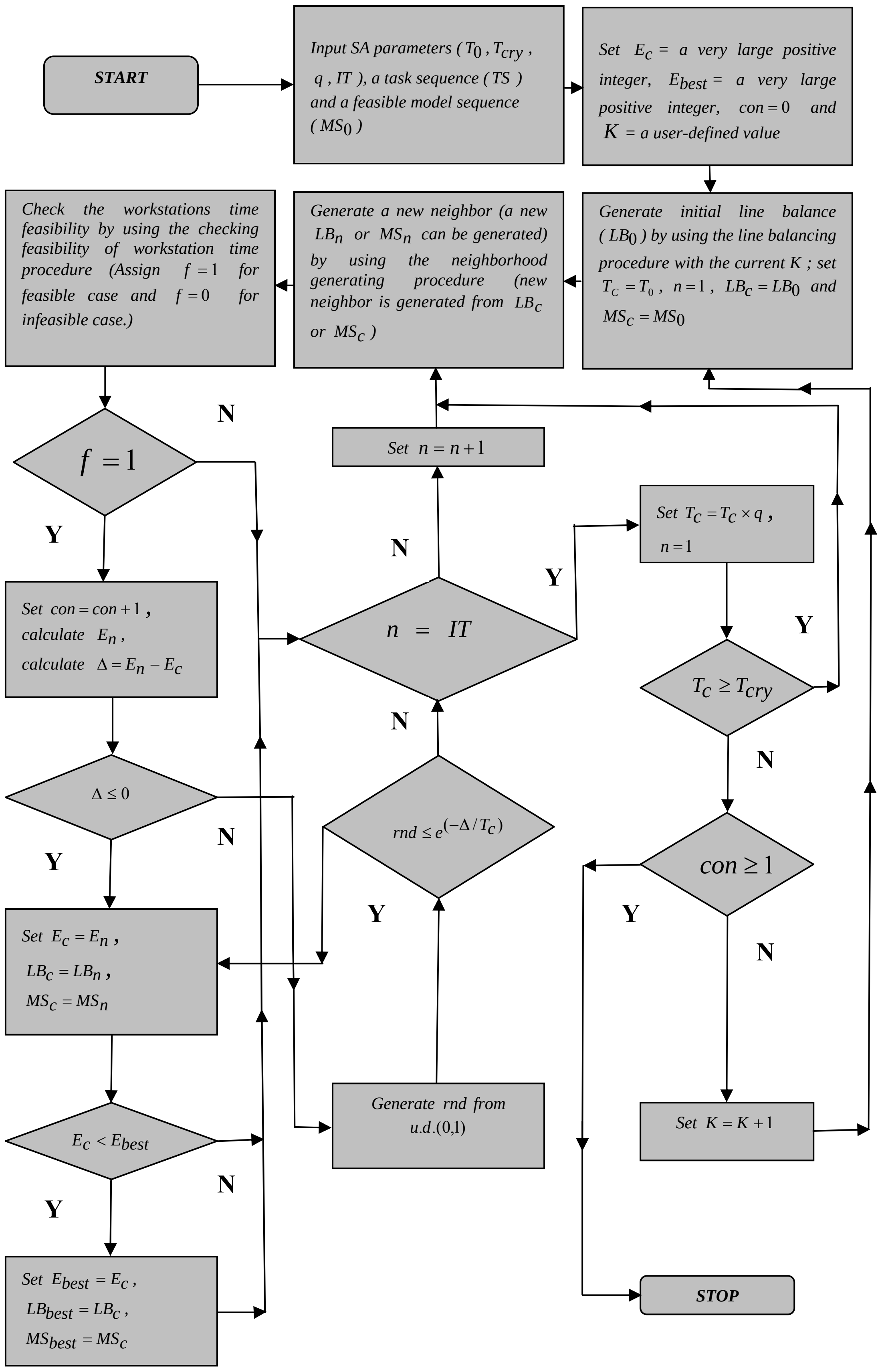

START
Input SA parameters (T_0 , T_{cry} , q , IT), a task sequence (TS) and a feasible model sequence (MS_0)
Set E_c = a very large positive integer, E_{best} = a very large positive integer, $con = 0$ and K = a user-defined value
Generate initial line balance (LB_0) by using the line balancing procedure with the current K ; set $T_C = T_0$, $n = 1$, $LB_c = LB_0$ and $MS_c = MS_0$
Generate a new neighbor (a new LB_n or MS_n can be generated) by using the neighborhood generating procedure (new neighbor is generated from LB_c or MS_c)
Check the workstations time feasibility by using the checking feasibility of workstation time procedure (Assign $f = 1$ for feasible case and $f = 0$ for infeasible case.)
$f = 1$
N
Y
Set $n = n + 1$
Set $T_c = T_c \times q$, $n = 1$
N
Y
$n = IT$
Set $con = con + 1$, calculate E_n , calculate $\Delta = E_n - E_c$
Y
$T_c \geq T_{cry}$
N
$\Delta \leq 0$
N
Y
$rnd \leq e^{(-\Delta / T_c)}$
N
$con \geq 1$
Y
Y
N
Set $E_c = E_n$, $LB_c = LB_n$, $MS_c = MS_n$
Generate rnd from $u.d.(0,1)$
Set $K = K + 1$
$E_c < E_{best}$
N
Y
Set $E_{best} = E_c$, $LB_{best} = LB_c$, $MS_{best} = MS_c$
STOP

Figure 4.12 Flow-chart of the proposed SA-based fitness evaluation approach

depending on the current number of stations, and in a manner at least one task will be assigned to each station. New neighborhoods for model sequencing are generated depending on the probability $(1 - p_1)$. New neighborhoods for model sequencing (MS_n) are generated by swapping (depending on the probability p_2) or inserting (depending on the probability $(1 - p_2)$) by depending on the current model sequence (MS_c). Iteration as much as the length of each temperature level (IT) is carried out at each energy level $(T_c = T_c \times q)$ until initial temperature (T_0) reaches at crystallization temperature (T_{cry}). As a result, if a feasible solution is found in the system ($f = 1, con \geq 1$), the algorithm continues searching solution until termination conditions of the algorithm are ensured. Cost functions (E_n) of obtained new feasible solutions are calculated by our proposed fitness function. Depending either on the fact that the cost function of the new solution is lesser or the probability of accepting bad results (metropolis criterion); E_c, LB_c *and* MS_c are updated. After the termination of the algorithm, the solution giving the minimum cost function (E_{best}) is the solution giving the best line balancing (LB_{best}) and model sequencing (MS_{best}) configuration. If feasible solution ($f = 0$) cannot be found, the number station is increased by one $(K = K + 1)$ and these processes continue until the feasible solution is found by updating simulated annealing algorithm parameters with initial parameters. Therefore, it may start and stop more than once.

4.3.5.1 Initial Solution $(LB_0$ - $MS_0)$

The initial solution for the proposed SA based algorithm contains solutions for two problems: MMUL/LB and MMUL/MS. The combination of these solutions provides an initial solution to MMUL/BS problem. So, the proposed SA based algorithm is adopted to generate initial solution (LB_0), randomly, for MMUL/LB by using a procedure which will be discussed in Chapter 4.3.5.2.1, and a user defined feasible model sequence (MS_0) is used as an initial solution for MMUL/BS.

4.3.5.2 Neighboring Solutions (LB_n - MS_n)

We modified the neighborhood generation logic developed by Kara et al. (2007a) in a manner corresponding to our problem characteristic. A neighbor solution can be either a new line balancing or a new model sequence. New line balancing solution is randomly generated depending on current number of stations (*K*) and consists of three successive phases. A new model sequence is generated by changing the positions of models in the model sequence.

The modified neighborhood generation logic enables us to consider the line balancing and model sequencing problems of mixed-model U-lines simultaneously as shown in Figure 4.13.

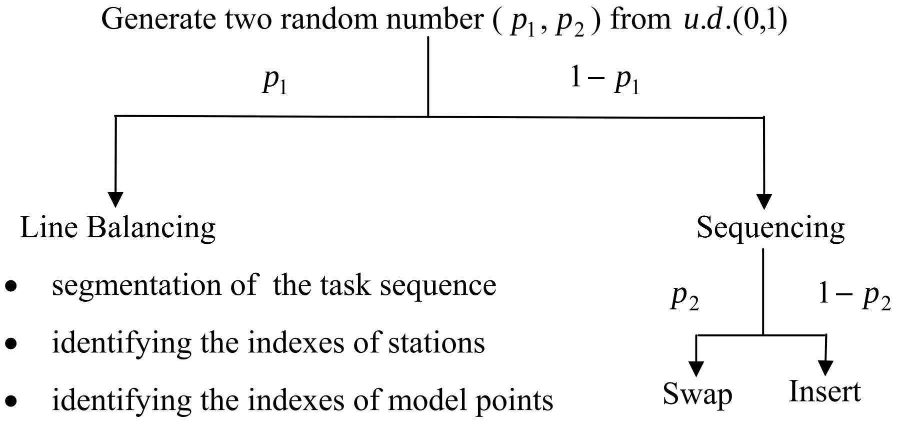

Generate two random number (p_1, p_2) from $u.d.(0,1)$
p_1
$1-p_1$
Line Balancing
• segmentation of the task sequence
• identifying the indexes of stations
• identifying the indexes of model points
Sequencing
p_2
$1-p_2$
Swap
Insert

Figure 4.13 Neighborhood generation mechanism of the proposed SA

Initially, two random numbers (p_1, p_2) are specified to determine the type of the new neighbor solution.

4.3.5.2.1 Line Balancing (LB_n). Line balancing neighborhood is generated depending on the probability p_1 and consists of three successive phases.

- First of all, task sequence is randomly divided into segments as much as the number of stations (*K*) in a manner at least one task will be assigned to each station, and each length of segment is placed in the list that we will call as assignment list (*AL*) in sequence.

Division of the task sequence (*TS*) to segments and formation of assignment list (AL) is shown in following procedure step-by-step.

Input: *a feasible task sequence, number of tasks (N), number of stations (K)*
Output: *assignment list (AL)*

```
Step0. Set; $u = [N-(K-1)]$, $z = 1$, $I = 1$ and then, go to Step 1.
Step1. If $I$ not equal to $K$, then go to Step 2. Otherwise; go to Step 3.
Step2. Generate a random number between $z$ and u. Set;
       $AL(I) = [rnd(z,u)-(z-1)]$, $z = z + AL(I)$, $u = u+1$, $I = I+1$
       and then, go to Step 1.
Step3. Set; $AL(I) = [u-(z-1)]$ and then, Stop.
```

For example, let a feasible task sequence obtained by using priority based decoding procedure from precedence diagram consisting of 10 tasks in Figure 4.6 be $TS = \{1,4,10,3,2,9,5,6,7,8\}$, and suppose that the tasks in this task sequence will be assigned to 4 stations. Because the task sequence length is 10 and at least one task must be assigned to each workstation, a number between 1 and 7 is randomly generated, and suppose that generated random number is 3. This shows that the tasks in this segment will be at the same station and placed in the assignment list, $AL = \{3\}$. Because of the fact that remaining number of stations is 3 for carrying out segmentation, a random number is generated between 4 and 8 in order to form next segment. Suppose that generated random number is 6. Due to the fact that the stand point of its previous segment is 3, the difference between generated number and the length of previous segment is assigned for this segment (6-3). This segment is also placed in this assignment list, $AL = \{3,3\}$. Now, because the remaining number of stations for making segmentation is 2, a random number is generated between 7 and 9. Suppose that generated random number is 8. Again this segment is placed in this assignment list according to stand point of the previous segment in task sequence,

$AL = \{3,3,2\}$. Finally, because of the fact that remaining number of stations is 1, remaining tasks (two tasks) in task sequence are placed for the last segment in the assignment list $AL = \{3,3,2,2\}$. The output of this procedure is shown in Figure 4.14.

