DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

A NOVEL LINE BALANCING PROBLEM: COMPLEX CONSTRAINED ASSEMBLY LINE BALANCING

by
Aliye Ayça SUPÇİLLER

May, 2010
İZMİR

A NOVEL LINE BALANCING PROBLEM: COMPLEX CONSTRAINED ASSEMBLY LINE BALANCING

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

by
Aliye Ayça SUPÇİLLER

May, 2010
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"A NOVEL LINE BALANCING PROBLEM: COMPLEX CONSTRAINED ASSEMBLY LINE BALANCING"** completed by **ALİYE AYÇA SUPÇİLLER** under supervision of **ASSOC. PROF. DR. LATİF SALUM** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Latif SALUM

Supervisor

Asst. Prof. Dr. Şeyda A. TOPALOĞLU	Asst. Prof. Dr. Ahmet ÖZKURT
Thesis Committee Member	Thesis Committee Member
Asst. Prof. Dr. Özcan KILINÇCI	Prof. Dr. M. Bülent DURMUŞOĞLU
Examining Committee Member	Examining Committee Member

Prof. Dr. Mustafa SABUNCU
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

First, and foremost, I would like to thank my supervisor Assoc. Prof. Dr. Latif SALUM for his insights and creative thinking on my research project. He encouraged me during my doctoral program with his invaluable guidance and suggestions. I would like to thank to thesis committee members, Asst. Prof. Dr. Şeyda A. TOPALOĞLU and Asst. Prof. Dr. Ahmet ÖZKURT, for their valuable comments and suggestions. In addition, I would like to thank to Prof. Dr. Bülent DURMUŞOĞLU for his valuable comments and suggestions.

I would like to express my appreciation to the members of Industrial Engineering Department of Dokuz Eylul University for supporting, hospitality and tolerance. I am grateful to all my instructors and professors in Istanbul Technical University, Pamukkale University and Dokuz Eylul University for equipping me with their knowledge and academic skills. I have already been a member of Pamukkale University and I would like to thank to the staff of Industrial Engineering due to their patience.

I would like to express my special thanks to Olcay POLAT and Alper HAMZADAYI for their help and suggestions. I would like to express my special thanks for their best friendship and encouragement to Asst. Prof. Dr. Özcan KILINÇCI, Asst. Prof. Dr. Bilge BİLGEN, Asst. Prof. Dr. Özlem UZUN ARAZ, Nazan GÜNEY and all my colleagues, who are special for me and made my life enjoyable.

This dissertation is dedicated to the memory of my father, Ömer Fevzi Kayalıoğlu. I am forever grateful to my all family but especially to my mother and my sister for their sincere support and tolerance. I would also like to express my deep gratitude to my husband, Murat, for his endless love, patience and support.

Aliye Ayça SUPÇİLLER

A NOVEL LINE BALANCING PROBLEM: COMPLEX CONSTRAINED ASSEMBLY LINE BALANCING

ABSTRACT

The primary aim of this dissertation is to extend the rule-based assembly modeling and to introduce a novel assembly line balancing problem: *complex-constrained assembly line balancing problem* (CCALBP), which is of the general ALBPs, in order to model all assembly constraints through a rule-base to tackle alternative ways of assembling a product and their effects on task times, precedence relations and the line balance simultaneously.

A genetic algorithm (GA) based on the rule-base is proposed and discussed in detail to solve CCALBP. The specific characteristics of the proposed GA are explained on an example problem. The control parameters of the GA are optimized to improve the performance. Since CCALBP is a novel problem, there is no set of benchmark instances for testing. Therefore, the computational experiments are carried out on a set of self-made instances generated by adapting well-known benchmark problems from the literature. Some alternative routes are created and added to these literature problems. Based on the experiments, the proposed GA is proven to perform better. It is shown that line balancing improves when more alternatives are added to CCALBP.

It is also shown how to map a rule-based assembly model to a constraint programming (CP) model and an integer programming (IP) model. CCALBP can be solved only through rule-based modeling, but not graph-based modeling. The efficiency and modeling capability of CP and IP models are discussed, and compared with that of traditional precedence graphs.

Keywords: Assembly line balancing, Precedence constraints, Rule-based representation, Genetic algorithms

YENİ BİR MONTAJ HATTI DENGELEME PROBLEMİ: KARMAŞIK KISITLI MONTAJ HATTI DENGELEME

ÖZ

Bu doktora çalışmasının temel amacı, kural tabanlı montaj modellemesini genişletmek ve bir ürünün tüm alternatif montaj yolları ile bunların iş süreleri, öncelik ilişkileri ve hat dengesi üzerindeki etkilerini aynı anda ele almak amacıyla tüm montaj kısıtlarını bir kural tabanı ile modellemek için genel montaj hattı dengeleme problemlerinden olan yeni bir montaj dengeleme problemini, karmaşık kısıtlı montaj hattı dengeleme problemini (KKMHDP), tanıtmaktır.

KKMHDP'ni çözmek için kural tabanıyla bütünleşmiş bir genetik algoritma (GA) önerilmiş ve detaylıca tartışılmıştır. Önerilen GA'nın performansını iyileştirmek için kontrol parametreleri en uygun hale getirilmiştir. KKMHDP yeni bir problem olduğu için, test etmek için kıyaslama örnekleri seti yoktur. Bu nedenle, deneyler literatürden iyi bilinen kıyaslama problemlerinden adapte edilerek oluşturulan problem setleri ile yapılmıştır. Bazı alternatif rotalar yaratılmış ve bu literatür problemlerine eklenmiştir. Deneylere göre, önerilen genetik algoritma daha iyi sonuçlar vermiştir. KKMHDP'ne yeni alternatifler eklendikçe hat dengelemenin geliştiği gösterilmiştir.

Çalışmada bir kural tabanlı modelin kısıt programlama modeline ve tamsayılı programlama modeline nasıl eşleştirildiği de gösterilmiştir. KKMHDP, grafik tabanlı modelleme ile değil, yalnızca kural tabanlı modelleme ile çözülebilmektedir. Kısıt programlama modeli ve tamsayılı programlama modelinin modelleme kabiliyetleri ve etkinlikleri tartışılmış, geleneksel öncelik diyagramları ile karşılaştırılmıştır.

Anahtar Sözcükler: Montaj hattı dengeleme, Öncelik kısıtları, Kural tabanlı gösterim, Genetik algoritmalar

CONTENTS

CHAPTER ONE

INTRODUCTION

1.1 Background and Motivations

In ancient times assembly techniques were used to make tools, weapons, ships, machinery, furniture, and garment. Manufacturing and assembly systems evolved time by time and two important principles were introduced during Industrial Revolution. The first principle is *division of labor* (work simplification, standardization, and specialization) argued by Adam Smith in his book in 1776, and the second one is *interchangeable parts* (individual components that make up the final product must be interchangeable) based on efforts of Eli Whitney and others at the beginning of the nineteenth century. In the mid- and late- 1800s, modern production lines were used in meat packing plants. After an automotive industrialist, Henry Ford, had observed these plants, he designed and invented an assembly line with his friends (Groover, 2001).

Originally, assembly lines were developed in order to deal with mass production of standardized products in a cost efficient way (Boysen, Fliedner, & Scholl, 2007). Mass production was characterized by specialization of equipment and labor. A single product was manufactured in large quantities with a high productivity by designing and balancing dedicated assembly lines (Bukchin, Dar-el, & Rubinovitz, 2002).

Recently, mass production has been challenged by mass customization. Production systems and supply chains must be designed to handle high variety of products while at the same time achieve mass production quality and productivity (Hu, Zhu, Wang, & Koren, 2008). They are needed to be flexible and responsive to changes in demand for different product types. Today, assembly lines are still up to date, because the principle to increase productivity by division of labor is timeless (Amen, 2001). Assembly lines gain importance even in low volume production of customized products (Scholl & Becker, 2006).

An *assembly line* is a production line which consists of a number of workstations where assembly tasks are performed by human workers or automation. Products are assembled as they move along the line. Work pieces are moved from station to station manually or by a material transport system. The decision problem of optimally partitioning the assembly work among the stations with respect to some objectives is known as the *assembly line balancing problem* (ALBP) (Scholl, 1999).

The ordering in which tasks must be performed in an assembly line are called *precedence constraints*. They are technological restrictions or physical sequencing requirements on the assembly line. A *precedence graph* is generally used to represent the precedence constraints. But, there are some shortcomings of the precedence graphs. They usually fail to represent all the possible assembly sequences of a product in a single graph (Lambert, 2006), and exclude some logic statements, e.g., the precedence relation "(2 or 3) → 7" cannot be represented properly on a precedence graph (De Fazio & Whitney, 1987). Hence, they allow limited flexibility.

One or more parts of a product's assembly process may admit alternative precedence sub-graphs. Because of the great difficulty of the problem and the impossibility of representing alternative sub-graphs in a precedence graph, a line designer selects, a priori, one of such alternative sub-graphs (Capacho & Pastor, 2008).

Precedence graphs fail to describe some complicated constraints, e.g., constraints indicating that some pairs of tasks cannot be assigned into the same station because of incompatibility between them caused by some technological factors (Park, Park, & Kim, 1997).

Alternative ways of assembling a product and their effects on task times, precedence relations and the line balance should be tackled simultaneously. In this regard, a rule-based assembly model is proposed in this dissertation to address this issue.

1.2 Objectives and Research Methodology

The main objective of this dissertation is to extend the rule-based assembly modeling and to introduce the *complex-constrained assembly line balancing problem* (CCALBP), which is of the general ALBPs, in order to model all assembly constraints through a rule-base to tackle alternative ways of assembling a product and their effects on task times, precedence relations and the line balance simultaneously.

This dissertation addresses a new ALBP that has not been considered in the literature before. Hence, the main objectives of this dissertation are to define, to formalize and to solve CCALBP.

CCALBP is defined and explained with an illustrative example. In order to formalize the problem, constraint programming and integer programming formulations are developed and are used to solve some illustrative problems.

To show how to model all assembly constraints through the well known If-then rules, and how to solve CCALBP, a genetic algorithm (GA) based on the rule-based model is proposed.

Since CCALBP is a new problem, benchmark problems are generated for computational experiment to evaluate the proposed GA.

1.3 Outline of the Thesis

This dissertation is divided into seven chapters. The present chapter briefly introduces the theme of the study, points out the novel problem and presents the main objectives of the work.

Chapter 2 gives an overview of the ALBP. It presents the main characteristics of assembly line systems and defines the ALBP. Different types of ALBPs and particular solution methods to tackle the line balancing problems are also presented.

Chapter 3 describes the main characteristics of the selected meta-heuristic, GA.

Chapter 4 is dedicated to review the available literature on application of GAs to solve ALBPs. The literature review of GA applications on line balancing problems according to their specifications is given in a chronological order.

In Chapter 5, a novel problem, the complex-constrained assembly line balancing problem (CCALBP), is introduced. Rule-based modeling of assembly constraints is discussed through an illustrative example. Mapping the rule-based model into the constraint programming (CP) model and into the integer programming (IP) model is shown. The CP model and IP model are developed to formally describe CCALBP. The performance of the developed mathematical programming models is evaluated by using commercial optimization software ILOG OPL Studio (2003).

In Chapter 6, a GA based on the rule base is proposed to solve CCALBP. The proposed GA is explained through an example step by step. Since CCALBP is a new problem, benchmark problems are generated. Conclusions are withdrawn based on a set of computational experiments. An industrial case study is also presented.

Finally, the summary and the contributions of the dissertation are pointed out with the directions for future research in Chapter 7.

CHAPTER TWO

ASSEMBLY LINE BALANCING

2.1 Introduction

The aim of this chapter is to provide an overview of the main features of assembly lines and to introduce the basic concepts on assembly line balancing. This chapter is organized as follows: First, the main features and additional characteristics of assembly line systems are given. Next, the assembly line balancing problem is described in detail with the classification schemes. Then, the most common solution methods of the problem presented in the literature are discussed. Finally, the chapter is summarized.

2.2 Assembly Lines

Assembly lines are most commonly used methods in a mass production environment, because they allow the assembly of complex products by workers with limited training, by dedicated machines and/or by robots.

In an assembly line, products are assembled as they move along the line, visiting each workstation sequentially. Assembly tasks are performed at each station. Raw material or semi-finished product enters at the one end and the desired product comes out from the other end of the assembly line. The designers aim at increasing the efficiency of the assembly line by maximizing the ratio between throughput and the total cost required (Rekiek, Dolgui, Delchambre, & Bratcu, 2002).

In this section, a terminology is first given to describe assembly lines. Then, additional characteristics of assembly line systems are given in order to understand the assembly line balancing problem.

2.2.1 Terminology

The terminology for the basic concepts of an assembly line based on Scholl (1999) is given below:

Assembly: It is the process of putting two or more parts, subassemblies, and components together in order to make a finished product.

Assembly line: It is a production line that consists of a sequence of workstations arranged along a conveyor belt or a similar mechanical material handling equipment. The workpieces are consecutively launched down the line and are moved from station to station. At each station, a task is performed on each unit.

Task: It is a portion of total work content in an assembly process, having an operational processing time and a set of precedence relations. Tasks (*operations*) are considered indivisible; they cannot be split into smaller work elements without unnecessary additional work. When all tasks are allocated to the workstations, a feasible solution will be obtained.

Task time (t_i): The time required to perform a task.

Workstation (Station): It is a part of an assembly line where a certain amount of work (a set of assigned tasks) is manually performed by workers using simple tools or by semi-automated machines.

Workstation time (Station time): It is total time of the tasks allocated in the workstation. Each task assignment process updates the workstation time by adding the time of the new assigned task to the time of the previous assigned tasks.

Cycle time (C): The interval of time between the completions of successive products. In the case of paced assembly lines, the cycle time represents the maximum amount of time a product (or a job) can be processed by a station necessary. In unpaced flow lines, the cycle time is the maximum possible average station time. The cycle time must not exceed the station time, and it must not be less than maximum task time on the assembly line. *Idle time* is a positive difference between the cycle time and the station time. The sum of idle times of all stations in the line is called the *delay time*. The planning department asks for the *desired cycle time* (C), but due to failures or setup-times the real cycle time by which the line will operate is the *effective cycle time* (EC).

Precedence constraints: The technological restrictions or/and physical sequencing requirements on the assembly line.

Precedence graph (diagram): A graphical representation of the sequence of tasks as defined by the precedence constraints. The partial ordering in which tasks must be performed is illustrated by means of a precedence graph. Nodes symbolize tasks, and arrows connecting the nodes indicate the precedence relations. The sequence proceeds from left to right. For example, in Figure 2.1, task 4 is preceded by tasks 1 and 2, and task 5 is preceded by tasks 3 and 4.

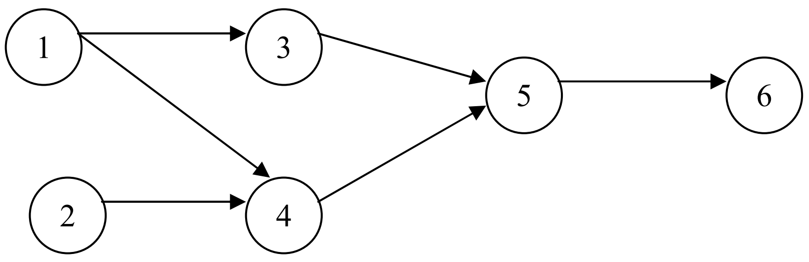
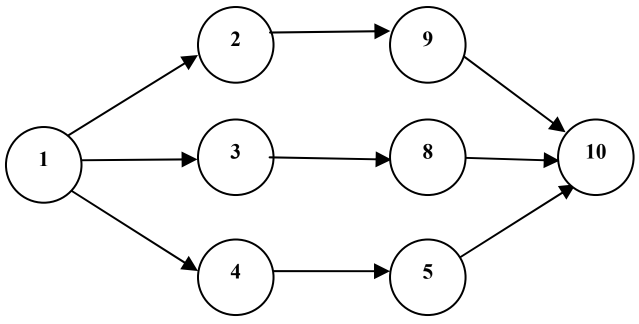
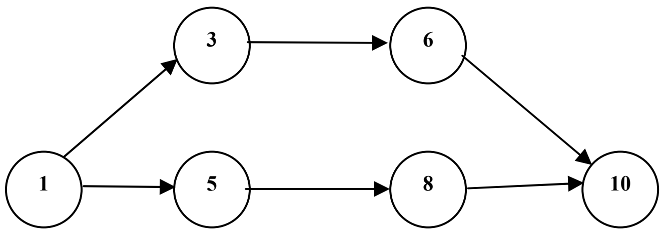


Figure 2.1 A precedence graph

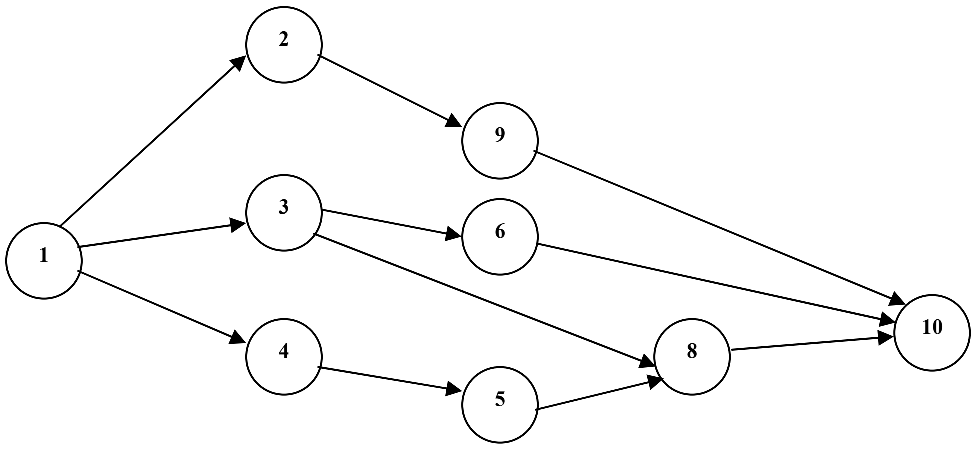
Combined precedence graph (diagram): A graphical representation that alters different models of a product into one equivalent single model. A product family is composed of several product variants. Each variant has its own distinctive tasks, but also shares some common tasks (Macaskill, 1972). Precedence relations for a set of models of a product family are defined by a single graph instead of different graphs as given in Figure 2.2.



(a)



(b)



(c)

Figure 2.2 Precedence diagrams of (a) model 1, (b) model 2 and (c) combined.

Line efficiency (E): A measure for the capacity utilization of the line. It is computed as follows (n: number of stations):

$$E\,(\%) = \left[\sum_{i=1}^{n} t_i \Big/ (n \times C) \right] \times 100 \qquad (2.1)$$

Balance delay ratio (BR): A measure of the line efficiency which results from idle time due to the imperfect allocation of tasks among stations. The unused capacity is reflected by this ratio. It is computed as follows:

$$BR\,(\%) = 1 - E = \frac{n \times C - \sum_{i=1}^{n} t_i}{n \times C} \times 100 \qquad (2.2)$$

2.2.2 Characteristics of Assembly Lines

In the literature, various classification schemes of assembly lines are given by Baybars (1986), Ghosh & Gagnon (1989), Erel & Sarin (1998), Scholl (1999), Rekiek et al. (2002), and Boysen, Fliedner, & Scholl (2008). Scholl (1999) classified assembly lines as in Figure 2.3. The continuous lines show that any combination of characteristics is typical; broken lines indicate that it is unusual.

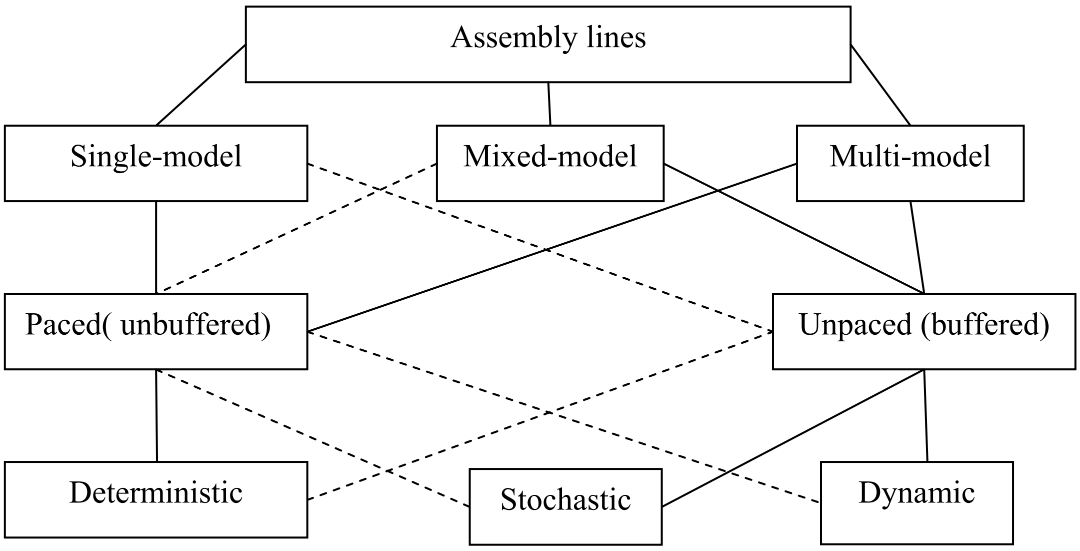


Figure 2.3 Classification of assembly lines (Scholl, 1999)

2.2.2.1 Product Variety

Because of the versatility of human workers, the design of assembly lines has to deal with differences in assembled products (Groover, 2001). The number and variety of products to be assembled on the same line have an important influence on the line architecture. With respect to product variety, there are three types of assembly lines described below.

Single-Model Lines: Only one homogeneous product is continuously manufactured in large quantities.

Mixed-Model Lines: Several models of a basic product are manufactured on the same line in an arbitrarily inter-mixed sequence.

Multi-Model Lines: Family of products which present significant differences in processes are manufactured on one or several assembly lines separately in batches.

The different line types are illustrated in Figure 2.4, where different models are symbolized by different geometric shapes. Depending on these line types, balancing problems for single-model, mixed-model and multi-model versions of assembly lines are modeled and solved.

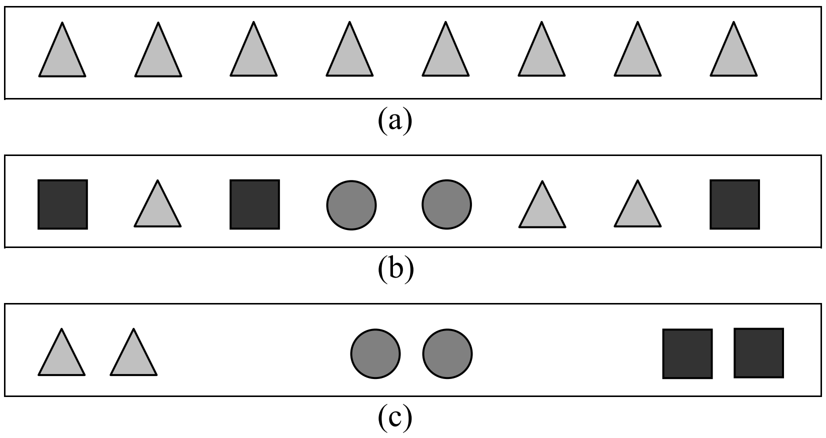


Figure 2.4 Assembly lines for (a) single-model, (b) mixed-model, and (c) multi-model.