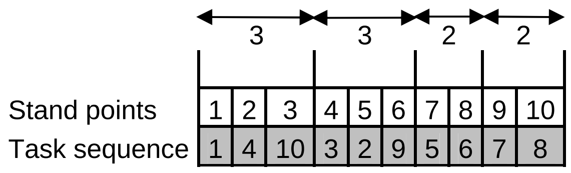


Figure 4.14 The output of the example for dividing into segments

- And then, precedence relations of the tasks forming each segment in the assignment list are controlled, and which station index (SI) will be taken is determined. Station indexes start from 1 in a manner not breaking process flow of the tasks and are as much as current station number. These station indexes are placed respectively in the list to be called as station index sequence (AIS).

Determination of station index numbers of the tasks divided into segments (SI) and placement of these indexes to station index sequence (AIS) are shown step-by-step in the following procedure.

Input: *assignment list ($AL(.)$) from previous phase, task sequence ($TS(.)$), number of tasks (N), number of stations (K), the set of task nodes, and an empty precedence control list (CPR)*

Output: *station index sequence (AIS)*

```
Step0. Set $SI = 1$ (index) and go to Step 1.
Step1. Set $z = 1$, $I = 1$ and go to Step 2.
Step2. Set $u = \sum_{i=1}^{I} AL(i)$ and go to Step 3.
Step3. for ( $j = z$ to $u$ ; $j = j + 1$ ){
            $control = 0;$
            if  (CPR contains $IP_{TS(j)}$){
              $control = 1$;
              for ($i = z$ to $u$ ; $i = i + 1$){
                  Set $AIS(i) = SI$ ;
                  Set $CPR(i) = TS(i)$ ;
              }end for.
              Set $SI = SI + 1$ ;
              if (($SI - 1$) equal to $K$ ) Stop.
            } end if.
            if (control equal to 1){
              Go to Step 4 (break).
            } end if.
        } end for. Then, go to Step 4
Step4. if $u$ is not equal to $N$ , set $z = \sum_{i=1}^{I} AL(i) + 1$, $I = I + 1$, and then go to Step
        2. Otherwise; go to Step 1.
```

For example; let's determine which station indexes will be possessed by tasks in task sequence ($TS = \{1,4,10,3,2,9,5,6,7,8\}$) corresponding to the segments in assignment list ($AL = \{3,3,2,2\}$) obtained from abovementioned example.

Tasks corresponding to the first segment are 1, 4 and 10. As seen from the precedence diagram in Figure 4.6, immediate predecessors of these tasks are $IP_1 = \{\}$, $IP_4 = \{1\}$ and $IP_{10} = \{9\}$, respectively. Because there is no task to be performed before Task 1, all tasks in this segment take the index numbered 1($SI = 1$) and they are placed in station index sequence ($AIS = \{1,1,1\}$). Tasks numbered 3, 2 and 9 take place in the second segment. Immediate predecessors of these tasks are $IP_3 = \{\}$, $IP_2 = \{\}$ and $IP_9 = \{3,7\}$, respectively. Because there is no task to be performed before Task 2 and 3, all tasks of that segment take the subsequent station index and are placed in the station index sequence ($AIS = \{1,1,1,2,2,2\}$). Tasks numbered 5 and 6 whose immediate predecessors are $IP_5 = \{4\}$, $IP_6 = \{2,5\}$ take place in the third segment. Because the tasks that must be performed immediately before these tasks have previously taken their station indexes, these tasks numbered 5 and 6 in queue take the station index numbered 3 ($SI = 3$**)** and are placed in station index sequence list ($AIS = \{1,1,1,2,2,2,3,3\}$). Finally, immediate predecessors of the tasks numbered 7 and 8 taking place in the fourth segment are $IP_7 = \{6\}$, $IP_8 = \{7\}$, respectively. Because task numbered 6 has previously taken station index, these tasks (7 and 8) takes the station index numbered 4 ($SI = 4$) and are placed in station index sequence list ($AIS = \{1,1,1,2,2,2,3,3,4,4\}$). The output of this procedure is shown in Figure 4.15.

TS	1	4	10	3	2	9	5	6	7	8
SI	**1**	**1**	**1**	**2**	**2**	**2**	**3**	**3**	**4**	**4**

Figure 4.15 The output of the example of station index determination

- Finally, model point indexes (MP) are detected by considering precedence relationship among tasks in order to determine which model time will be taken by tasks whose station indexes are set, and these model point indexes are placed respectively in the list that we will call model point index sequence (MIS*)*. As already mentioned before, once all tasks take their model point indexes, maximum model point index becomes equal to the length of cycle sequence.

Determination of which model point index will be taken by tasks whose station indexes (SI) are set, and placement of these determined indexes to the list of model point index sequence (MIS) are shown step-by-step in following procedure.

Input: *number of tasks (N), number of workstations (K), assignment list ($AL(.)$) from first phase, station index sequence ($AIS(.)$) from second phase, task sequence ($TS(.)$) (the same task sequence with previous phase), and an empty precedence control list (CPR)*

```
**Output:** Model index sequence ($MIS$)
**Step0.** Set $MP = 1$ (index), $z = 1$, $I = 1$ and then go to Step 1.
**Step1.** Set $u = \sum_{i=1}^{I} AL(i)$. Then, go to Step 2.
**Step2.** For ($i = z$ to $u$ ; $i = i + 1$){
            Set $CPR(i) = TS(i)$ ;
      }end for. Then, go to Step 3.
**Step3.** Set $control = 0$ ;
      for ($i = z$ to $u$ ; $i = i + 1$){
         if ($CPR$ contains $IP_{TS(i)}$){
            Set $control = 1$;
            Set $MIS(i) = MP$ ;
         } end if.
      }end for. Then, go to Step 4.
**Step4.** if ($control$ equal to 1){
         Set $MP = MP + 1$
      }end if.
      if all location of $MIP$ have indexes of model points ($MP$) Stop;
      Otherwise go to Step 5.
**Step5.** Set $z = \sum_{i=1}^{I} AL(i) + 1$, $I = I + 1$. if $z$ bigger than $N$ and then go to Step 6.
      Otherwise; go to Step 1.
**Step6.** Set $I = K - 1$ and then, go to Step 7.
**Step7.** Set $u = \sum_{i=1}^{I} AL(i)$. If $I$ not equal to 1 then, set $z = \sum_{i=1}^{I-1} AL(i) + 1$.
      Otherwise; set $z = 1$ and go to Step 8.
**Step8.** Set $control = 0$ ;
         for ($i = u$ to $z$ ; $i = i - 1$){
            if $MIS(i)$ has not any index of model point ($MP$){
               Set $MIS(i) = MP$
               Set $control = 1$;
            }end if.
        }end for. Then, go to Step 9.
Step9. If control equal to 1 then, set $MP = MP + 1$ and go to Step 10.
    Otherwise; go to Step 10.
Step10. Set $I = I - 1$. If $I$ is equal to 0 Stop. Otherwise; go to Step 7.
```

Let's determine which model point index (MP) will be possessed by tasks whose station indexes are determined ($AIS = \{1,1,1,2,2,2,3,3,4,4\}$) and see how these are transferred to the model point index sequence by continuing the example explained in previous procedure.

Tasks assigned to the station having station index numbered 1 are 1, 4 and 10; and immediate predecessors of these tasks are $IP_1 = \{\}, IP_4 = \{1\}$ and $IP_{10} = \{9\}$, respectively. Because there is no task in the precedence of the task numbered 1, the model point index numbered 1 ($MP = 1$) is assigned to the task numbered 1 (to the said task), $MIS = \{1\}$. Because the task numbered 4 is at the same station with the task numbered 1, and the task numbered 1 takes place in immediate predecessor of the task numbered 4, the task numbered 4 also takes the model point index numbered 1 ($MP = 1$), $MIS = \{1,1\}$. Due to the fact that a model point index is not assigned yet to the task numbered 9 in immediate predecessor of the task numbered 10, a model point index cannot be assigned now to the task numbered 10, $MIS = \{1,1,\phi\}$. It is understood from this situation that this station is a crossover station. After having evaluated model point indexes of all of the remaining tasks in task sequence order, the tasks in task sequence are re-evaluated from back to front for the tasks whose model point indexes are not assigned. The model point index is increased by one, $MP = 2$, and we pass to the segment having next station index, $SI = 2$. Tasks assigned to the station having the station index numbered 2 are 3, 2 and 9; and immediate predecessors of these tasks are $IP_3 = \{\}, IP_2 = \{\}$, and $IP_9 = \{3,7\}$ respectively. Because there is no task in the precedence of the task numbered 3 and 2, the model point index of the tasks numbered 3 and 2 is assigned as 2 ($MP = 2$), $MIS = \{1,1,\phi,2,2\}$. Because the tasks in immediate predecessor of the task numbered 9 are task 3 and 7, and task 3 is already at the same station, it does not hinder the assignment of the model point index for the task numbered 9, however, a model point index cannot be assigned now to task numbered 9 because a model point index is not assigned previously to the task numbered 7, $MIS = \{1,1,\phi,2,2,\phi\}$. This station is also a crossover station. Model point index is increased by one, $MP = 3$, and we pass to the station having the next station index, $SI = 3$. Tasks assigned to the station having the station index numbered 3 are 5 and 6, and immediate predecessors of these tasks are $IP_5 = \{4\}$ and $IP_6 = \{2,5\}$ respectively. Because the task numbered 4 in immediate predecessor of the task 5 has previously taken a model point index, $MP = 3$ is assigned to the task 5, $MIS = \{1,1,\phi,2,2,\phi,3\}$. Because the task 5 in immediate predecessor of the task 6 is at the same station, and a model point index is also assigned to task 2 previously, $MP = 3$ is assigned to the task 6, $MIS = \{1,1,\phi,2,2,\phi,3,3\}$. Model point index is increased by one, $MP = 4$, and we pass to the station having the next station index, $SI = 4$. Tasks assigned to the station having the station index numbered 4 are 7 and 8, and immediate predecessors of these tasks are $IP_7 = \{6\}$ and $IP_8 = \{7\}$ respectively. Because a model point index is previously assigned to the task 6 in immediate precedence of the task 7 and they are at the same station with the task 7 in immediate precedence of the task 8, $MP = 4$ is assigned to these two tasks, $MIS = \{1,1,\phi,2,2,\phi,3,3,4,4\}$. Model point index is increased by one, $MP = 5$. All tasks in the task sequence were evaluated; however, a model point index could not be assigned yet to tasks 10 and 9. Therefore, the tasks whose station indexes are determined in the task sequence are re-evaluated from the station having the last station index until the station having the station index numbered 1. All of the tasks assigned to the station having the station index numbered 4 and 3 have a model point index. The model point index of the tasks numbered 9 assigned to the station having the station index numbered 2 is not assigned yet. The task 3 is assigned to the same station and a model point index is also assigned to the task 7 previously in immediate precedence of the task 9, so $MP = 5$ is assigned to the task 9, $MIS = \{1,1,\phi,2,2,5,3,3,4,4\}$. Model point index is increased by one, $MP = 6$, and we pass to the station having the station index numbered 1. At this station, because a model point index is just assigned to the task

9 in immediate precedence of the task 10 whose model point index is not assigned, $MP = 6$ is assigned to the task 10, $MIS = \{1,1,6,2,2,5,3,3,4,4\}$. Thus, all tasks have taken their model point indexes. At the end of this procedure, which tasks will be processed at which zone of the assembly line will have been determined. The output of the end of the application of this procedure to the example is as in Figure 4.16.

TS	1	4	10	3	2	9	5	6	7	8
AIS	**1**	**1**	**1**	**2**	**2**	**2**	**3**	**3**	**4**	**4**
MIS	**1**	**1**	**6**	**2**	**2**	**5**	**3**	**3**	**4**	**4**

Figure 4.16 The output of the example of the determination of model point indexes

This line balancing solution (see output of the example) contains six model point indexes (maximum MP of MIS) and four workstations (maximum SI of AIS). The production process must follow these model point indexes of locations from 1 to the last index so as to provide precedence constrains. These indexes for this line balance configuration are shown in Figure 4.17.