2.2.2.2 Line Control

With respect to the line control, there are assembly lines that can be designed with alternatives as given below (Groover, 2001).

Paced Lines: In case of a paced assembly line, each workstation is given the same amount of time to perform tasks assigned to the workstation. The synchronization is achieved by transferring the jobs between stations at pre-determined and fixed time intervals. This transfer takes place irrespective of whether or not the individual stations complete their task. The station time of each station is limited to the cycle time as a maximum value for each workpiece. Therefore, in paced lines, there is a fixed production rate equal to the reciprocal of the cycle time. The pace is either kept by a continuous material handling equipment, e.g. a conveyor belt, or by an intermittent transport.

Unpaced Lines: In the absence of a common cycle time, workpieces may have to wait before they can enter the next station(s) and/or may get idle when they have to wait for the next workpiece. Workpieces are transferred when all tasks are completed, rather than being a bound to a given time span. Under *asynchronous* movement, a workpiece is always moved as soon as all tasks of a station are completed and the next station is not blocked anymore by another workpiece. By buffers between the stations, these difficulties can be partially overcome. If there is too much variability in the task process times, it is preferable to have unpaced or asynchronous line. In such a line, each station works at its own pace and advances the part to the next station whenever it completes its assigned tasks. Under *synchronous* movement, all stations wait for the slowest station to finish all tasks before workpieces are transferred at the same point in time. Buffers are then not necessary (Boysen et al., 2008).

2.2.2.3 Variability of Task Times

The task processing time is an important parameter for assembly lines. The nature of tasks and the skills of operators or the reliability of the machines can change the task processing time. All these variations have a great influence on the assembly line (Rekiek et al., 2002). With respect to variations of task times; there are three types of assembly lines described below.

Deterministic Time: The task times are considered to be deterministic (constant or known with certainty) whenever the expected variance of task times is sufficiently small, as in case of highly qualified and motivated workers or highly reliable automated stations (Johnson, 1983).

Stochastic Time: Significant variations of task times due to the work rate, skill and motivation of the workers, and the failure sensitivity of complex processes require considering task times to be stochastic (Robinson, McClain, & Thomas, 1990) rather than to be fixed at a known value.

Dynamic Time: Systematic reductions are possible due to learning effects (Toksari, Isleyen, Guner, & Baykoc, 2008, 2010) or successive improvements of the production process.

2.2.2.4 Line Configuration

The flow of materials partially determines the layout of flow-line production systems. There exist several line configurations (Becker & Scholl, 2006).

Serial Lines: Single stations are arranged in a straight line along a linear conveyor belt. Operators perform tasks on a continuous portion of the line. Figure 2.5 illustrates a serial line.

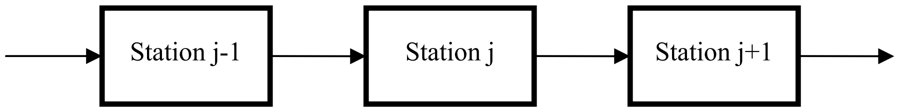


Figure 2.5 Configuration of serial lines

U-Shaped Lines: Both ends of the line are close to each other to form a narrow "U" shape. Operators can move between the two segments of the line to perform combinations of tasks (Miltenburg & Wijngaard, 1994). Thus, there are improvements in the visibility of the whole process and communication of workers. Job enrichment and enlargement lead to higher motivation, improved quality of products and increased flexibility (Rekiek et al., 2002). Figure 2.6 illustrates the configuration of U-shaped lines.

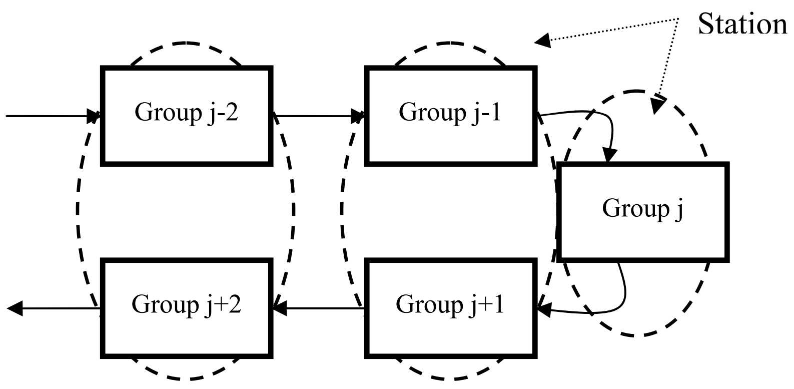


Figure 2.6 Configuration of U-shaped lines

Parallel Lines: When the demand is high enough, it is common to duplicate the entire assembly line. This has the advantage of shortening the assembly line, but may require more equipment and tooling. If failure occurs at a given station, other lines can continue to run. This reduces the risk of production stops. Parallel lines increase flexibility with better line balances and horizontal job enlargement (Rekiek & Delchambre, 2006). An example of the use of parallel lines is shown in Figure 2.7.

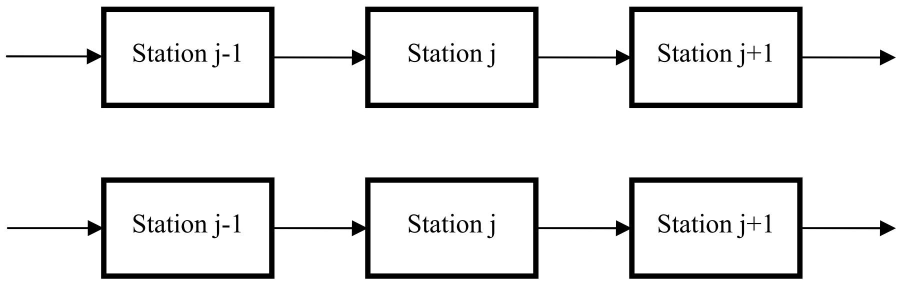


Figure 2.7.Configuration of parallel lines

Parallel Stations: There are many advantages of parallelization even by installing parallel stations in a single line. Each station in a set of parallel stations performs similar activities. The workpieces are distributed among the operators who perform the same tasks. This is a common layout when a series of product variations are being manufactured. If certain task times exceed the desired cycle time, parallel stations allow decreasing the cycle time (Becker & Scholl, 2006). Figure 2.8 illustrates the configuration of parallel stations.

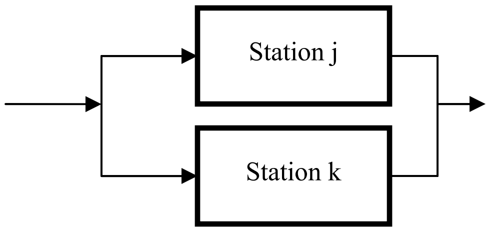


Figure 2.8 Configuration of parallel stations

Two-sided Line: In the assembly of large-sized and heavy workpieces, such as trucks and buses, both the left-side and the right-side of the line are used in parallel. The operators working in opposite sides of the line perform their tasks on the same component simultaneously. In two-sided assembly lines, some tasks can be assigned to only one side of the two sides: L (left) and R (right)-type tasks, while others can be assigned to either side of the line: E (either)-type tasks.

2.3 The Assembly Line Balancing Problem

The installation of an assembly line is a long-term decision and usually requires large capital investments. Therefore, it is important to design and balance an assembly line in a way that it should work as efficiently as possible. Most of the studies related to the assembly lines concentrate on the assembly line balancing. The *assembly line balancing* is the allocation of the tasks among stations so that the precedence relations are not violated and a given objective function is optimized. The *assembly line balancing problem* (ALBP) deals with balancing the assembly line with respect to the precedence constraints and objective function(s).

Based on the problem structure, ALBP can be classified into two groups as given in Figure 2.9. The first group is the classification according to the assembly line models, and the second group is the classification of Baybars (1986) (Gen, Cheng, & Lin, 2008).

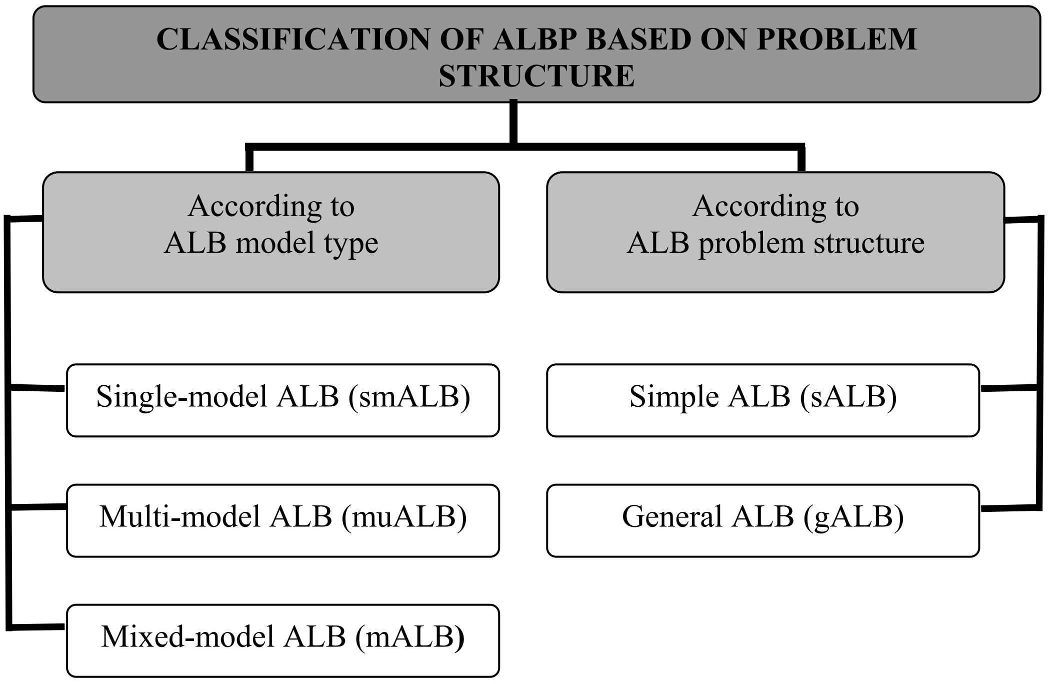

CLASSIFICATION OF ALBP BASED ON PROBLEM STRUCTURE
According to ALB model type
According to ALB problem structure
Single-model ALB (smALB)
Multi-model ALB (muALB)
Mixed-model ALB (mALB)
Simple ALB (sALB)
General ALB (gALB)

Figure 2.9 Classification of assembly line balancing problems based on problem structure

The classification according to the assembly line models has three kinds of ALBP. These are *Single Model assembly line balancing problem* (SMALBP), *Multi Model assembly line balancing problem* (MuMALBP), and *Mixed Model assembly line balancing problem* (MMALBP). SMALBP includes balancing of assembly lines producing only one product. MuMALBP includes balancing of assembly lines producing a family of product in batches. MMALBP includes balancing of assembly lines producing several models of a basic product in an arbitrarily inter-mixed sequence (Boysen et al., 2008).

According to the classification proposed by Baybars (1986) with respect to the problem structure, the problem can be grouped into two types: The original and simplest form of the problem is *simple assembly line balancing problem* (SALBP). When additional constraints are added to the model, the problem becomes the *general assembly line balancing problem* (GALBP).

If only one homogeneous product is continuously manufactured in large quantities on the line, the problem is SMALBP. In the literature, the deterministic SMALBP is called as *simple assembly line balancing problem* (SALBP) and specifies the following assumptions (Baybars, 1986):

1. All of the parameters relating to the line must be known with certainty.
2. A task cannot be divided between two or several stations.
3. Tasks cannot be treated in an arbitrary order due to the precedence constraints.
4. All the tasks of an assembly line must be processed.
5. All the stations are equipped with various resources, and can process any task.
6. The task process time is independent of the station on which it will be processed.
7. Any task can be made on any station.
8. The assembly line is serial, and contains neither feeding system, nor parallel subassembly lines.
9. The assembly system is to be designed for a unique model of a single product.