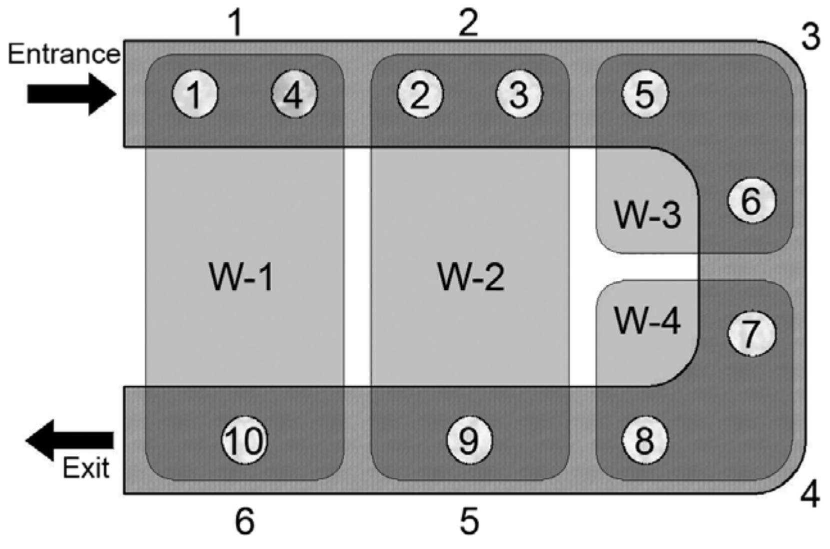


Figure 4.17 Indexes of model points

As mentioned before, CS_r represents the cycle sequence at the cycle r. The length of each cycle sequence is equal to maximum MP of MIS. For example, if the model sequence is selected as AABBC, the cycle sequences (CS_r) which are proper according to these model points can be seen in Table 4.5. Cycle sequences will be used for controlling whether line configuration solutions are feasible or not.

Table 4.5 Cycle sequences

CS_r \ MP	**1**	**2**	**3**	**4**	**5**	**6**
CS_1	A	C	B	B	A	A
CS_2	A	A	C	B	B	A
CS_3	B	A	A	C	B	B
CS_4	B	B	A	A	C	B
CS_5	C	B	B	A	A	C

4.3.5.2.2 Sequencing (MS_n). A new model sequence is generated using swapping or inserting.

- *Swap:* A new model sequence generated by swapping two randomly selected models in the model sequence with the probability of $(1-p_1)$ $\times$ p_2. Note that these models should be different.
- *Insert:* A new model sequence generated by inserting a randomly selected model before another randomly selected model with the probability of $(1-p_1)$ $\times (1-p_2)$.

4.3.5.3 Checking Feasibility of Workstation Times

Every station time in each cycle will be checked with regard to following procedure and the proposed algorithm keeps running by taking into consideration if the solution is feasible (f=1) or not (f=0).

```
$f = 1$; (feasible)
for ($k = 1$ to $K$ (for all workstations); $k = k + 1$){
   for ($r = 1$ to $R$ (for all cycles); $r = r + 1$){
     $W_{kr}$ =0;(initially, workload of the workstation k at the cycle r is equal to zero)
        for ($i = 1$ to $N$ (for all tasks); $i = i + 1$){
           if ($TS(i) \in (XF_k + XB_k)$){
              $MP = MIS(i)$;
```

```
                $W_{kr} = W_{kr} + t_{TS(i)CS_r(MP)}$;
                  if ($W_{kr} > C$){(workload of workstation k at the cycle r can not
                exceed the cycle time)
                      $f = 0$; (infeasible)
                      Stop.
                  }end if.
              }end if.
          }end for.
      }end for.
  }end for.
```

Output of this procedure $f = 1$ (feasible) or $f = 0$ (unfeasible).

4.3.6 Identifying Efficient Control Parameters

In this section, a comprehensive experiment is conducted to evaluate the performance of proposed algorithm. The performance of proposed algorithm may vary according to some problem factors. The parameters of the proposed algorithm can be classified into two categories. These are Genetic Algorithm's control parameters and proposed SA-based fitness evaluation approach's control parameters.

Each control parameter varied at two levels (low and high).

Genetic Algorithm's Control Parameters:

- The population size (PS): 60-200,
- The crossover rate (RC): 0.50-0.95,
- The mutation rate (RM): 0.005-0.20,
- Maximum number of generations ($GN_{\max}$): 50-200.

Simulated Annealing Algorithm's Control Parameters:

- The cooling rate (q): 0.70-0.95,
- The probability of new line balancing(p_1) : 0.55-0.90,
- The initial temperature (T_0): 20-100.

Some parameters of the proposed algorithms are fixed to following statements:

- The minimum part set (MPS) is fixed to: {3,2,1},
- The length of the each temperature level (IT) is fixed to: 2,
- The crystallization temperature (T_{cry}) is fixed to: 1, and
- The probability of new sequencing (p_2) is fixed to: 0.5.

To identify the efficient control parameters, we employed statistical design of experiments (DOE) approach (Montgomery and Runger, 2005). DOE is a well-regarded investigative method both for its effectiveness and its efficiency in evaluating the effect of multiple factors upon a process. Thomopoulos19 problem (Thomopoulos, 1970) was chosen as the example for identifying the effect of different control parameters.

We conducted the 2^{7-2} fractional factorial design to analyze how much proposed algorithm's parameters interrelated with each other. The 2^{7-2} fractional factorial experimental layout was used for carrying out the experiments (see Table 4.6).

At each parameter setting, we performed multiple runs, i.e., 5 runs to determine the variation in the results. As a result, a total of (32*5) 160 runs were carried out. The proposed algorithm was coded in Matlab 7.6.0 and run on a 3.00 GHz Pentium 4 computer. Minitab 14 statistical package was used for analyzing the data.

Table 4.6 The 2^{7-2} fractional factorial design

Experiment no	PS	RC	RM	q	p_1	T_0	$GN_{\max}$
1	(-) 60	(-) 0.50	(-) 0.005	(-) 0.70	(-) 0.55	(+) 100	(+) 200
2	(+) 200	(-) 0.50	(-) 0.005	(-) 0.70	(-) 0.55	(-) 20	(-) 50
3	(-) 60	(+) 0.95	(-) 0.005	(-) 0.70	(-) 0.55	(-) 20	(-) 50
4	(+) 200	(+) 0.95	(-) 0.005	(-) 0.70	(-) 0.55	(+) 100	(+) 200
5	(-) 60	(-) 0.50	(+) 0.20	(-) 0.70	(-) 0.55	(-) 20	(+) 200
6	(+) 200	(-) 0.50	(+) 0.20	(-) 0.70	(-) 0.55	(+) 100	(-) 50
7	(-) 60	(+) 0.95	(+) 0.20	(-) 0.70	(-) 0.55	(+) 100	(-) 50
8	(+) 200	(+) 0.95	(+) 0.20	(-) 0.70	(-) 0.55	(-) 20	(+) 200
9	(-) 60	(-) 0.50	(-) 0.005	(+) 0.95	(-) 0.55	(-) 20	(-) 50
10	(+) 200	(-) 0.50	(-) 0.005	(+) 0.95	(-) 0.55	(+) 100	(+) 200
11	(-) 60	(+) 0.95	(-) 0.005	(+) 0.95	(-) 0.55	(+) 100	(+) 200
12	(+) 200	(+) 0.95	(-) 0.005	(+) 0.95	(-) 0.55	(-) 20	(-) 50
13	(-) 60	(-) 0.50	(+) 0.20	(+) 0.95	(-) 0.55	(+) 100	(-) 50
14	(+) 200	(-) 0.50	(+) 0.20	(+) 0.95	(-) 0.55	(-) 20	(+) 200
15	(-) 60	(+) 0.95	(+) 0.20	(+) 0.95	(-) 0.55	(-) 20	(+) 200
16	(+) 200	(+) 0.95	(+) 0.20	(+) 0.95	(-) 0.55	(+) 100	(-) 50
17	(-) 60	(-) 0.50	(-) 0.005	(-) 0.70	(+) 0.90	(+) 100	(-) 50
18	(+) 200	(-) 0.50	(-) 0.005	(-) 0.70	(+) 0.90	(-) 20	(+) 200
19	(-) 60	(+) 0.95	(-) 0.005	(-) 0.70	(+) 0.90	(-) 20	(+) 200
20	(+) 200	(+) 0.95	(-) 0.005	(-) 0.70	(+) 0.90	(+) 100	(-) 50
21	(-) 60	(-) 0.50	(+) 0.20	(-) 0.70	(+) 0.90	(-) 20	(-) 50
22	(+) 200	(-) 0.50	(+) 0.20	(-) 0.70	(+) 0.90	(+) 100	(+) 200
23	(-) 60	(+) 0.95	(+) 0.20	(-) 0.70	(+) 0.90	(+) 100	(+) 200
24	(+) 200	(+) 0.95	(+) 0.20	(-) 0.70	(+) 0.90	(-) 20	(-) 50
25	(-) 60	(-) 0.50	(-) 0.005	(+) 0.95	(+) 0.90	(-) 20	(+) 200
26	(+) 200	(-) 0.50	(-) 0.005	(+) 0.95	(+) 0.90	(+) 100	(-) 50
27	(-) 60	(+) 0.95	(-) 0.005	(+) 0.95	(+) 0.90	(+) 100	(-) 50
28	(+) 200	(+) 0.95	(-) 0.005	(+) 0.95	(+) 0.90	(-) 20	(+) 200
29	(-) 60	(-) 0.50	(+) 0.20	(+) 0.95	(+) 0.90	(+) 100	(+) 200
30	(+) 200	(-) 0.50	(+) 0.20	(+) 0.95	(+) 0.90	(-) 20	(-) 50
31	(-) 60	(+) 0.95	(+) 0.20	(+) 0.95	(+) 0.90	(-) 20	(-) 50
32	(+) 200	(+) 0.95	(+) 0.20	(+) 0.95	(+) 0.90	(+) 100	(+) 200

Table 4.7 Analysis of variance for responses

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	7	7.7263	7.7263	1.10375	21.34	0.000
2-Way Interactions	18	2.1417	2.1417	0.11898	2.30	0.004
3-Way Interactions	6	0.1556	0.1556	0.02593	0.50	0.806
Residual Error	128	6.6192	6.6192	0.05171		
Pure Error	128	6.6192	6.6192	0.05171		
Total	159	16.6428				

In order to determine which control parameter effects are significant, a statistical analysis of variance (ANOVA) is conducted. Table 4.7 shows the results of ANOVA, which indicates that each of the main effects is significant and some interactions are also found to be significant for the proposed performance measure (Z). Table 4.8 shows estimated effects and coefficients for responses.