10. The cycle time is fixed, and the goal is to minimize the number of stations. Or, the number of stations is fixed, and the goal is to minimize the cycle time.

When the other restrictions or factors are introduced into the model, the problem becomes the *general assembly line balancing problem* (GALBP). Thus, GALBP is a generalization of SALBP and includes all of the problems that are not SALBP. Multi/mixed-model cases, zoning constraints, restrictions on balance delay, parallel stations, forms of positional restrictions, feeder or subassembly lines, parallel, U-shaped, robotic or two-sided lines, workcenters, stochastic or dependent processing times, cost functions, equipment selection are the factors of GALBP. Therefore, GALBP is more realistic (Becker & Scholl, 2006; Boysen et al., 2007, 2008).

Besides balancing a newly designed assembly line, an existing assembly line has to be re-balanced in a periodic way or after some changes in the production process or the production plan. Due to the long-term effect of balancing decisions, the strategic goals of the enterprise require the objective functions be carefully chosen.

Additionally, based on the objective function, ALBP have several versions (Kim, Kim, & Kim, 1996). These are with objectives to minimize the number of workstations (Type-1), to minimize cycle time (Type-2), to maximize workload smoothness (Type-3), to maximize work relatedness (Type-4), and the multiple-objective with the objective of Type-3 and Type-4 (Type-5). The most common type of ALBP is Type-E, with the objective of maximizing the line efficiency by simultaneously minimizing the cycle time and number of workstations. Another type of ALBP is the feasibility problem (Type-F); finding a feasible balance for a given number of stations and a given cycle time (Scholl, 1999).

Main constraints in ALBP are the *cycle time constraint* and *task precedence constraints*. Their explanations are given in Section 2.2.1. In addition to these constraints, some other constraints given below may restrict possible assignments of tasks to stations (Baybars, 1986; Scholl, 1999; Boysen et al., 2007):

Task zoning constraints: Some zoning constraints force and others forbid the assignment of different tasks to the same workstation, being called *positive* or *negative* zoning constraints, respectively. Positive zoning constraints are related with the use of common equipment or tooling. Some tasks may need the same equipment or may have similar processing conditions (temperature, moisture, etc.). Then, it is required to assign them to the same workstation. Negative zoning constraints are usually related with the technological issues. It may not be possible to perform some tasks in the same workstation because of safety reasons or etc.

Workstation related constraints: If some tasks need special equipment or material which is only available at a determined workstation, then these tasks are assigned to that workstation.

Position related constraints: These constraints group tasks according to the position in which they are performed, especially when the workpieces of large and heavy products have a fixed position and cannot be turned.

Operator related constraints: Some tasks require different levels of skill depending on their complexity. A sufficiently qualified operator is assigned to a determined task. It is better to combine more monotonous tasks and more variable tasks in the same workstation in order to induce higher levels of job satisfaction and motivation, from the ergonomic point of view.

2.4 Solution Methods for the Assembly Line Balancing Problem

The idea of balancing was first introduced by Bryton (1954) in his graduate thesis (Kilbridge & Wester, 1962). The first analytical statement of ALBP was formulated by Helgeson, Salveson, & Smith (1954), while the first published study of ALBP modeled mathematically with a linear programming solution belonged to Salveson (1955) (Ghosh & Gagnon, 1989). Since then, many solution procedures were developed to solve ALBP (Agpak & Gokcen, 2005). Generally branch and bound

(B&B) procedures (Amen, 2006; Peeters & Degraeve, 2006) and dynamic programming (DP) approaches were used.

In the last decade, a large variety of heuristic approaches were in the focus of the researchers (Gamberini, Grassi, & Rimini, 2006). These were constructive procedures based on priority rules or enumeration techniques (Dimitriadis, 2006) and improvement procedures using metaheuristics like tabu search (Lapierre, Ruiz, & Soriano, 2006), ant colony optimization (Bautista & Pereira, 2002, 2007; Mcmullen & Tarasewich, 2003, 2006), simulated annealing (Baykasoglu, 2006; Kara, Ozcan, & Peker, 2007a, 2007b) and genetic algorithms (Baykasoglu & Ozbakir, 2007; Haq, Jayaprakash, & Rengarajan, 2006; Levitin, Rubinovitz, & Shnits, 2006; Simaria & Vilarinho, 2004; Tseng & Tang, 2006; Wong, Mok, & Leung, 2006; Yu, Yin, & Chen, 2006).

Baybars (1986) described and commented on a number of optimum seeking methods for SALBP. The heuristic procedures for ALBP were critically examined and summarized in details by Ghosh & Gagnon (1989) and Erel & Sarin (1998) for SALBP and GALBP. A survey of existing solution methods for different extensions of SALBP and GALBP was given by Rekiek et al. (2002).

Up-to-date analysis of the bibliography and available state of the art procedures for SALBP family of problems were given by Scholl & Becker (2006) and for GALBP by Becker & Scholl (2006). Boysen et al. (2007) classified the ALBP literature with a scheme including the extension of the problem and solution method.

According to the classification of studies surveyed by Scholl & Becker (2006) and review of existing methods by Rekiek & Delchambre (2006), Figure 2.10 gives a classification scheme for solution approaches of ALBPs.

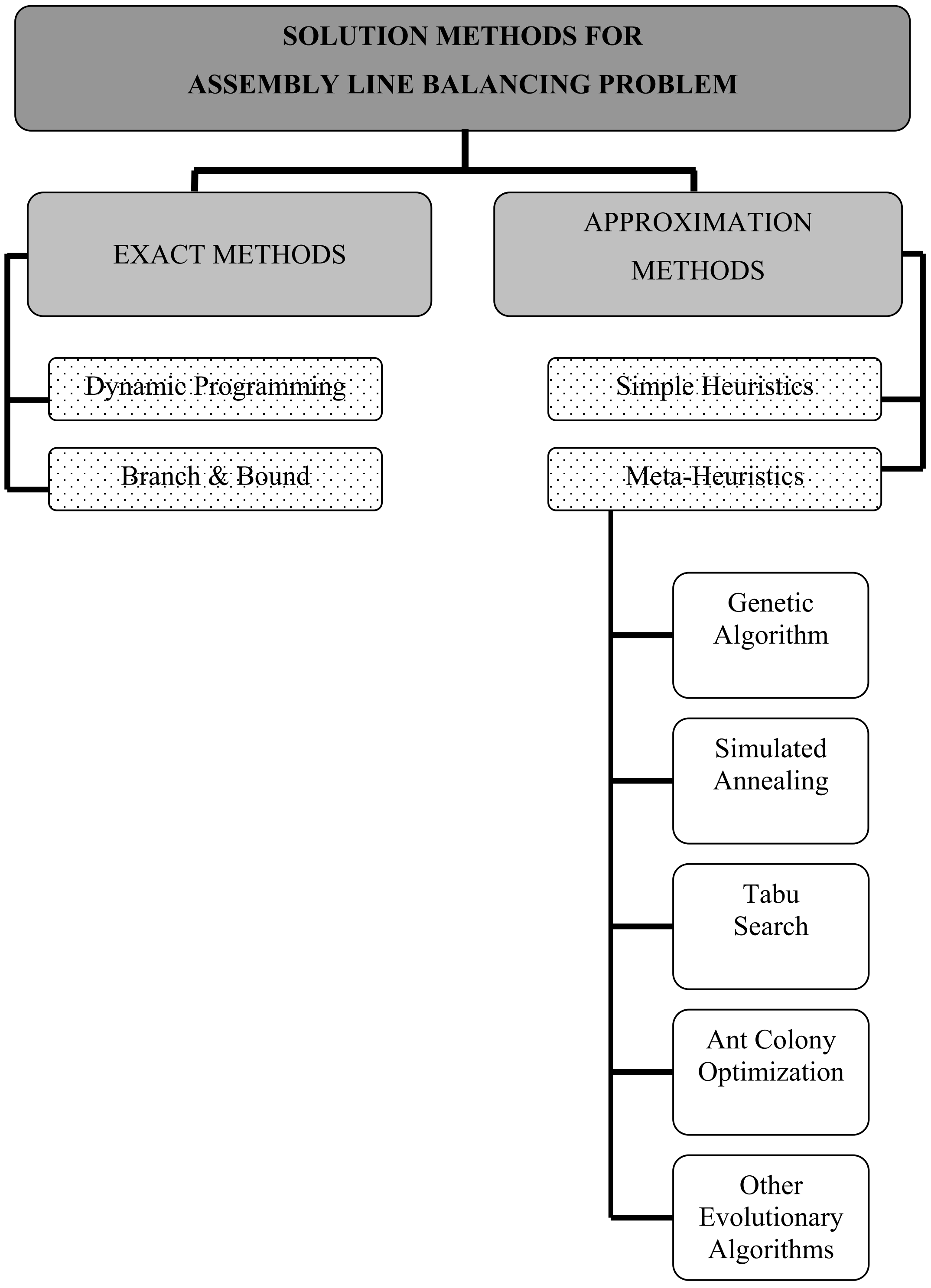


Figure 2.10 Classification of solution approaches for ALBP

2.4.1 Optimum Seeking Methods

Several approaches for determining lower bounds on the objectives of ALBPs are proposed in the literature. The lower bounds are obtained by solving problems which are derived from the considered problem by omitting or relaxing constraints. Most of these techniques fall into two categories, i.e., dynamic programming and branch and bound methods. Baybars (1986) described and commented on a number of optimum seeking methods for SALBP. A survey on exact methods for the ALBP can also be found in Scholl (1999).

2.4.1.1 Dynamic Programming

Dynamic programming (DP) is a very powerful algorithmic paradigm to tackle multistage decision processes. DP is applied mostly to combinatorial optimization problems (Rekiek & Delchambre, 2006). Any given problem is solved by identifying a collection of sub-problems and tackling them sequentially one by one, smallest first, using the answers to small problems to help figure out larger ones, until the initial problem is solved by the aggregation of the sub-problem solutions. By dynamic programming, the problem can be divided into stages with a decision required at each stage. Each stage has a number of states associated with it. The states describe all possible conditions of the process in the current decision stage, which corresponds to every feasible partial solution. The decision at one stage transforms one state into a state in the next stage. The problem is solved by finding the optimal policy from an initial state to a final state in a chain (Bautista & Pereira, 2009). The studies given in the following are linked to DP procedures.

The first published study of ALBP formulated mathematically with a linear programming (LP) solution belonged to Salveson (1955). Salveson's LP model to solve SALBP included all possible combinations of station assignments. Later, Bowman (1960) modified the formulation. Bowman (1960) was the first to provide "nondivisibility" constraint, by changing the LP formulation to zero-one integer programming (IP) (Baybars, 1986). Other formulations have been proposed by many researchers, e.g. White (1961), Klein (1963), Thangavelu & Shetty (1971), Patterson & Albracht (1975), Talbot & Patterson (1984), Ugurdag, Rachamadugu, & Papachristou (1997), and Corominas (1999).

MMALBP with an IP model was first solved by Robert & Villa (1970). In the model proposed, the objective was the minimization of the total idle time. The authors stated that the formulation is of more theoretical than practical interest due to the excessive number of constraints and variables. Later, Gokcen & Erel (1997) proposed a zero-one IP model utilizing a precedence diagram which combines different models of the problem. The performance of this model was superior to the model of Robert & Villa (1970).

Agpak & Gokcen (2005) developed a zero-one IP model to solve resource constrained SMALBP Type-1 with the objective of minimizing the number of workstations and the number of resources used. Gokcen, Agpak, & Benzer (2006) proposed a zero-one IP model to solve SMALBP Type-1 with parallel lines. Hop (2006) developed a fuzzy zero-one IP model to solve MMALBP Type-1 with fuzzy processing times. Peeters & Degraeve (2006) presented a Dantzig-Wolfe type reformulation of SALBP Type-1, the LP-relaxation which was solved using column generation combined with subgradient optimization. Urban & Chiang (2006) proposed an IP model, using a piecewise approximation for the chance constraints, to solve U-shaped SMALBP Type-1 with stochastic processing times. Corominas, Pastor, & Plans (2008) presented a zero-one IP model to solve the rebalancing of SMALBP with skilled and unskilled workers with the objective of minimizing the number of unskilled temporary workers.

Toksari et al. (2010) developed a mixed nonlinear IP (MNIP) model SMALBP Type-1 with deterioration tasks and learning effects. “Learning effect” is a phenomenon for improving continuously as a result of repeating the same or similar activities (Mosheiov, 2001). The processing time of a job is shorter if it is done again later, because the processing time is dependent on learning of workers for repeating tasks. Modeling the effect of task deterioration was introduced by Mosheiov (1991). Deterioration tasks are the tasks whose processing times are increasing functions of their starting times.

The first DP method was developed by Jackson (1956) to solve SALBP using a tree notion. The solution process was subdivided in stages corresponding to stations. States were given by the feasible subsets of tasks already assigned at a given stage. The algorithm started by generating all feasible assignments to the first station. Then, this generated all feasible assignments to the next station, given the first station assignments. The process was repeated, each time adding one station. The optimal solution was searched for stage-by stage in a forward recursion (Baybars, 1986). A number of researchers have employed DP methods, e.g. Held & Karp (1961), Held, Karp, & Shareshian (1963), Van Assche & Herroelen (1979), Johnson (1981), Bard (1989), and Carraway (1989).

Gutjahr & Nemhauser (1964) transformed SALBP Type-1 to an equivalent shortest path problem. The states were represented by nodes and the station loads by arcs which were weighted with the corresponding station idle times. Each path corresponded to a feasible solution and each shortest path to an optimal solution of SALBP Type-1. Later, Gokcen, Agpak, Gencer, & Kizilkaya (2005) presented a shortest route formulation of U-shaped SMALBP Type-1 based on the study of Gutjahr & Nemhauser (1964).

Miltenburg & Wijngaard (1994) introduced and modeled the U-shaped ALBP and proposed a DP procedure to identify the optimal solution for problems with small size. Guerriero & Miltenburg (2002) presented a DP approach to solve U-shaped SMALBP Type-1 with stochastic processing times. Bautista & Pereira (2009) proposed a new DP based heuristic, called Bounded DP, which mixed a set of heuristic rules within a DP to solve SALBP Type-1.

Goal programming (GP) is an important technique for decision-makers to consider simultaneously conflicting objectives in finding a set of acceptable solutions. GP models were used by researchers dealing with more than one goal in order to utilize IP formulations of ALBPs.

Gokcen & Agpak (2006) were the first to solve U-shaped SMALBP using a GP model with a preemptive approach as a multi-criteria decision making approach. Kara & Tekin (2009) presented a mixed IP formulation to solve U-shaped MMALBP Type-1. Kara, Paksoy, & Chang (2009) presented binary fuzzy GP approach and employed IP method to solve U-shaped SMALBP with the objectives of minimizing the number of workstations and the cycle time at the same time in a fuzzy environment.

Ozcan & Toklu (2009) presented a new MIP model to solve two-sided SMALBP Type-1 with an objective of minimizing the number of mated-stations. The authors also developed a mixed-integer GP model (MIGP) and a fuzzy mixed-integer GP model (FMIGP). The proposed goal programming models were the first multiple-criteria decision-making approaches to solve two-sided SMALBP with multiple objectives. Choi (2009) presented a new zero-one IP model and an algorithm based on GP to solve MMALBP that concerned both processing time and physical workload at the same time as total workload.

2.4.1.2 Branch & Bound Algorithm

Branch and bound (B&B) is a general algorithm for finding optimal solutions of various optimization problems, especially in discrete and combinatorial optimization. It consists of a systematic enumeration of all candidate solutions, where large subsets of fruitless candidates are discarded, by using upper and lower estimated bounds of the quantity being optimized. The B&B algorithm consists of two main components: the branching and the bounding. To reduce the solution effort, dominance and reduction rules are additionally used. The initial solution of the B&B algorithm is developed into several sub-problems, which is called branching. A multi-level enumeration is constructed by continuously developing such sub-problems. The sub-problems for which the optimal solution is already known and for which there is no need to be branched are called as leaf nodes. A leaf node is also used for nodes which are excluded from further consideration because they cannot lead to an optimal solution. Branch is a path from the root node to any other node of the tree. B&B procedures differ with respect to search strategy, a sequence in which the nodes of the enumeration tree are generated and branched: Depth-first-search and a minimal-lower-bound strategy. Bounding is applied to reduce the size of the enumeration trees. This is achieved by computing lower bounds at least necessary for a feasible solution in each node. If the global lower bound is found, then an optimal solution is found (Rekiek & Delchambre, 2006).

FABLE by Johnson (1988) and EUREKA by Hoffmann (1992) were the most effective key developments of B&B methods introduced to solve SALBP Type-1. Later, Klein & Scholl (1996) combined EUREKA and FABLE, and developed B&B methods called SALOME-1 to solve SALBP Type-1 and SALOME-2 to solve SALBP Type-2. The authors proposed the local lower bound method which was a new enumeration technique and pointed out the similarities and differences between proposed and existing methods, such as FABLE and EUREKA.

Scholl & Klein (1999) compared the most effective branch and bound procedures for SALPB-1, such as Johnson’s FABLE, Nourie & Venta's OptPack, Hoffmann's EUREKA, and Scholl & Klein's SALOME-1. In this computational comparison, the authors used totally 268 problem instances from Talbot’s data set, Hoffmann’s data set, and Scholl’s data set. In Hoffman’s data set OptPack was found to be the most effective. SALOME was the most effective procedure in Talbot’s data set and in Scholl’s data set, so that it was determined as a most effective B&B procedure in the study. However other procedures had got some superior properties. OptPack was very effective in reducing the size of the enumeration tree. Therefore, Scholl & Klein (1999) extended SALOME by adding dynamic renumbering and some dominance rules and called the new version of SALOME as SAL-All. SAL-All outperformed previous version of SALOME for all data sets.

Sprecher (1999) developed a B&B method to solve SALBP Type-1, called adapted general sequencing algorithm (AGSA), which was based on the precedence guided enumeration scheme introduced for dealing with resource-constrained project scheduling problems. Sprecher (1999) reformulated this problem as a resource constrained project scheduling problem by reflecting cycle time as a single renewable resource whose availability varied with time.

Bukchin & Tzur (2000) presented an optimum seeking method and a heuristic to solve SMALBP with equipment selection. They developed a B&B algorithm and also a B&B based heuristic to solve large problems. Later, Bukchin & Rubinovitz (2003) adapted this B&B optimal algorithm which was developed for the equipment selection problem by Bukchin & Tzur (2000) to solve SMALBP with station paralleling.

Amen (2006) used B&B techniques with LP-relaxation and implicit enumeration technique to solve cost-oriented ALBP. Bukchin & Rubinowitch (2006) developed an optimal solution procedure based on a backtracking B&B method to solve MMALBP allowing a common task to be assigned to different stations for different models with the objectives of minimizing the number of the workstations (Type-1) and task duplication cost. Peeters & Degraeve (2006) developed a B&B algorithm to solve SALBP Type-1. Liu, Ng, & Ong (2008) presented new B&B algorithms to solve SALBP Type-1, a constructive algorithm and two destructive algorithms.

Miralles, Garcia-Sabater, Andres, & Cardos (2008) introduced a new kind of ALBP called Assembly Line Worker Assignment and Balancing Problem (ALWABP) Type-2 and presented a basic B&B approach with three possible search strategies and different parameters to solve this new problem. Wu, Jin, Bao, & Hu (2008) proposed B&B algorithms for two-sided ALBP and carried out some experiments.

Ege, Azizoglu, & Ozdemirel (2009) proposed two B&B algorithms, one for optimal solutions and one for near optimal solutions to solve GALBP with station paralleling. The objective was to minimize the sum of station opening and equipment costs. Scholl & Boysen (2009) used ABSALOM, a method based on an extension of SALOME (Klein & Scholl, 1996), to solve SMALBP Type-1 with parallel assembly lines considered by Gokcen et al. (2006). Later, Scholl, Fliedner, & Boysen (2010) used ABSALOM to solve SMALBP Type-1 with assignment restrictions.

2.4.2 Approximation Methods

Due to the problem size limitation of the exact methods, approximation procedures are required to solve more realistic problems, i.e., medium and big scaled problems. A variety of simple heuristics and meta-heuristics have been proposed in the literature to solve ALBP. In this section, some of the well-known will be considered.

2.4.2.1 Heuristic Methods

Many heuristics proposed in the literature use different criteria (Talbot, Patterson, & Gehrlein, 1986). Many proposed heuristics are a combination of these methods. The most effective ones are: RPWT (Helgeson & Birnie, 1961), Killbridge & Wester’s (1961), Hoffmann’s precedence matrix procedure (Hoffmann, 1963), COMSOAL (Arcus, 1966), Moodie & Young's (1965), and Lapierre & Ruiz’s (1999) improved COMSOAL heuristics.

One of the first proposed heuristic was the ranked positional weight technique (RPWT) (Helgeson & Birnie, 1961). RPWT works by assigning the tasks which have long chains of succeeding tasks. The length of the chain can be measured either by the number of successors or the sum of the task times of the successors. The sum of the task process time and the process times of the successors is defined as the positional weight of the task. The tasks are then listed in descending order of weight, and an attempt is made to assign them in that order to the assembly stations, starting with the first station and proceeding, station by station, along the line.

Killbridge & Wester (1961) proposed a method, which groups tasks into columns in the precedence diagram where tasks are placed as far left as possible without violating precedence relations.

Hoffmann (1963) proposed a heuristic based on a method for generating permutations using a precedence matrix. In the procedure, from the available tasks, a subset is selected such that the current station is loaded as much as possible. The procedure is repeated until all tasks are assigned. The procedure tends to concentrate tasks either at the first few stations or the last few stations depending on whether a forward or reverse problem is solved.

Moodie & Young (1965) presented a modified formulation of the ALB problem that includes task time variability. The developed heuristic places tasks into workstations according to the longest task processing time. A task cannot be placed into a station unless all of its immediate predecessors have been already assigned.

Arcus (1966) developed a heuristic known as COMSOAL, essentially a computer simulation technique that randomly generates a number of feasible solutions and adopts the best of these solutions by using ‘priority-based’ heuristics. In COMSOAL, for each task in the precedence graph, the numbers of immediate predecessors of all tasks are enumerated in a list. Then the tasks which have no immediate predecessors in this list are determined and enumerated in a second list. A task is selected randomly and removed from this second list. The second list is updated by moving all the tasks which are numerated at the bottom of the selected task in the list to an upper position. The selected task is removed from the precedence graph and the first list is updated. These steps are repeated until all the tasks are assigned according to the cycle time constraint.

Lapierre & Ruiz (1999) programmed the COMSOAL algorithm (Arcus, 1966) on the software package Microsoft ACCESS97 with a modification to deal with constraints such as the position (rear, front, centre, etc.) and the level (high and low) of tasks. Thus, the method aims to avoid grouping tasks having different levels on the same station.