Table 4.8 Estimated effects and coefficients for responses

Term	**Effect**	**Coef**	**SE Coef**	**T**	**P**
Constant		3.3681	0.01798	187.35	0.000
PS	-0.1635	-0.0818	0.01798	-4.55	0.000
RC	-0.1027	-0.0513	0.01798	-2.86	0.005
RM	-0.1184	-0.0592	0.01798	-3.29	0.001
q	-0.1038	-0.0519	0.01798	-2.89	0.005
p1	-0.1036	-0.0518	0.01798	-2.88	0.005
T0	-0.2351	-0.1176	0.01798	-6.54	0.000
GN(max)	-0.2550	-0.1275	0.01798	-7.09	0.000
PS*RC	0.0626	0.0313	0.01798	1.74	0.084
PS*RM	-0.0176	-0.0088	0.01798	-0.49	0.626
PS*q	0.0066	0.0033	0.01798	0.18	0.856
PS*p1	0.0147	0.0074	0.01798	0.41	0.682
PS*T0	0.0642	0.0321	0.01798	1.79	0.077
PS*GN(max)	0.0530	0.0265	0.01798	1.47	0.143
RC*RM	-0.0117	-0.0058	0.01798	-0.32	0.746
RC*q	0.0113	0.0056	0.01798	0.31	0.754
RC*p1	0.0422	0.0211	0.01798	1.17	0.243
RC*T0	0.0036	0.0018	0.01798	0.10	0.919
RC*GN(max)	0.0197	0.0098	0.01798	0.55	0.585
RM*q	0.0110	0.0055	0.01798	0.31	0.759
RM*p1	0.1324	0.0662	0.01798	3.68	0.000
RM*T0	0.0850	0.0425	0.01798	2.37	0.020
RM*GN(max)	0.0574	0.0287	0.01798	1.60	0.113
q*p1	-0.0244	-0.0122	0.01798	-0.68	0.499
q*T0	0.0983	0.0492	0.01798	2.74	0.007
q*GN(max)	0.0350	0.0175	0.01798	0.97	0.333
PS*RM*p1	-0.0139	-0.0069	0.01798	-0.39	0.701
PS*RM*GN(max)	0.0152	0.0076	0.01798	0.42	0.673
RC*RM*p1	0.0091	0.0045	0.01798	0.25	0.801
RC*RM*GN(max)	0.0038	0.0019	0.01798	0.10	0.917
RM*q*p1	-0.0576	-0.0288	0.01798	-1.60	0.112
RM*q*GN(max)	-0.0072	-0.0036	0.01798	-0.20	0.843
S = 0.227404 R-Sq = 60.23% R-Sq(adj) = 50.60%					

Normal probability plot of the standardized effects and Pareto chart of the largest 30 effects are shown in Figure 4.18, Figure 4.19, respectively.

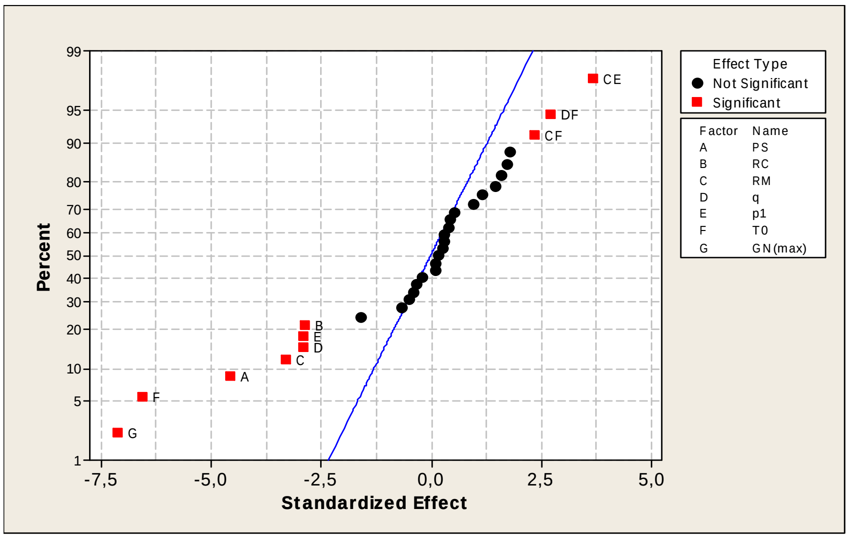


Figure 4.18 Normal probability plots of the standardized effects

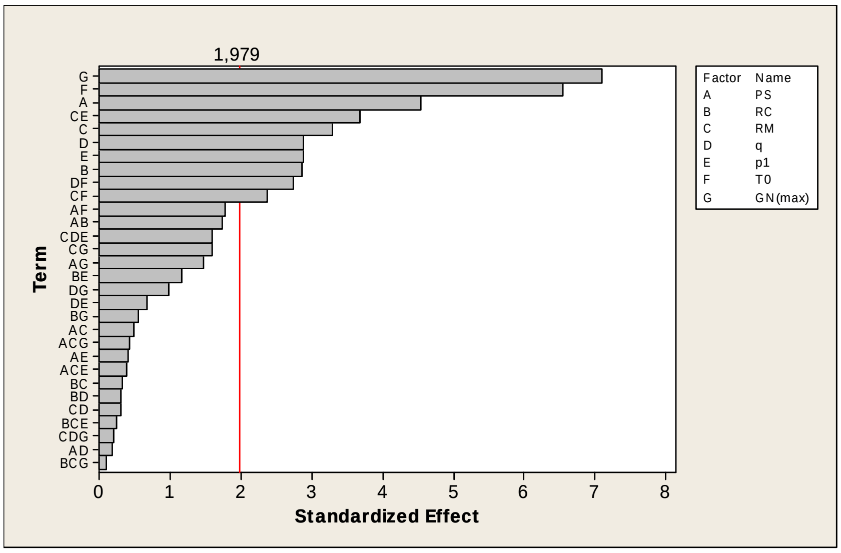


Figure 4.19 Pareto charts of the standardized effects

In addition to analysis above, the following linear equation is estimated from the results of the experiment:

$$\text{Response} = 3.37 - 0.0818 * PS - 0.0513 * RC - 0.0592 * RM - 0.0519 * q - 0.0518 * p1$$
$$-0.118 * T_0 - 0.128 * GN_{\max}$$

It is possible to estimate the response of the solution when the parameters of the algorithm changed. For example, if the cooling rate is changed from 0.70 to 0.99, we expect to have a better solution for all problems, and this situation is valid for the other handled control parameters.

4.3.7 Numerical Illustration

To describe the characteristics of proposed solution method for the problem of MMUL/BS, we used 10-task problem with three models (Kara et al., 2007a). Combined precedence diagram relationships among tasks and task completion times are given in Figure 4.20 and Table 4.9, respectively.

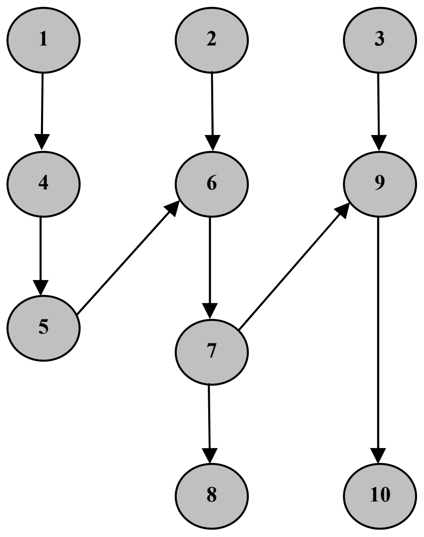


Figure 4.20 Combined precedence diagram

Table 4.9 Task completion times for the example problem

Task	Completion time		
	A	B	C
1	5	4	5
2	2	0	7
3	2	6	5
4	4	0	2
5	0	6	6
6	4	1	5
7	9	4	0
8	3	7	5
9	0	6	5
10	3	7	1

Suppose the demand rates of products A, B, and C is 40, 40, and 20 units in a planning period (P) of 1200 minutes (D={40, 40, 20}). The greatest divisor (cd) of vector D is 20. By dividing the elements of D by 20, MPS={2, 2, 1}. So, the example line is running at a cycle (C) of 12 minutes/units (1200/(40+40+20)). The initial model sequence (MS_0) of the example U-line is selected as AABBC and, thus the length of model sequence (R) is 5.

As mentioned, each task sequence (TS) is evaluated for calculating their fitness value by using the SA based fitness evaluation approach. SA based approach starts with the proper SA algorithm parameters, a task sequence (TS), a feasible model sequence (*MS*) and the user defined station number (*K*), at first. If the feasible solution is not found, the value of (*K*) is increased by one. Then, the parameters of SA algorithm are updated with initial parameters (T_{cry}, T_0, q, IT). This case continues whenever a feasible solution is found. After a feasible solution is found, the algorithm will continue the search until the SA termination conditions are met. The parameters that are used in the GA and SA based evaluation algorithm runs are listed in Table 4.10.

Table 4.10 Selected control parameters

Parameters	**Value**
Minimum part sets	{2,2,1}
Initial number of workstation (*K*)	1
Population size	100
Number of generations	100
Crossover probability	0.95
Mutation probability	0.2
Initial temperature	100
Cooling rate	0.95
Length of the each temperature level	2
Crystallization temperature	1
Probability of p1	0.75
Probability of p2	0.5

Table 4.11 Step-by-step illustration of the solution process

Stage 0: Evaluation of the initial population

Station (k)	**Task (i) $\{XF_k\}$, $\{XB_k\}$**	**Idle time (S_{kr}) $\{r = 1,2,...R\}$**
1	$\{1,3\},\{\phi\}$	$\{2,2,5,5,2\}$
2	$\{2\},\{10\}$	$\{4,5,5,7,7\}$
3	$\{4\},\{8\}$	$\{1,3,9,9,3\}$
4	$\{5,6\},\{\phi\}$	$\{8,8,1,5,5\}$
5	$\{7,9\},\{\phi\}$	$\{2,3,3,7,2\}$

$MS : CBBAA$; $Z = 5.1336$

Stage 1: Evaluation of the 1.generation

Station (k)	**Task (i) $\{XF_k\}$, $\{XB_k\}$**	**Idle time (S_{kr}) $\{r = 1,2,...R\}$**
1	$\{1,4\},\{10\}$	$\{1,0,1,0,4\}$
2	$\{2,3\},\{9\}$	$\{0,0,8,1,2\}$
3	$\{5,6\},\{\phi\}$	$\{8,1,5,8,5\}$
4	$\{7,8\},\{\phi\}$	$\{1,0,7,1,0\}$

$MS : BABAC$; $Z = 4.8265$

Stage 10: Evaluation of the 10.generation

Station (k)	**Task (i) $\{XF_k\}$, $\{XB_k\}$**	**Idle time (S_{kr}) $\{r = 1,2,...R\}$**
1	$\{1,4\},\{10\}$	$\{0,1,1,4,0\}$
2	$\{2,3\},\{9\}$	$\{2,2,1,6,0\}$
3	$\{5,6\},\{\phi\}$	$\{1,8,8,5,5\}$
4	$\{7,8\},\{\phi\}$	$\{1,7,0,0,1\}$

$MS : ABBCA$; $Z = 4.6196$

Stage 20: Evaluation of the 20.generation

Station (k)	**Task (i) $\{XF_k\}$, $\{XB_k\}$**	**Idle time (S_{kr}) $\{r = 1,2,...R\}$**
1	$\{2\},\{10\}$	$\{7,7,4,5,5\}$
2	$\{1,3,4\},\{\phi\}$	$\{2,1,1,0,2\}$
3	$\{5,6\},\{9\}$	$\{5,0,2,2,1\}$
4	$\{7,8\},\{\phi\}$	$\{7,1,1,0,0\}$

$MS : AACBB$; $Z = 4.4910$

Stage 60: Evaluation of the 60.generation

Station (k)	**Task (i) $\{XF_k\}$, $\{XB_k\}$**	**Idle time (S_{kr}) $\{r = 1,2,...R\}$**
1	$\{1,3,4\},\{\phi\}$	$\{0,1,2,1,2\}$
2	$\{5\},\{10\}$	$\{5,3,5,3,5\}$
3	$\{2,6\},\{9\}$	$\{6,5,0,0,6\}$
4	$\{7,8\},\{\phi\}$	$\{1,0,1,7,0\}$

$MS : CABAB$; $Z = 4.4587$

Table 4.11 (cont) Step-by-step illustration of the solution process

Stage 70: Evaluation of the 70.generation

Station (k)	***Task (i) $\{XF_k\}$, $\{XB_k\}$***	***Idle time (S_{kr}) $\{r=1,2,...R\}$***
1	$\{1\},\{8\}$	$\{5,0,5,0,2\}$
2	$\{4,5\},\{10\}$	$\{1,3,1,5,1\}$
3	$\{2,6\},\{9\}$	$\{0,0,6,6,5\}$
4	$\{3,7\},\{\phi\}$	$\{2,7,1,2,1\}$

$MS: ABABC$; $Z = 4.4271$

Stage 80: Evaluation of the 80.generation

Station (k)	***Task (i) $\{XF_k\}$, $\{XB_k\}$***	***Idle time (S_{kr}) $\{r=1,2,...R\}$***
1	$\{1,3\},\{\phi\}$	$\{5,2,2,5,2\}$
2	$\{4,5\},\{10\}$	$\{3,1,5,1,1\}$
3	$\{2,6\},\{9\}$	$\{0,6,6,5,0\}$
4	$\{7,8\},\{\phi\}$	$\{7,0,1,0,1\}$

$MS: ABCAB$; $Z = 4.4251$

Stage 100: Evaluation of the 100.generation

Station (k)	***Task (i) $\{XF_k\}$, $\{XB_k\}$***	***Idle time (S_{kr}) $\{r=1,2,...R\}$***
1	$\{1,3\},\{\phi\}$	$\{2,5,2,2,5\}$
2	$\{4,5\},\{10\}$	$\{1,3,1,5,1\}$
3	$\{2,6\},\{9\}$	$\{0,0,6,6,5\}$
4	$\{7,8\},\{\phi\}$	$\{1,7,0,1,0\}$

$MS: BABCA$; $Z = 4.4251$

Table 4.11 illustrates some steps of the procedure applied to the numerical example. A final MMUL/BS is shown in each of the small tables. To simplify the schema, only the fitness function where reductions occurred is represented in the proposed GA approach. The content of each column in these small tables is the following: (1) workstation index, k, (2) set of tasks assigned to the workstations, and (3) the idle time of the workstations at each cycle, S_{kr}. Given information under the in each small table contains the best model sequence of the final line balance for the evaluated generation, MS, and the best objective function for evaluated generation, Z.

After the evaluation of initial population, a total of 5 workstation with Z=5.1336 is detected. Beginning this solution, the proposed heuristic approach is able to reduce the objective function. The best solution found at the evaluation of the eightieth generation a total of 4 workstation with Z=4.4251. The best model sequence of the final line balance is found as BABCA. As shown in Table 4.11, there is no workload exceeding the pre-determined cycle time for the final U-line balance of the evaluated generation.

The final U-line balance that consists two of the workstations have tasks at both sides of U-line, and the others have tasks at only one side of U-line. Final U-line balance of the example is shown Figure 4.21. Model mixes of workstations for each cycle of the final U-line balance are also given in Table 4.12.

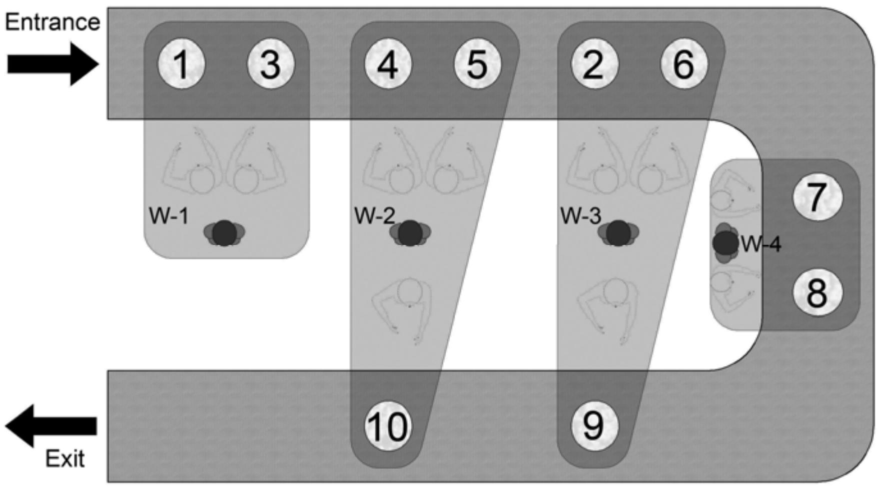


Figure 4.21 Final U-line balance of the example

Table 4.12 Model mixes of workstations for each cycle of the final U-line balance

Station (k)	**Model** (m) $\{f_k^1\},\{b_k^1\}$	**Model** (m) $\{f_k^2\},\{b_k^2\}$	**Model** (m) $\{f_k^3\},\{b_k^3\}$	**Model** (m) $\{f_k^4\},\{b_k^4\}$	**Model** (m) $\{f_k^5\},\{b_k^5\}$
1	{B},{ϕ}	{A},{ϕ}	{B},{ϕ}	{C},{ϕ}	{A},{ϕ}
2	{A},{B}	{B},{A}	{A},{B}	{B},{C}	{C},{A}
3	{C},{A}	{A},{B}	{B},{C}	{A},{A}	{B},{B}
4	{B},{ϕ}	{C},{ϕ}	{A},{ϕ}	{B},{ϕ}	{A},{ϕ}

ADW (absolute deviation of workload) was used for evaluating the workload smoothness of MMULs by Sparling and Miltenburg (1998), Miltenburg (2002), Kara et al. (2007a, 2007b) and Kim et al. (2000b, 2006). It is shown by the following equation:

$$ADW = \sum_{k=1}^{K} \sum_{r=1}^{R} \left| W_{kr} - C_{\min} \right|$$

When ADW is used alone as performance measure, handled problem may give more than one solution with the same fitness value, and line configurations of these solutions with the same fitness values can be completely different from each other. Therefore, it is not obvious that which of these solutions has more fitted line configuration. This situation becomes more important when the number of task and the number of products to be produced in the problem are increased. Our proposed performance measure may be used as secondary goal in order to determine which configuration is more stable when such situations are encountered.

For example, although the ADW of the stage 70 and the stage 100 is the same with each other (41.6) in numerical example in Table 4.11, their line configurations are completely different. Again as it can be seen in Table 4.11, our proposed performance measure ensures us to handle these problems from a different aspect and help us to determine which solution has more balanced line configuration.

Note: MMUL/BS solution of the stage 70 is equal to final solution of the Kara et al.'s (2007a) illustrative example.

4.3.8 Computational Experiments and Analysis

No comparable study dealing with the balancing and sequencing problems of mixed-model U-lines in minimizing the number of workstations (Type I) exists in the literature. Thus, to evaluate the performance of the proposed algorithm, we randomly generated different numbers of MPS (see Table 4.13) for three sets of problems. The number of tasks performed on real world U-lines varies between 1 and 24 with an average value of 10.2 (Miltenburg 2001). At this juncture, the selected sets of problems are 10-task with 3-model in Kara et al. (2007a), 19-task with 3-model in Thomopoulos (1970) and 20-task with 5-model in Kara et al. (2007b). The experiment is repeated 5 times for every test problem by taking into account only the proposed fitness function; and the minimum, mean and maximum value of the solutions were shown in the last three column of Table 4.13, respectively.

Table 4.13 Test problems

Problem	Name of problem	Number of models	Cycle time	MPS	Min.	Mean	Max.
1	Kara10	3	12	{1,1,1}	4.9586	4.9586	4.9586
2	Kara10	3	12	{2,1,2}	4.4755	4.4755	4.4755
3	Kara10	3	12	{2,2,1}	4.4251	4.4251	4.4251
4	Kara10	3	12	{2,3,2}	5.0767	5.0767	5.0767
5	Kara10	3	12	{4,2,3}	4.3116	4.3116	4.3116
6	Kara10	3	12	{5,4,2}	4.3376	4.3447	4.3732
7	Thomopoulos19	3	2.2	{1,1,1}	3.2952	3.2952	3.2952
8	Thomopoulos19	3	2.2	{2,1,2}	3.2918	3.7094	4.0183
9	Thomopoulos19	3	2.2	{2,2,1}	3.1713	3.1806	3.1947
10	Thomopoulos19	3	2.2	{2,3,2}	3.3702	3.3874	3.4389
11	Thomopoulos19	3	2.2	{4,2,3}	3.3482	3.4711	3.6085
12	Thomopoulos19	3	2.2	{5,4,2}	3.1258	3.1453	3.2168
13	Kara20	5	55	{1,1,1,1,1}	3.0396	3.0708	3.1131
14	Kara20	5	55	{2,1,1,3,2}	3.2393	3.2948	3.4326
15	Kara20	5	55	{1,3,2,2,1}	3.0973	3.1086	3.1182
16	Kara20	5	55	{5,3,2,1,1}	3.0589	3.0717	3.0889
17	Kara20	5	55	{1,2,4,5,8}	3.1869	3.1942	3.2113
18	Kara20	5	55	{1,4,8,3,1}	3.1234	3.1349	3.1573

As can be seen from Table 4.13, the value of fitness function may be varying according to different *MPS*s for the same test problem. This situation indicates that the sequence in which different models are produced cannot be set independently of the line balance. In fact, this is because of the difference of the combination of models assigned to stations according to varying cycles. As a result, the configuration of the *MPS* on the line is more important than the total number of models for the *MPS*. Actually, this is the effect of the best model sequence derived from the proposed *SA* based algorithm with regard to *MPS*. For example, tasks assigned to 5 workstations with *MPS*={2,3,2} while tasks assigned to 4 workstations with the other five *MPS*s for the test problem of Kara10.

4.4 Use of Parallel Workstations and Zoning Constraints

In this subchapter, the proposed algorithm is extended to efficiently tackle the U-shape mixed-model balancing and sequencing problem simultaneously with some particular features such as parallel workstations and zoning constraints.

The workload corresponding to the set of tasks assigned to a workstation cannot exceed the workstation's capacity, a crucial factor for the line production rate. The production rate is limited by the longest task time so that the longest task time is a lower bound on the cycle time. Parallel stations allow the reduction of the (global) cycle time of the system if certain tasks have task times longer than the desired cycle time (Buxey 1974; Pinto et al., 1981; Sarker and Shanthikumar, 1983; Bard, 1989). There are many important benefits by allowing stations to perform tasks in parallel (Buxey, 1974). One benefit is the potential improvement of balance efficiency (reduction of station idle time). Each station that is duplicated has an effective cycle time of (cycle time$\times$station multiple), thus a range of times is available and there is more likelihood of a good fit. Another benefit of using stations in parallel, is enabling to meet required high production rates (resulting in short cycle times) when some work element times exceed the required cycle time. Last benefit of a line design with stations in parallel is increased flexibility. Thus, a failure of a station stops the entire line, while a failure of a parallel station allows continuing the line operation at a reduced production rate.

Most of the works to solve assembly line balancing problem with parallel stations aim at attaining cost oriented objectives that are a trade-off between the incremental tooling/equipment cost of the duplicated workstations and the cost of hiring workers for the original line in order to satisfy the demand. Pinto et al. (1975) present a branch and bound procedure for selecting tasks to be paralleled, with the objective to minimize total cost (labor, including overtime, and equipment duplication costs). Other most important works related to cost oriented objectives are provided in Pinto et al. (1981), Johnson (1983), Bard (1989), Daganzo and Blumenfeld (1994), Askin and Zhou (1997) and Bukchin and Tzur (2000). Sarker and Shantikumar (1983) suggest a general approach that can be applied for both serial and parallel line balancing, and they define a limit on the number of parallel workstations to control the replication process. McMullen and Frazier (1997) suggest a simple heuristic procedure to solve a mixed-model assembly line problem with stochastic task times when paralleling of tasks is permitted and they allow the replication of a workstation as long as its utilization increases. In another work, they use a simulated annealing heuristic to solve the same problem for a multi-objective combined mainly of the total cost of labor and equipment and the balance efficiency (McMullen and Frazier, 1998). Vilarinho and Simaria (2002) present a two-stage simulated annealing approach to solve MALBP-1 with additional assignment restrictions and parallel stations. As secondary objective, terms for measuring vertical and horizontal imbalances are minimized. In another work, they use an ant colony optimization algorithm to solve the same problem (Vilarinho and Simaria, 2006). A detailed survey paper including parallel station on assembly line balancing problems is provided by Becker and Scholl (2006).