Fonseca, Guest, Elam, & Karr (2005) developed fuzzy versions of RPWT and COMSOAL methods to solve SMALBP Type-1 with a fuzzy representation of the time variables by triangular fuzzy numbers. Gokcen et al. (2006) developed two new procedures based on the COMSOAL algorithm of Arcus (1966) to solve SMALBP Type-1 with parallel lines. Jiao, Kumar, & Martin (2006) proposed the design and implementation of a web-based advisor composed of a schedule based on various heuristic algorithms such as RPWT, Killbridge & Wester's method, and COMSOAL embedded in its library to solve SALBP Type-1 and Type-2. Kara & Tekin (2009) developed a new heuristic procedure based on the COMSOAL algorithm of Arcus (1966) to solve U-shaped MMALBP Type-1.

Toksari et al. (2008) used the shortest task rule to solve SALBP and U-shaped SMALBP Type-1 with learning effects. Later, Toksari et al. (2010) adapted the COMSOAL algorithm of Arcus (1966) to solve large scale SMALBP Type-1 with deterioration tasks and learning effects.

Boctor (1995) introduced a four-rule heuristic to solve SALBP Type-1. Bukchin et al. (2002) presented a mathematical model and a new three-stage heuristic, in which one of the stages was based on B&B, to solve MMALBP Type-1 in a make-to-order environment.

Jin & Wu (2002) developed a new heuristic algorithm called "variance algorithm" to solve MMALBP with an objective of minimizing the variance in the rate of resources used by the units.

Zhao, Ohno, & Lau (2004) proposed a one-pass heuristic, based on the lower bound of the total overload time, to solve paced MMALBP with an objective of minimizing the total overload time.

Liu, Ong, & Huang (2005) proposed a bi-directional heuristic to solve SMALBP Type-2 with stochastic processing times. Hoffmann's procedure (Hoffmann, 1963) was applied to guarantee the best task assignment. The tasks were assigned to workstations from two directions of the assembly line alternatively. The proposed method was superior to Moodie & Young's (1965) method.

Chiang & Urban (2006) presented a hybrid heuristic composed of an initial feasible solution module and a solution improvement module to solve U-shaped SMALBP Type-1 with stochastic processing times. The first module consisted of two approaches as "First-Fit" and "Priority Based". The second module consisted of approaches as "Least Number of Tasks" and "Least Task Time".

Dimitriadis (2006) developed a heuristic based on an enumeration method, Hoffmann's precedence matrix procedure (Hoffmann, 1963), to solve paced ALBP with multi-manned workstations to achieve higher space utilization while the total effectiveness still remained optimized.

Battini, Faccio, Ferrari, Persona, & Sgarbossa (2007) introduced a new heuristic procedure to solve unpaced MMALBP Type-2 with multi-turns circular transfer systems, such as a multi-station rotating table.

Kilincci & Bayhan (2006) developed a Petri net based heuristic to solve SALBP Type-1. Later, Kilincci & Bayhan (2008) developed a heuristic based on the P-invariants of Petri nets to solve SALBP Type-1. Kilincci (2010) developed a two-stage heuristic adapted from a Petri net approach of Kilincci & Bayhan (2006) to solve SALBP Type-2.

Cevikcan, Durmusoglu, & Unal (2009) presented a team-oriented mathematical programming model for creating assembly teams (physical stations) in MMALBP. The authors developed a scheduling based heuristic algorithm for this design methodology including horizontal and vertical balancing and model sequencing for mixed-model assembly lines.

2.4.2.2 Meta-Heuristics

Meta-heuristics are general search principles organized in a general search strategy used to solve combinatorial optimization problems (Pirlot, 1996). Meta-heuristics start with an initial solution obtained with a heuristic and improve it, so they are the natural extension of priority-based heuristics. They are able to search large regions of the solution's space without being trapped in local optima, a major disadvantage of pure local search algorithms. They have provided effective approximate solutions for difficult NP-hard combinatorial optimization problems. In the last decade, the focus of researchers has been on improvement procedures using meta-heuristics like Tabu Search (TS), Simulated Annealing (SA), Genetic Algorithm (GA), and Ant Colony Optimization (ACO) to solve ALBPs. This section focuses on literature review of their applications to ALBPs.

Tabu Search (TS) is a generalized local search procedure proposed by Glover (1986) to guide other methods to escape the trap of local optimum. TS starts from an initial solution and iteratively moves to a neighbor solution which either improves on the previous solution or not. It uses problem-specific operators to explore a search space and memory (which is called the tabu list) to keep track of parts already visited. Some applications of TS for solving ALBP can be found in Peterson (1993), Scholl & Voss (1996), Chiang (1998), Pastor, Andris, Duran, & Pirez (2002), Lapierre et al. (2006), and Suwannarongsri & Puangdownreong (2008).

A TS algorithm was used to solve ALBP firstly by Peterson (1993). An initial solution was adjusted according to tabu to improve the solution to a near-optimum condition with this method. To solve SALBP Type-1 and Type-2, Scholl & Voss (1996) presented basic TS algorithms. Chiang (1998) proposed another TS approach to solve SALBP Type-1. Although both of the methods were rather simple versions of TS, good results were obtained on classical data sets. Pastor et al. (2002) proposed a TS algorithm for an industrial multi-product and multi-objective ALBP. Lapierre et al. (2006) presented a new TS algorithm to solve SALBP Type-1 and discussed its differences with respect to those in the literature. The differences of the proposed SA were the use of two different complementary neighborhoods redefinition of the solution space and the objective function in order to allow the algorithm to visit infeasible solutions.

A recent application of TS can be found in Suwannarongsri & Puangdownreong (2008). The authors proposed a TS algorithm hybridized with the partial random permutation (PRP) technique to solve SALBP with the objective of minimizing workload variance. The TS algorithm was used to address the number of tasks assigned for each workstation, while the PRP technique was used to arrange the sequence of tasks.

Simulated Annealing (SA) was introduced by Kirkpatrick, Gelatt, & Vecchi (1983) to solve NP-hard combinatorial optimization problems, by using the analogy with the simulation of the physical annealing of solids, in order to optimize the value of an objective function. The SA algorithm starts with a non-optimal initial solution and tries to improve it according to an annealing schedule that controls temperature. In each iteration, the difference between current position and the next possible position is calculated. If there is an improvement, the change is automatically accepted. If not, the change may still be accepted according to a probability, which decreases exponentially with the badness of the move. Some applications of SA for solving ALBP can be found in Suresh & Sahu (1994), McMullen & Frazer (1998), Erel, Sabuncuoglu, & Aksu (2001), Vilarinho & Simaria (2002), Baykasoglu (2006), and Kara et al. (2007a, 2007b).

Suresh & Sahu (1994) developed a SA algorithm to solve SMALBP with stochastic processing times. To solve multi-objective MMALBP with parallel stations, McMullen & Frazer (1998) presented a SA algorithm for stochastic processing times. Erel et al. (2001) developed a heuristic based on SA to solve U-shaped ALBP. Vilarinho & Simaria (2002) developed a two-stage SA algorithm to solve MMALBP with additional restrictions and parallel stations.

Mendes, Ramos, Simaria, & Vilarinho (2005) proposed a heuristic procedure combined of a version of RPWT and a SA algorithm to solve MMALBP Type 1. At first, the version of RPWT computed the initial solution, and then the SA algorithm tried to improve the solution.

Baykasoglu (2006) presented a multi-rule multi-objective SA algorithm to solve SALBP and U-shaped SMALBP multiple objectives with Type 1 and Type-3.

Recent applications of SA can be found in Kara et al. (2007a, 2007b). Kara et al. (2007a) was the first to deal with simultaneously balancing and sequencing problems of MMALBP Type-1 by using the SA method. Kara et al. (2007b) proposed a SA algorithm to solve simultaneously balancing and sequencing problems of MMALBP with multiple objectives of minimizing part usage rate, minimizing setup cost, and minimizing deviations of workload across workstations.

Ant Colony Optimization (ACO) presented by Dorigo, Maniezzo, & Colorni (1996) and Dorigo, Di Caro, & Gambardella (1999) is a population-based procedure inspired on the behavior of real ant colonies. Ants are known for being able to find the shortest path between their nest and a food source, without making use of visual cues; only by following pheromone trails released by other ants. It is the colony as a whole that coordinates the activities without a direct communication between individual ants, as an isolated ant basically moves at random. ACO exploits a similar mechanism for solving optimization problems. In ACO, a number of artificial ants build solutions to an optimization problem and exchange information on the quality of these solutions via a communication scheme that is reminiscent of the one adopted by real ants. Some implementations of ACO to solve ALBP can be found in Bautista & Pereira (2002, 2007), McMullen & Tarasewich (2003, 2006), Vilarinho & Simaria (2006), Boysen & Fliedner (2008), Baykasoglu & Dereli (2008, 2009), Sabuncuoglu, Erel, & Alp (2009), and Simaria & Vilarinho (2009).

Bautista & Pereira (2002) presented an ACO algorithm to solve SALBP-2. McMullen & Tarasewich (2003) proposed an ACO algorithm to solve MMALBP with parallel stations and stochastic task processing times. Later McMullen & Tarasewich (2006) presented an ACO technique to solve MMALBP with stochastic task processing times and multiple objectives via a composite function. This study was an extension of their previous research where only single-objective functions were addressed.

Blum, Bautista, & Pereira (2006) proposed a Beam-ACO algorithm to solve the time and space constrained SMALBP Type-1 with the objective of minimizing the number of necessary work stations. This problem was denoted as TSALBP-1 in the literature. The proposed Beam-ACO approach was a state-of-the-art meta-heuristic that resulted from hybridizing ACO with beam search.

Vilarinho & Simaria (2006) presented an ACO algorithm to solve MMALBP for two objectives of Type-1 and Type-3 with zoning restrictions and parallel workstations. Bautista & Pereira (2007) presented an ACO algorithm to solve the time and space constrained ALBP with various objectives. Boysen & Fliedner (2008) proposed a two-stage general procedure (AVALANCHE) to solve several extensions of SALBP and GALBP with constraints such as parallel workstations and tasks, cost synergies, processing alternatives, zoning restrictions, stochastic processing times or U-shaped assembly lines. In the first stage, the ACO algorithm was used for sequence generation. Then, the task assignment was carried out by well-known mathematical tools such as IP.

Baykasoglu & Dereli (2008) proposed an ACO based heuristic to solve two-sided ALB problems with zoning constraints (2sALBz). This paper was one of the first attempts to show how an ant colony heuristic (ACH) can be applied to solve 2sALBz problems. Later, Baykasoglu & Dereli (2009) proposed an ACO algorithm integrated with COMSOAL method and RPWT to solve SALBP and U-shaped SMALBP Type-1. Sabuncuoglu et al. (2009) proposed an ACO algorithm to solve U-shaped SMALBP Type-1. Simaria & Vilarinho (2009) proposed an ACO algorithm to solve two-sided MMALBP Type-1 with additional goals. In the proposed procedure, two ants worked simultaneously, one at each side of the line.

Genetic Algorithms (GA) (Holland, 1975) are an iterative search method, based on the biological process of natural selection and genetic inheritance, which maintain a population of a number of candidate members over many simulated generations. Falkenauer & Delchambre (1992) were the first to solve ALBP with GAs. Some of the applications of GAs for solving ALBP can be found in Leu, Matheson, & Rees (1994), Falkenauer (1997), Rekiek, De Lit, Pellichero, Falkenauer, & Delchambre (1999), Goncalves & De Almedia (2002), Stockton, Quinn, & Khalil (2004a, 2004b), Brown & Sumichrast (2005), and Rekiek & Delchambre (2006). A review of the GA applications for ALBPs will be given in Chapter 4.