This thesis is the first study dealing with the balancing and sequencing problems simultaneously of the U-lines using parallel workstations and zoning constraints for minimizing the number of workstations (Type 1). In addition, this newly developed performance measure aims at the workload balance within and between workstations at the end of all cycles as secondary goal.

4.4.1 Assumptions

The U-lines with parallel workstations and zoning constraints considered in this study operates in accordance with following assumptions:

- Product models having similar production attributions are produced on the same U-shaped production lines.
- The travel times of operators and setup times are ignored.
- Precedence diagrams of different models are known, and a combined precedence diagrams is employed (Macaskill 1972).
- The completion times of tasks may differ from one model to another and can be equal to zero. Common tasks among different models exist.
- Task completion times are deterministic and independent from each others.
- Paced assembly line considered and no work-in-process is allowed.
- Minimum Part Set (MPS) principle is used (Bard et al., 1992, Merengo et al., 1999, Kara et al., 2007a, Kim et al., 2000b).

- Equally equipped workstations and fixed rate launching are considered.

4.4.2 Notations and Equations

Parallel workstations are explained by the following notations and equations: RW_k Represents that minimum how many operators must be assigned to this workstation for being able to perform the task having maximal task time under a cycle time limitation when all cycles are taken into consideration at the workstation k.

$$RW_k = \left\lceil \frac{\max(t_{i(f_k^r + b_k^r)})}{C} \quad \begin{matrix} i \in (XF_k + XB_k); \\ r = 1,..R \end{matrix} \right\rceil^{+} \quad (k = 1,...,K)$$

RP_k is a decision variable defined as follows:

$$RP_k = \begin{cases} 1; \text{ if workstation } k \text{ is required more than one operator } (RW_k > 1) \\ 0; \text{ otherwise} \end{cases} \quad (k = 1,...,K)$$

A workstation can be replicated as long as its utilization increases (McMullen and Frazier, 1997, 1998). Therefore, we expanded our algorithm in a manner that other operators can be assigned to the same workstation additionally to minimum number of operators necessary for enabling the decision making under different performance measures in cases requiring parallel station.

To obtain alternative paralleling conditions, the decision maker may define the extra replicas of operator, CP. Thus, a operator in a particular workstation can be replicated up to an upper bound on the maximum number of replicas, $MAXRP$.

$MAXRP$ is defined as follows;

$$MAXRP = RW_k + CP \quad (k = 1,...,K)$$

The total number of operators working on the assembly line (S) is computed by the sum of the number of replicas of operators in all workstations, as follows:

$$S = \sum_{k=1}^{K} [1 + RP_k (MAXRP - 1)]$$

The workload capacity of the workstation k will be equal to:

$$ST_k = C \times [1 + RP_k (MAXRP - 1)] \quad k = 1,2,...,K$$

The workload of the workstation k cannot exceed the workload capacity in any cycle. This case is shown by following equation:

$$(W_{kr} \leq ST_k) \quad k = 1,2,...,K \; ; r = 1,2,...,R$$

Also, we included zoning constraints to our problem. As mentioned in Chapter 2, zoning constraints can be either positive or negative. Positive zoning constraints force the assignment of certain tasks to a specific workstation. In the proposed approach, the tasks that need to be allocated to the same Workstation are merged and treated by the procedure as only one task. Negative zoning constraints forbid the assignment of tasks to the same workstation. In the proposed procedure, a task is not available for being assigned to a workstation if there is an incompatible task already assigned to that workstation.

4.4.3 New Objective Function

We explained new fitness function by following equations in a manner comprising also parallel workstations.

$$S_{kr} = (ST_k - W_{kr}) \; (k = 1,2,...,K), (r = 1,2,...,R) \qquad (1)$$

$$KI_k = \sum_{r=1}^{R} S_{kr} \quad (k = 1,2,...,K) \tag{2}$$

$$C_b = \frac{R}{K(R-1)} \sum_{k=1}^{K} \sum_{r=1}^{R} \left(\frac{S_{kr}}{KI_k} - \frac{1}{R} \right)^2 \tag{3}$$

$$RI_r = \sum_{k=1}^{K} S_{kr} \quad (r = 1,2,...,R) \tag{4}$$

$$C_w = \frac{K}{R(K-1)} \sum_{r=1}^{R} \sum_{k=1}^{K} \left(\frac{S_{kr}}{RI_r} - \frac{1}{K} \right)^2 \tag{5}$$

$$\min \mathrm{Z} = \mathrm{S} + C_b + C_w \tag{6}$$

S_{kr} (Eq. 1) represents the idle time of workstation k at the cycle r. The idle time of a workstation is the difference between the capacity of the workstation and its workload.

KI_k (Eq. 2) represents the total idle time at the end of all cycles in the workstation k.

In the objective function (Eq. 6), C_b (Eq. 3) aims at smoothing the workload of workstations between cycles, i.e., the idle time is distributed across all cycles as equally as possible for any workstation. The value of function C_b varies between a maximum of 1, when the total idle time of a workstation at the end of all cycles equal to only one cycle's idle time, and a minimum of 0, when the idle times of a workstation at the each cycle are equal to each other.

RI_r (Eq. 4) represents the total idle time of all workstations at the cycle r.

In the objective function (Eq. 6), C_w (Eq. 5) aims at workload balance of all workstations within any cycles, i.e., the idle time is distributed across all workstations as equally as possible at any cycle. The value of function C_w varies between a maximum of 1, when the total idle times of all workstations at any cycle equal to only one workstation's idle time, and a minimum of 0, when the idle times of each workstation at any cycle are equal to each other.

The first term (S) of the fitness function (Eq. 6) is to minimize the total number of the operators required on the line. The second term (C_b) is to smooth the workloads of workstations between cycles. The third term (C_w) is to smooth the workloads of workstations within cycles. The second and the third terms are within the value range [0, 1]. So, the model minimizes the number of workstations before the secondary goal becomes active. The proposed performance measure may vary depending on the balance and the model sequence.

Note:

If KI_k equals to 0, $\sum_{r=1}^{R}\left(\frac{S_{kr}}{KI_k}-\frac{1}{R}\right)^2$ will be equal to 0.

If RI_r equals to 0, $\sum_{k=1}^{K}\left(\frac{S_{kr}}{RI_r}-\frac{1}{K}\right)^2$ will be equal to 0.

4.4.4 Numerical illustration

We used the precedence diagram of 10-task problem (Kara et al., 2007a) to describe the characteristics of the problem of MMUL/BS with parallel workstations and zoning constraints; and the task processing times were randomly generated for describing better the characteristics of our problem.

Combined precedence diagram relationships among tasks and task completion times are given in Figure 4.22 and Table 4.14, respectively.

- The example line is running at a cycle (C) of 6 s/model, an MPS of {221}, and, thus, a length of model sequence (R) of 5,
- Tasks 9 and 10 cannot be executed on the same workstation,

- If a task time exceeds the capacity of a workstation, the number of extra replicas of operator for this workstation is given as zero ($CP = 0$),
- The control parameters of the GA and SA based algorithm are listed in Figure 4.15.
- The initial model sequence of the example U-line is selected as AABBC.

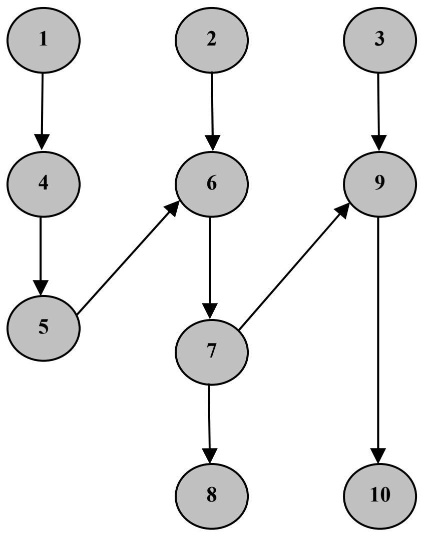


Figure 4.22 Combined precedence diagram

Table 4.14 Task completion times for the example problem

Task	Completion time		
	A	B	C
1	8	1	3
2	2	0	7
3	2	6	4
4	4	0	2
5	0	3	3
6	4	1	2
7	5	4	0
8	3	1	5
9	0	6	4
10	1	7	1

As mentioned before, each task sequence (TS) is evaluated to calculate their fitness value by using the SA based fitness evaluation approach. SA based approach starts with the proper initial SA parameters, a task sequence (TS), a feasible model sequence, at first. If the feasible solution is not found, the value of (K) is increased by one. Then, the parameters (T_{cry}, T_0, q, IT) of SA algorithm are updated with initial parameters. This case continues whenever a feasible solution is found. After a feasible solution is found, the algorithm will continue to search until the termination conditions are met.

Table 4.15 Selected control parameters

Parameters	**Value**
Initial number of workstation (K)	1
Population size	100
Number of generations	100
Crossover probability	0.95
Mutation probability	0.2
Initial temperature	100
Cooling rate	0.95
Length of the each temperature level	2
Crystallization temperature	1
Probability of p1	0.75
Probability of p2	0.5

Table 4.16 Step-by-step illustration of the solution process

Stage 0: Evaluation of the initial population

Station (k)	***Task*** (i) $\{XF_k\}$, $\{XB_k\}$	***Idle time*** (S_{kr}) $\{r = 1,2,...R\}$
1^*	{2},{10}	{9,3,11,4,5}
2^*	{1},{9}	{5,4,4,5,5}
3	{3,4},{ϕ}	{0,0,0,0,0}
4	{5,6},{ϕ}	{2,1,2,2,2}
5	{7},{ϕ}	{1,2,6,2,1}
6	{8},{ϕ}	{3,3,5,1,5}

$MS: CBAAB$; $Z = 8.1548$

Stage 1: Evaluation of the 1.generation

Station (k)	***Task*** (i) $\{XF_k\}$, $\{XB_k\}$	***Idle time*** (S_{kr}) $\{r = 1,2,...R\}$
1^*	{1},{8,10}	{3,0,3,0,3}
2	{3},{ϕ}	{0,0,4,2,4}
3	{4,5},{ϕ}	{2,3,3,2,1}
4^*	{2,6,7},{ϕ}	{3,1,7,7,1}
5	{9},{ϕ}	{6,2,6,0,0}

$MS: BACAB$; $Z = 7.3301$

Table 4.16 (cont) Step-by-step illustration of the solution process

Stage 10: Evaluation of the 10.generation

Station (k)	**Task (i) $\{XF_k\}$, $\{XB_k\}$**	**Idle time (S_{kr}) $\{r = 1,2,...R\}$**
1^*	$\{1\},\{10\}$	$\{4,3,4,3,8\}$
2	$\{3,4\},\{\phi\}$	$\{0,0,0,0,0\}$
3^*	$\{2,5,6,7\},\{\phi\}$	$\{1,0,4,1,4\}$
4	$\{9\},\{\phi\}$	$\{0,6,2,0,6\}$
5	$\{8\},\{\phi\}$	$\{3,5,3,1,5\}$

$MS : BABAC$; $Z = 7.2846$

Stage 30: Evaluation of the 30.generation

Station (k)	**Task (i) $\{XF_k\}$, $\{XB_k\}$**	**Idle time (S_{kr}) $\{r = 1,2,...R\}$**
1^*	$\{2,3\},\{10\}$	$\{5,5,1,1,0\}$
2^*	$\{1\},\{9\}$	$\{5,5,5,4,4\}$
3	$\{4,5\},\{\phi\}$	$\{2,1,3,3,2\}$
4	$\{6\},\{8\}$	$\{1,1,1,2,0\}$
5	$\{7\},\{\phi\}$	$\{2,1,1,6,2\}$

$MS : ACBBA$; $Z = 7.2251$

Stage 50: Evaluation of the 50.generation

Station (k)	**Task (i) $\{XF_k\}$, $\{XB_k\}$**	**Idle time (S_{kr}) $\{r = 1,2,...R\}$**
1^*	$\{1\},\{8,10\}$	$\{0,0,3,3,3\}$
2	$\{3,4\},\{\phi\}$	$\{0,0,0,0,0\}$
3^*	$\{2\},\{7,9\}$	$\{7,8,0,0,0\}$
4	$\{5,6\},\{\phi\}$	$\{1,2,2,2,2\}$

$MS : AACBB$; $Z = 6.6115$

Stage 60: Evaluation of the 60.generation

Station (k)	**Task (i) $\{XF_k\}$, $\{XB_k\}$**	**Idle time (S_{kr}) $\{r = 1,2,...R\}$**
1^*	$\{1\},\{8,10\}$	$\{3,3,0,3,0\}$
2	$\{3,4\},\{\phi\}$	$\{0,0,0,0,0\}$
3^*	$\{2,5,6\},\{9\}$	$\{4,6,2,0,0\}$
4	$\{7\},\{\phi\}$	$\{1,2,1,2,6\}$

$MS : BCABA$; $Z = 6.5745$

Stage 70: Evaluation of the 70.generation

Station (k)	**Task (i) $\{XF_k\}$, $\{XB_k\}$**	**Idle time (S_{kr}) $\{r = 1,2,...R\}$**
1^*	$\{1\},\{8,10\}$	$\{3,3,0,0,3\}$
2^*	$\{2\},\{7,9\}$	$\{2,0,7,0,6\}$
3	$\{3,4\},\{\phi\}$	$\{0,0,0,0,0\}$
4	$\{5,6\},\{\phi\}$	$\{2,2,2,1,2\}$

$MS : CBAAB$; $Z = 6.5529$

Table 4.16 (cont) Step-by-step illustration of the solution process

Stage 80: Evaluation of the 80.generation

Station (k ***)***	***Task (*** i ***)*** $\{XF_k\}$, $\{XB_k\}$	***Idle time (*** S_{kr} ***)*** $\{r = 1,2,...R\}$
1^*	{1},{8,10}	{3,0,3,3,0}
2^*	{2},{7,9}	{5,8,0,0,2}
3	{3,4},{ϕ}	{0,0,0,0,0}
4	{5,6},{ϕ}	{1,2,2,2,2}

$MS : BACBA$; $Z = 6.4853$

Stage 100: Evaluation of the 100.generation

Station (k ***)***	***Task (*** i ***)*** $\{XF_k\}$, $\{XB_k\}$	***Idle time (*** S_{kr} ***)*** $\{r = 1,2,...R\}$
1^*	{1},{8,10}	{3,0,3,3,0}
2^*	{2},{7,9}	{5,8,0,0,2}
3	{3,4},{ϕ}	{0,0,0,0,0}
4	{5,6},{ϕ}	{1,2,2,2,2}

$MS : BACBA$; $Z = 6.4853$ (* represents parallel workstations)

Table 4.16 illustrates some steps of the procedure applied to the numerical example. A final MMUL/BS is shown in each of the small tables. To simplify the schema, only the fitness function where reductions occurred is represented in the proposed GA approach. Followings are the content of each column in these small tables: (1) workstation index, k, (2) set of tasks assigned to the workstations, and (3) the idle time of the workstations at each cycle, S_{kr}. Given information under each small table shows the best model sequence of the final line balance for the evaluated generation, MS**,** and the best objective function for evaluated generation , Z ,. Workstations to which more than one operator is assigned are represented at the table with the sign (*).

After the evaluation of the initial population, it is determined that the line balancing consists of 6 stations in total and that the best model sequence is CBAAB at line configuration forming the best solution. Two operators each work at the first two workstations in the line balancing forming this solution, and one operator each works at others. The value of our proposed fitness function is 8, 1548. The proposed approach can reduce the value of the fitness function in next generations beginning from this solution; and the best solutions of the generations where decreases are observed only in the fitness function are shown at small tables in Table 4.16. The best solution until the satisfaction of the termination conditions of priority-based GA are found in the evaluation of the 80th generation, and the best solution did not change at the last 20 generations. It is detected that the line balancing consists of 4 workstations in the line configuration giving the best solution, and also that the model sequence is BACBA. Two operators each work at the first two workstations and one operator each works at other two workstations in the line balancing forming this solution (see Figure 4.21).

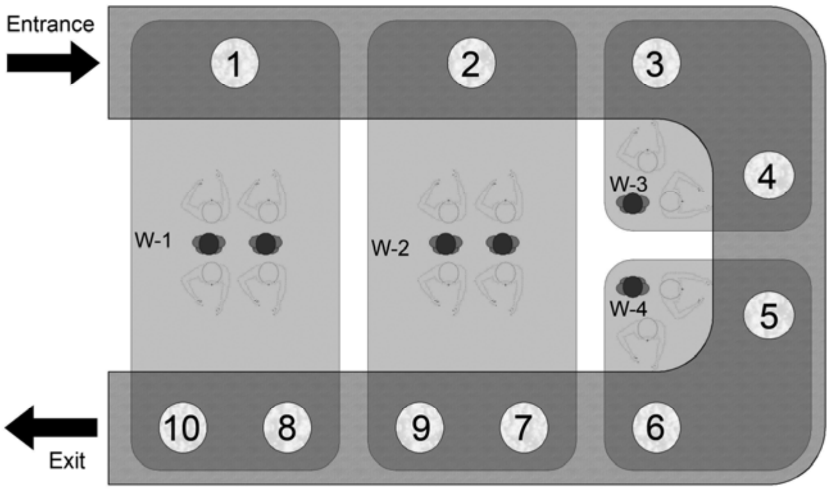


Figure 4.23 Final U-line balance of the example

Model mixes of workstations for each cycle of the final U-line balance are also given in Table 4.17.

Table 4.17 Model mixes of workstations for each cycle of the final U-line balance

Station (k)	**Model** (m) $\{f_k^1\},\{b_k^1\}$	**Model** (m) $\{f_k^2\},\{b_k^2\}$	**Model** (m) $\{f_k^3\},\{b_k^3\}$	**Model** (m) $\{f_k^4\},\{b_k^4\}$	**Model** (m) $\{f_k^5\},\{b_k^5\}$
1	{B},{B}	{A},{A}	{C},{C}	{B},{B}	{A},{A}
2	{A},{A}	{B},{C}	{A},{B}	{C},{A}	{B},{B}
3	{B},{ϕ}	{A},{ϕ}	{B},{ϕ}	{A},{ϕ}	{C},{ϕ}
4	{C},{ϕ}	{B},{ϕ}	{A},{ϕ}	{B},{ϕ}	{A},{ϕ}

4.4.5 Computational Experiments and Analysis

No comparable study dealing with the balancing and sequencing problems of mixed-model U-lines in minimizing the number of workstations (Type I) exists in the literature. Thus, we randomly generated different numbers of *MPS* (see Table 4.18) for three sets of problems in order to evaluate the performance of the proposed algorithm. The number of tasks performed on real world U-lines varies between 1 and 24 with an average value of 10.2 (Miltenburg 2001). At this juncture, the selected sets of problems are 10-tasks with 3-models in Kara et al. (2007a), 19-tasks with 3-models in Thomopoulos (1970) and 20-tasks with 5-models in Kara et al. (2007b). Task processing times of the tasks in the problem consisting of 10 tasks are randomly generated for being able to better understand general characteristics of the proposed approach. The number of extra operators to be assigned additionally in cases requiring more than one operator at any station is accepted to be 0. Negative zoning constraints between the tasks are shown at 6th column of Table 4.18. The experiment is repeated 5 times for every test problem by taking into account only the proposed fitness function; and the minimum, mean and maximum value of the solutions are shown in the last three column of Table 4.13, respectively.

Table 4.18 Test problems

Problem	**Name of problem**	**Number of models**	**Cycle time**	**MPS**	**Negative zoning constraints**	**Min.**	**Mean**	**Max.**
1	Kara10	3	6	{1,1,1}	Task 9-10	6.3158	6.3158	6.3158
2	Kara10	3	6	{2,1,2}	Task 9-10	6.5452	6.5452	6.5452
3	Kara10	3	6	{2,2,1}	Task 9-10	6.4853	6.4853	6.4853
4	Kara10	3	6	{2,3,2}	Task 9-10	7.1518	7.1518	7.1518
5	Kara10	3	6	{4,2,3}	Task 9-10	7.1496	7.1496	7.1496
6	Kara10	3	6	{5,4,2}	Task 9-10	7.1325	7.1325	7.1325
7	Thomopoulos19	3	0.8	{1,1,1}	Task 10-19	9.3552	9.3552	9.3552
8	Thomopoulos19	3	0.8	{2,1,2}	Task 10-19	9.2618	9.2651	9.3139
9	Thomopoulos19	3	0.8	{2,2,1}	Task 10-19	9.2363	9.2873	9.3635
10	Thomopoulos19	3	0.8	{2,3,2}	Task 10-19	9.2449	9.4312	10.0937
11	Thomopoulos19	3	0.8	{4,2,3}	Task 10-19	10.0776	10.1118	10.1401
12	Thomopoulos19	3	0.8	{5,4,2}	Task 10-19	9.1667	9.3786	10.0533
13	Kara20	5	14	{1,1,1,1,1}	Task 15-18	14.1981	14.4151	15.1631
14	Kara20	5	14	{2,1,1,3,2}	Task 15-18	14.1826	14.7628	15.1557
15	Kara20	5	14	{1,3,2,2,1}	Task 15-18	15.1146	15.3217	16.1238
16	Kara20	5	14	{5,3,2,1,1}	Task 15-18	13.1856	13.9255	14.1163
17	Kara20	5	14	{1,2,4,5,8}	Task 15-18	15.0751	15.2756	16.0684
18	Kara20	5	14	{1,4,8,3,1}	Task 15-18	14.1215	14.6964	15.0854

CHAPTER FIVE
CONCLUSION

Two important problems occur in mixed model U-shaped assembly lines. First one is the line balancing, and the other is model sequencing. These two problems are tightly interrelated in mixed model U-shaped assembly line, and these problems must be considered at the same time to be able to use these lines more efficiently. Therefore, these lines have more complex structures when compared to other lines.

In this thesis, simulated annealing based fitness evaluation approach is developed in order to ensure the performance of easy and effective fitness evaluations in priority based generic algorithm to be able to solve line balancing/model sequencing problems occurring in mixed model U-shape assembly lines, simultaneously.

We modified the neighborhood generating mechanism developed by Kara et al. (2007b) in a manner adaptable to the characteristics of our simulated annealing based solution method that we developed to evaluate our resulting task sequences by the method that we developed. The simulated annealing based solution method that we developed tries to find the minimum number of workstations rendering the possible solution by using the number of user defined stations and our modified fitness function (Type 1). As secondary goal, it tries to optimize workload balance between and within workstations by taking all cycles into consideration.

Our modified fitness function whose primary goal is to minimize the number of stations and as the secondary goal it tries to ensure the balance of the workload within and between stations at the end of all possible cycles is developed for responding to the use of parallel station. The number of replications can be increased in a user defined manner additionally to minimum number of replications necessary for the use of parallel stations in order to be able to allow the decision maker to make comparisons by considering different parallel station structures. This situation allows the decision maker to work in different scenarios.

This thesis is the first study dealing with simultaneously solving the balancing and sequencing problems of the mixed-model U-lines by using parallel workstations and zoning constraints to minimize the number of workstations (Type 1).