An iterative procedure named "balance for order", based on a modified GA, was proposed by Rekiek, De Lit, & Delchambre (2000) to solve problems of model sequencing and line balancing in a mixed-model assembly line simultaneously. The proposed algorithm was tested on randomly generated instances. The number of operations varied from 50 to 500 and the number of models varied from 1 to 50. The results of the experiments showed that the optimum solutions as the number of workstations and makespan depended on both desired cycle time and maximum peak time.

Symbiotic evolutionary algorithm (SEA), a special kind of GA, is a stochastic search algorithm that imitates the biological co-evolution process through symbiotic interaction (Potter, 1997). SEA maintains two or more populations (species) that represent sub-problems. Then, an individual of a population becomes a partial solution to the entire problem. Complete solution of the problem is constructed by combining all the partial solutions of each population.

Kim et al. (2000a) presented a new method, called SEA, using a co-evolutionary algorithm that could solve line balancing and model sequencing problems of MMALBP at the same time. The objective was minimizing utility work, which was defined as the amount of uncompleted works within the given length of a workstation. The balancing problem and sequencing problem were defined as populations. Generation, crossover and mutation operations, genetic representations, and adaptations of all of them to the line balancing problem were explained in detail by representing the proposed algorithm. Thomopoulos' 19-task and Arcus' 111-task problems and a real life problem with 61-task were used to perform the algorithm. The experimental results showed that the proposed algorithm was superior to existing approaches. Later, Kim et al. (2000b) proposed SEA to deal with the integration of balancing and sequencing of U-shaped MMALBP simultaneously. Totally 21 problems were solved. The proposed co-evolutionary algorithm was compared with some methods and the hierarchical approach which solved a sequencing problem after the solution of the line balancing problem. Thomopoulos' 19-task problem, Arcus' 111-task problem, and a real life problem with 61 tasks were used as test-bed problems. The results showed that the proposed algorithm outperformed the existing methods. Also it improved the results according to the hierarchical approach as from %28.20 to %73.20.

Endosymbiotic evolutionary algorithm (EEA), an extension of SEA, is an algorithm in which an evolutionary strategy imitating the endosymbiotic process is embedded in an existing SEA. The theory of endosymbiotic evolution was first proposed by Margulis (1980) and the basic idea was based on the algorithm by Kim et al. (2001).

Kim et al. (2006) proposed EEA to solve simultaneously line balancing and sequencing problems of U-shaped MMALBP Type-3. The proposed algorithm constructed and maintained a balancing population and a sequencing population. The individuals of each population became a partial solution of the problem. The algorithm maintained another population consisted of individuals formed by the integration of the two types of individuals. Then, they became the entire solution representing a combination of work assignment and model sequence.

Differential evolution algorithm (DEA) is an evolutionary algorithm introduced by Storn & Price (1997) for global optimization over continuous spaces.

Nearchou (2007) proposed a new heuristic based on DEA to solve SALBP Type-2. Two versions of the proposed FEA were implemented, one using random-keys encoding scheme and the other using priority-based. Their performances were compared with three types of GAs by testing two data sets from the literature including 17 problems with tasks varying from 29 to 297. The results of the proposed DEA were quite promising. Later, Nearchou (2008) proposed a multi-objective version of DEA to solve bi-criteria SALBP Type-2. The secondary objectives were to minimize balance delay time and workload smoothness index. The author compared three versions of the proposed DEA against two representative multi objective GAs, one proposed by Kim et al. (1996) and the other proposed by Murata, Ishibuchi, & Tanaka (1996). The version which used adaptive weights estimated in the objective function was superior to the others.

2.5 Chapter Summary

In this chapter, the terminology and characteristics of assembly lines, the assembly line balancing problem and various types of this problem with the solution methods were presented.

The literature review shows that there are many algorithmic developments as exact and heuristic procedures to solve mainly SALBP, because it is a benchmark problem with a large number of data sets with known optimal solutions. Due to the need to solve more realistic line balancing problems, recent studies evolve towards solving GALBPs with different extensions. There is a growing interest in the use of meta-heuristics to solve complex real world ALBPs. Many studies showed that meta-heuristics are able to solve SALBP and GALBP with a high performance and flexibility.

In this study, as a meta-heuristic approach, a GA will be employed to solve a novel GALBP. The following chapter will present detailed information about GAs.

CHAPTER THREE

GENETIC ALGORITHMS

3.1 Introduction

Since this study involves the application of GAs to solve CCALBP, this chapter will focus on the description of the main characteristics of GAs. The chapter is organized as follows. In Section 3.2, an introduction of the terminology for GAs is presented and the procedure of GAs is given. Finally, in Section 3.3, the context of this chapter is summarized.

3.2 Genetic Algorithms

Solving the combinatorial optimization problems by exact methods such as DP or B&B causes storage requirements and exponential growth in computation time. Solving by heuristics such as neighborhood search causes being trapped at locally optimal solutions. To avoid all, meta-heuristics have been developed (Pirlot, 1996).

Since the 1960s there has been an increasing interest in imitating living beings to develop powerful algorithms to solve difficult optimization problems. Recently, the term *evolutionary computation* is referred to such techniques. Many attempts have been made to understand the adaptive processes of natural systems. These methods, commonly called *meta-heuristics*, are general search principles organized in a general search strategy used to solve combinatorial optimization problems (Pirlot, 1996). They are high level strategies for exploring search spaces by using different methods (Blum & Roli, 2003). Figure 3.1 gives the basic chronology of well-known meta-heuristics including GA, SA, TS, ACO, particle swarm optimization, and differential evolution.

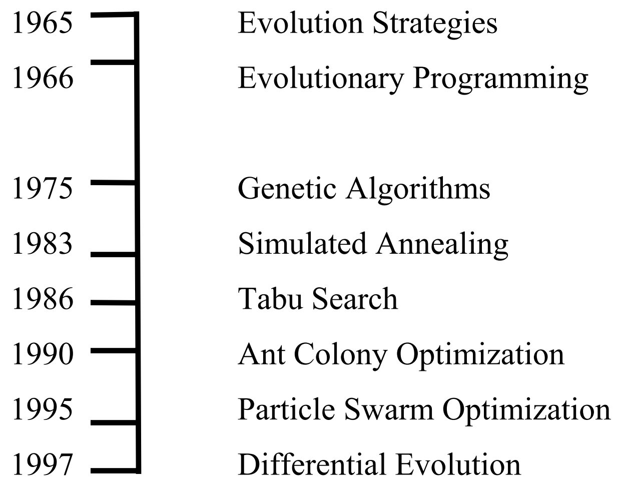


Figure 3.1 Chronology of meta-heuristics

The first work on GA was introduced by John Holland, from the University of Michigan at the beginning of 1960s. The first achievement was the publication of “Adaptation in Natural and Artificial System” in 1975 (Holland, 1975). Holland (1975) attempted to explain the adaptive processes of natural systems and to design an artificial system based upon these natural systems. GA as a search and optimization routine was popularized by Goldberg’s 1989 publication “Genetic Algorithms in Search, Optimization, and Machine Learning” (Goldberg, 1989).

GA is a stochastic search method; randomness is an essential role in GAs. GA simulates the natural process, and randomly searches the heuristic solution in the solution space, based on the mechanism of natural selection and natural genetics (Goldberg, 1989). Most stochastic search methods operate on a single solution to the problem, but GA operates on a population of solutions.

GAs have been applied for solving various combinatorial optimization problems (COPs) in the literature and have become increasingly popular among approximation techniques for finding optimal or near optimal solutions in a reasonable time to COPs (Dowsland, 1996) (Reeves, 1997).

GA differs from conventional optimization techniques in several ways (Goldberg, 1989):

1. GAs work with coded versions of the problem parameters and not with the parameters themselves, i.e., GA works with the coding of solution set rather than the solution itself.
2. Almost all conventional optimization methods search from a single point but GAs always search from a whole population of points, i.e., GA uses population of solutions, not a single solution for searching. This improves the chance of reaching the global optimum and also helps in avoiding a local optimum.
3. GA uses fitness function for evaluation, not derivatives. Therefore, GAs can be applied to any kind of optimization problem by identifying and specifying a meaningful decoding function.
4. GAs use probabilistic transition rules rather than deterministic rules.

Beside its advantages, there are some difficulties in adjustment of the GA control parameters (population size, crossover probability, and mutation probability), for specification of the termination condition, and for encoding problems into fixed-length chromosomes. There are limited available commercial software products to solve various problems.

3.2.1 Terminology for GAs

To understand the procedure of a general GA, there are some basic components to learn. In this section, general terminology of GA is given.

3.2.1.1 Representation

GAs do not operate directly on the solution space. The solutions are coded in the form of symbolic strings called *chromosomes*. An encoding is selected in a way that each solution in the search space is represented by one chromosome.

A chromosome is subdivided into genes. A *gene* is the GA's representation of a single factor for a control factor. Each factor in the solution set corresponds to a gene in the chromosome. Genes are the basic "instructions" for building a GA. A chromosome is a sequence of genes. Genes may describe a possible solution to a problem, without actually being the solution. The most used gene type is the binary one with binary digits as Holland (1975) used (Sivanandam & Deepa, 2008) (See Figure 3.2.a). Later, different types of genes have been used according to the problem studied (See Figure 3.2.b).

A chromosome, in some way, stores information about solution that it represents. That requires a mapping mechanism between the solution space and chromosome. This mapping is called *representation (encoding)* of the solution, which is an abstract representation (Sivanandam & Deepa, 2008). A chromosome representation describes an individual in the population. There are a number of ways to represent a solution in a way that it is suitable for GA such as binary, real number, vector of real numbers, permutations, general data structure (array, tree, matrix and so on), and they are mostly depend on the nature of the problem (See Figure 3.2) (Rotlauf, 2006). The first step of designing a GA for a particular problem is to devise a suitable representation so that the problem becomes easily solvable by GA. The suitable representation also allows for easy application of genetic operators and computation of fitness (Suresh, Vinod, & Sahu, 1996).

1	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**

(a)

A	**C**	**A**	**B**	**C**	**D**	**E**	**D**	**E**	**E**

(b)

Figure 3.2 Chromosome representations

3.2.1.2 Initialization

Chromosomes evolve through successive iterations (*generations*). The set of individuals (solutions, chromosomes) of each generation is called a *population*. The *diversity* of a population is a measure of the number of the different solutions present. The number of individuals in the population gives the *population size*.

The *initial population* is created during an initialization phase and often generated *randomly* by assigning random values to the genes in the chromosomes. Some knowledge can be used by the GA to start the search from promising regions of the search space. Seeding the initial population with known good solutions or including a high-quality solution, obtained from another heuristic technique, can help a GA find better solutions rather more quickly than it can from a random start. However, there is also the possibility of inducing premature convergence to a poor solution (Reeves & Rowe, 2003).

3.2.1.3 The Fitness Function

During each generation of a GA, the chromosomes are evaluated, using some measures of fitness. Fitness is assigned to each chromosome in the current population by a fitness function. The evaluation procedure rates chromosomes in terms of their fitness. The fitness value reflects the quality of the solution represented by the chromosome. The fitness function is the same as the objective function to be optimized, so it is adjusted to the problem at hand (Kim et al., 1996). Sometimes, the fitness function can be the transformation of the objective function (Yu & Yin, 2009). The fitness value of each chromosome is calculated according to the given fitness (objective) function.

3.2.1.4 Selection

The *selection* mechanism determines which individuals will have all or some of their genetic material passed to the next generation. The most fit individuals (chromosomes) are selected from a population to form a basis for subsequent generations, i.e., for reproduction (Haupt & Haupt, 2004).

Selection can be based on many different criteria but it is usually based on a fitness value. The idea behind this is to select the best chromosomes for parents in a way by combining them to produce better offspring chromosomes. A comparative analysis of selection schemes used in GA was given by Goldberg & Deb (1991).