Our proposed approach is tested separately in cases with and without parallel station-zoning constraints in minimum part sets (MPS) produced for problem sets consisting of 10 tasks-3 models, 19 tasks-3 models, and 20 tasks-5 models by thinking that real life U-shaped mixed model assembly lines consist of 10.2 tasks in average.

Results showed that our proposed simulated annealing based fitness evaluation approach works with the generic algorithm very concordantly; and it is an effective method in solving simultaneous sequencing-line balancing problems in mixed-model U-lines with and without the existence of parallel station-zoning constraints.

REFERENCES

Aase, G. R., Schniederjans, M. J., & Olson, J. R. (2003). U-OPT: an analysis of exact U-shaped line balancing procedures. *International Journal of Production Research*, 41, 4185-4210.

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

Agpak, K., & Gokcen, H. (2007). A chance-constrained approach to stochastic line balancing problem. *European Journal of Operational Research*, 180, 1098–1115.

Ajenblit, D. A., & Wainwright, R. L. (1998). Applying genetic algorithms to the U-shaped assembly line balancing problem. *In the Proceeding of the 1998 IEEE International Conference on Evolutionary Computation,* Anchorage, Alaska, USA, 96-101.

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

Bard, J. F. (1989). Assembly line balancing with parallel workstations and dead time. *International Journal of Production Research*, 27, 1005-1018.

Bard, J. F., Dar-el, E. M., & Shtub, A. (1992). An analytic framework for sequencing mixed model assembly lines. *International Journal of Production Research,* 30, 35-48.

Batini, D., Faccio, M., Ferrari, E., Persona, A., & Sgarbossa, F. (2007). Design configuration for a mixed-model assembly system in case of low product demand.

The International Journal of Advanced Manufacturing Technology, 34(1-2), 188-200.

Baudin, M. (2002). *Lean Assembly: The nuts and bolts of making assembly operations flow*. Productivity, New York.

Baybars, I. (1986). A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem. *Management Science*, 32, 909-932.

Baykasoglu, A., & Ozbakir, L. (2007). Stochastic U-line balancing using genetic algorithms. *International Journal of Advanced Manufacturing Technology*, 32, 139–147.

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

Boysen, N., & Fliedner, M. (2008). A versatile algorithm for assembly line balancing. *European Journal of Operational Research*, 184, 39–56.

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

Bukchin, J., & Tzur, M. (2000). Design of flexible assembly line to minimize equipment cost. *IIE Transactions*, 32(7), 585-598.

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

Buxey, G. M. (1974). Assembly line balancing with multiple stations. *Management Science*, 20, 1010-1021.

Cheng, C.H., Miltenburg, J., & Motwani, J. (2000). The effect of straight- and u-shaped lines on quality. *IEEE Transactions on Engineering Management*, 47, 321-334.

Choi, G. (2009). A goal programming mixed-model line balancing for processing time and physical workload. *Computers and Industrial Engineering*, 57, 395–400.

Daganzo, C. F., & Blumenfeld, D. E. (1994). Assembly system design principles and trade-offs. *International Journal of Production Research*, 32, 669-681.

Dar-el, E. M., & Navidi, A. (1981). A mixed-model sequencing application. *International Journal of Production Research,* 19, 69-84.

Deckro, R. F., & Rangachari, S. (1990). A goal approach to assembly line balancing. *Computers and Operations Research*, 17, 509–521.

Duplaga E. A., & Bragg D. J. (1998). Mixed-model assembly line sequencing heuristics for smoothing component parts usage: a comparative analysis. *International Journal of Production Research*, 36(8), 2209–2224.

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

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

Erel, E., Sabuncuoglu, I., & Sekerci H. (2005). Stochastic assembly line balancing using beam search. *International Journal of Production Research*, 43(7), 1411–1426.

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

Gen, M., & Cheng, R. (1997). *Genetic Algorithms & Engineering Design.* New York: John Wiley & Sons.

Gen, M., & Cheng, R. (2000). *Genetic Algorithms & Engineering Optimization.* New York: John Wiley & Sons.

Gen, M., Cheng, R., & Lin, L. (2008). *Network Models and Optimization*. Springer-Verlag London Limited.

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

Gokcen, H., & Agpak, K. (2006). A goal programming approach to simple U-line balancing problem. *European Journal of Operational Research*, 171, 577-585.

Gokcen, H., Agpak, K., Gencer, C., & Kizilkaya, E. (2005). A shortest route formulation of simple U-type assembly line balancing problem. *Applied Mathematical Modelling,* 29, 373–380.

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

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

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

Goldberg, D. E. (2002). *Design of innovation: Lessons from and for competent genetic algorithms*. Boston, MA: Kluwer Acadamic Publishers.

Grupe, F. H., & Jooste, S. (2004). Genetic algorithms: A business perspective. *Information Management & Computer Security, 12* (3), 289-298.

Guerriero, F., & Miltenburg, J. (2003). The stochastic U-line balancing problem. *Naval Research Logistics*, 50, 31-57.

Gutjahr, A. L., & Nemhauser, G. L. (1964). An algorithm for the line balancing problem. *Management Science*, 11 (2), 308–315.

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

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

Hwang, R. K., Gen, M., & Katayama, H. (2006). A performance evaluation of multiprocessor scheduling with genetic algorithm. *Asia Pacific Management Review*, 11, 67–72.

Hwang, R. K., Katayama, H., & Gen, M. (2008). U-shaped assembly line balancing problem with genetic algorithm. *International Journal of Production Research*, 46(16), 4637–4649.

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

Jackson, J. R. (1956). A Computing Procedure for a Line Balancing Problem. *Management Science,* 2, 261-272.

Johnson, R. V. (1983). A branch and bound algorithm for assembly line balancing problems with formulation irregularities. *International Journal of Production Research*, 19, 277-287.

Johnson, R. V. (1988). Optimally balancing large assembly lines with fable. *Management Science*, 34, 240–253.

Jin, Y., Olhofer, M., & Sendhoff, B. (2002). A Framework for Evolutionary Optimization with Approximate Fitness Functions. *IEEE Transactions on Evolutionary Computation*, 6(5), 481-494.

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

Kara Y., Ozcan, U., & Peker, A. (2007b). Balancing and sequencing mixed-model just-in-time U-lines with multiple objectives. *Applied Mathematics and Computation*, 184, 566–588.

Kara, Y., & Tekin, M. (2009). A mixed integer linear programming formulation for optimal balancing of mixed-model U-lines. *International Journal of Production Research*, 47(15), 4201–4233.

Karabati, S., & Sayin, S. (2003). Assembly line balancing in a mixed-model sequencing environment with synchronous transfers. *European Journal of Operational Research*, 149(2), 417–429.

Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller R.E & Thatcher J.W. Editors: *Complexity of Computer Applications*, 85-104, New York: Plenum Press.

Kilbridge, M. D., & Wester, L. (1961). A heuristic method of assembly line balancing. *The Journal of Industrial Engineering*, 12(4), 292-298.

Kim, Y. K., Kim, J. Y., & Kim, Y. (2000a). A Coevolutionary Algorithm for Balancing and Sequencing in Mixed Model Assembly Lines. *Applied Intelligence*, 13, 247-258.

Kim, Y. K., Kim, J. Y., & 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 (2006), 838–852.

Kim, Y. K., Kim, S. J., & Kim, J. Y. (2000b). Balancing and sequencing mixed model U-lines with a co-evolutionary algorithm. *Production Planning and Control*, 11, 754-764.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220, 671-680.

Macaskill, J. L. C. (1972). Production-line balances for mixed model lines. *Management Science* 19(4), 423–434.

Martinez, U., & Duff, W. S. (2004). Heuristic approaches to solve the U-shaped line balancing problem augmented by Genetic Algorithms. *In the Proceedings of the 2004 Systems and Information Engineering Design Symposium*, 287-293.

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

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

McMullen, P. R., & 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.

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

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics,* 21(6), 1087–1092.

Miltenburg, J. (1989). Level schedules for mixed-model assembly lines in just-in-time production systems. *Management Science*, 35(2):192–207.

Miltenburg, J. (1998). Balancing U-lines in a multiple U-line facility. *European Journal of Operational Research*, 109, 1-23.

Miltenburg, J. (2001). U-shaped production lines: a review of theory and practice. *International Journal of Production Economics*, 70, 201-214.

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

Miltenburg, J., & Sinnamon, G. (1989). Scheduling mixed model multi-level just-in-time production systems. *International Journal of Production Research*, 27(9), 1487–1509.

Miltenburg, J., & Sinnamon, G. (1992). Algorithms for scheduling multi-level just-in-time production systems. *IIE Transactions,* 24(2), 121–130.

Miltenburg, J., & Sinnamon, G. (1995). Revisiting the mixed-model multi-level just-in-time scheduling problem, *International Journal of Production Research*, 33, 2049–2052.

Miltenburg, J., & Wijngaard, J. (1994). The U-line line balancing problem. *Management Science*, 40(10), 1378-1388.

Monden, Y. (1993). *Toyota production system, 2nd edition*. Engineering and Management Press, Norcross, Georgia.

Moodie, C. L., & Young, H. H. (1965). A heuristic method of assembly line balancing for assumptions of constant or variable work element times. *Journal of Industrial Engineering*, 16, 23-29.

Montgomery, D. C., & Runger, G. C. (2005). *Applied Statistics and Probability for Engineers.* New York: John Wiley & Sons.

Pinto, P. A., Danneenbring, D. G., & Khumawala, B. M. (1975). A branch and bound algorithm for assembly line balancing with paralleling. *International Journal of Production Research*, 13, 183-196.

Pinto, P. A., Dannenbring, D. G., & Khumawala, B. M. (1981). Branch and bound and heuristic procedures for assembly line balancing with paralleling of stations. *International Journal of Production Research*, 19, 565-576.

Ponnambalam, S. G., Aravindan, P., Naidu, G., & Mogileeswar, G. (2000). Multiobjective genetic algorithm for solving assembly line balancing problem. *International Journal of Advanced Manufacturing Technology*, 16(5), 341-352.

Rekiek, B., & Delchambre, A. (2006). *Assembly line design: The balancing of mixed-model hybrid assembly lines with genetic algorithms*. Springer Series in Advanced Manufacturing, London.

Sabuncuoglu, I., Erel, E., & Alp, A. (2009). Ant colony optimization for the single model U-type assembly line balancing problem. *International Journal of Production Economics*, 120, 287–300.

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

Sarker, B. R., & Shantikhumar, J. G. (1983). A generalised approach for serial or parallel line balancing. *International Journal of Production Research*, 21, 109-133.

Sastry, K., & Goldberg, D. (2005). *Genetic Algorithm (Search Methodologies)*. Springer.

Scholl, A. (1993). *Data of assembly line balancing problems*. Schriften zur Quantitativen Betriebswirtschaftslehre 16/93, TU Darmstadt.

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

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

Scholl, A., & Klein, R. (1999). ULINO: optimally balancing U-shaped JIT assembly lines. *International Journal of Production Research,* 7(4), 721–736.

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

Sivanandam, S. N., & Deepa, S. N. (2008). *Introduction to Genetic Algorithms*. Springer-Verlag Berlin Heidelberg.

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

Talbot, F. B., Patterson, J. H., & Gehrlein, W. V. (1986). A comparative evaluation of heuristic line balancing techniques. *Management Science*, 32, 430–454.

Thomopoulos, N. T. (1967). Line balancing-sequencing for mixed-model assembly. *Management Science*, 14(2), 59–75.

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

Toklu, B., & Ozcan, U. (2008). A fuzzy goal programming model for the simple U-line balancing problem with multiple objectives. *Engineering Optimization,* 40(3), 191–204.

Urban, T. L. (1998). Optimal balancing of U-shaped assembly lines. *Management Science*, 44, 738-741.

Urban, T. L., & Chiang, W. (2006). An optimal piecewise-linear program for the U-line balancing problem with stochastic task times. *European Journal of Operational Research,* 168, 771–782.

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

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

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