The most popular selection techniques are given in the following:

- *Roulette Wheel Selection (RWS)*: This technique works like a roulette wheel in which each slot on the wheel is paired with an individual of the population. The size of each slot is proportional to the corresponding individual’s fitness. The wheel is spun just many times as the population size. On each spin, the individual under the wheel's marker is selected to be in the pool of parents for the next generation (Mitchell, 1996). This procedure selects chromosomes proportional to their fitness scores.

- *Stochastic Universal Sampling*: This method uses a single wheel spin which is spun once, but with a number of equally spaced markers equal to the population size. This method gives each individual the proper number of trials to eliminate selection noise (Mitchell, 1996).

- *Tournament Selection*: At each iteration, this method chooses a number (tournament size) of individuals and selects the best one as a parent from this group into the next generation. This process is repeated for every parent needed as often as the population size (Goldberg & Deb, 1991). A larger value of tournament size increases the selective pressure while decreasing the population diversity (Kim et al., 1996).

- *Ranking selection*: The individuals of the population are ranked according to fitness, and the expected value of each individual depends on its rank rather than on its absolute fitness. The solutions are selected proportionally to their rank. The population is sorted from the best to the worst one, and each individual is copied as many times as possible, and then the proportionate selection is performed.

3.2.1.5 Genetic Operators

Genetic operators provide the basic searching mechanism of GAs. They are used to create new solutions based on existing solutions in the population. GAs use two main operators: crossover and mutation. The *crossover* operator has the role of combining pieces of information from different individuals in the population. The selected individuals called *parents* are joined in pairs and combine their genetic material to produce two new individuals called *offspring* with a probability equal to the *crossover rate* (Coley, 1999). There are many types of crossover operators such as binary-coded, real-coded, statistic-based, and permutation-based crossover operators. The popular permutation-based crossover operators which are used generally for line balancing problems are partially mapped crossover (PMX), order crossover (OX), cycle crossover (CX), position-based, and uniform crossover. Figure 3.3 shows two-point crossover

(2PX) which exchanges all genes between the two cutpoints mostly determined in a random way.

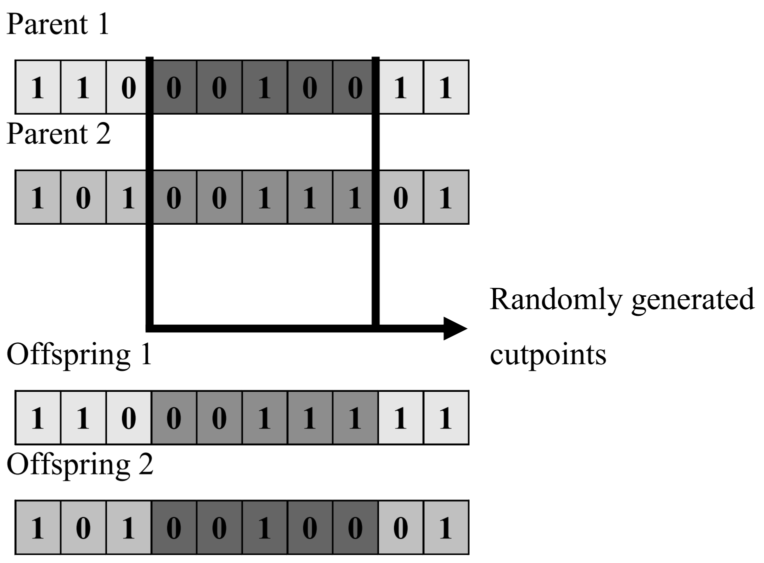


Figure 3.3 Two-point crossover

The main objective of the crossover operator is to transfer good characteristics of parents to offspring. Crossover depends on chromosome representation and can be very complicated. Although general crossover operations are easy to implement, building specialized crossover operation for a specific problem can greatly improve performance of GA (Mitchell, 1996).

After GA performs crossover, it performs *mutation* to finish production of new chromosomes. Mutation alters one or more genes with a probability equal to the *mutation rate* (Dreo, Siarry, Petrowski, & Taillard, 2006). Mutation makes small random changes to encoded solution; therefore it introduces a certain amount of randomness to the search. The aim of mutation is to prevent all solution being trapped into local optimum and to extend search space of the algorithm. This ensures diversity among individuals, preventing premature convergence. Mutation, as well as crossover, depends on chosen representation. There are many types of mutation operators such as insertion, inversion, reciprocal, and scramble mutation. Figure 3.4 shows mutation of a bit which involves flipping a bit, changing 1 to 0.

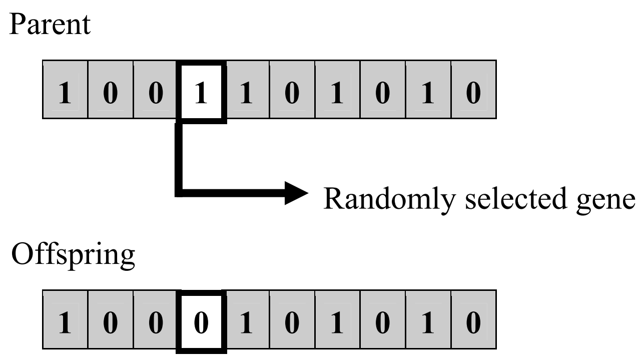


Figure 3.4 Mutation

3.2.1.6 Survival

The members of the new generation can be individuals from the current generation and/or offspring product of crossover or mutation. A survival approach is necessary to determine which individuals stay in the next population and which are replaced by offspring to keep the population size constant. The most common approach is *elitism* which allows the best chromosome in each generation to survive in the next generation. This is to make sure that the final population contains the best solution ever found. It is guaranteed that the best solution obtained during the generations to be preserved without being accidentally destroyed by genetic operators (Kim, Kim, & Cho, 1998). There are several approaches for the replacement. It is common to make one or a few exact copies of the best individual and place them directly in the next generation. But some approaches do the maintenance of the parents in the population. In either case, a random component is always present to avoid premature convergence to local optima. The tournament strategy or a local search algorithm can also be used as well as elitism for survival.

3.2.1.7 Termination

There are various stopping conditions for GAs as listed in the following (Sivanandam & Deepa, 2008):

- *Maximum generations:* When the specified number of generations has evolved, GA stops.
- *Elapsed time*: When a specified time has elapsed, GA will end. If the maximum number of generation has been reached before the specified time has elapsed, GA will end.
- *No change in fitness*: If there is no change in the best fitness of population for a specified number of generations, GA will end. If the maximum number of generation has been reached before the specified number of generation without any change, GA will end.
- *Stall generations*: GA stops if there is no improvement in the objective function for a sequence of consecutive generations during the length of stall generations.
- *Stall time limit*: GA stops if there is no improvement in the objective function during an interval of time in seconds equal to stall time limit.

3.2.2 Procedure of GAs

In the application of GAs, there are some steps which should be taken as stated below:

1. Choose a *representation scheme* for a possible solution (coding or chromosome representation)
2. Decide on how to create *the initial population.*
3. Define the *fitness function*.
4. Define the *genetic operators* to be used (reproduction, crossover, mutation, elitism).
5. Choose the *parameters* (population size, probability of genetic operators).

6. Define the *termination rule*.

By initialization, GA maintains a population of individuals to start the search. Each individual is coded as a chromosome and represents a solution to the problem at hand. Each individual is evaluated to give measure of its fitness.

After the evaluation of the initial population, chromosomes are selected on which the genetic operators are applied. In order to create new individuals, some individuals of the population undergo stochastic transformations by means of genetic operations. GAs use mainly two genetic operators, crossover and mutation, to direct the population to the global optimum. Crossover creates new individuals by combining parts (mating) from two individuals. This allows exchanging information between different solutions (chromosomes). Mutation creates new individuals by making changes (mutating) in a single individual and increases the variety in the population. Then new individuals, called offspring, are evaluated and a new population is formed. Passing through these steps completely is known as one generation.

After several generations (iteration number), the algorithm converges to the most fit individual, which represents an optimal or suboptimal solution to the problem at hand. This process is continued until a termination criterion is met. Figure 3.5 illustrates the main steps as a general flowchart of a GA.

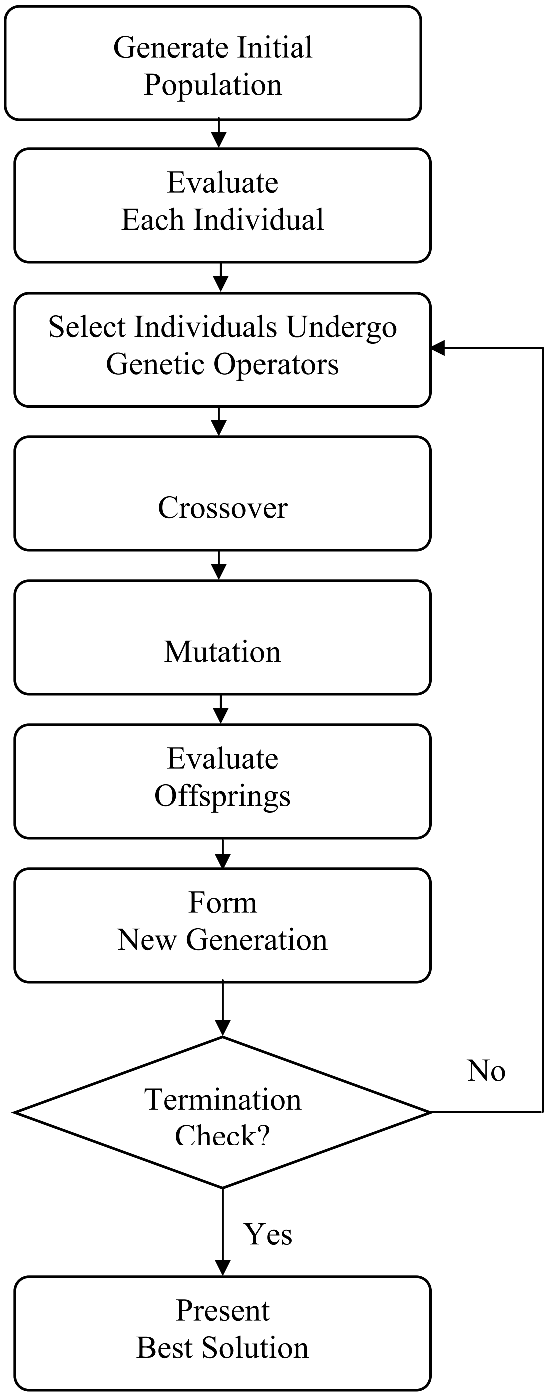


Figure 3.5 Main steps of a generalized genetic algorithm

3.2.3 Parameter Setting for GAs

One of the important issues in implementing a GA is setting the values of the various parameters, such as population size, crossover rate, and mutation rate. The well determined values can cause the algorithm to find an optimal or near-optimal solution efficiently. But suboptimal parameter values set by the user can result in a suboptimal algorithm performance. A given parameter value can have a different optimal value in different phases of the search. Choosing the appropriate parameter values is a hard task. Eiben, Michalewicz, Schoenauer, & Smith (2007) classified parameter setting approaches into two major groups: parameter tuning and parameter control as given in Figure 3.6.

Parameter tuning is the approach of searching for good values of the parameters before the run of the algorithm and then running the algorithm using these fixed values. Parameter tuning is a typical approach, but it is very time consuming. The reason is that it is done by experimenting with different values of many parameters and selecting the ones that give the best results on the test problems at hand.

Parameter control is the approach of starting a run with initial parameter values which are changed during the run. Different values of parameters might be optimal at different stages of the algorithm. Methods for changing the value of a parameter can be classified into three categories:

1. *Deterministic Parameter Control*: This method is used when the value of a strategy parameter is altered by some deterministic rule. This rule modifies the strategy parameter in a predetermined way without using any feedback from the search. A user-specified way, such a time-varying schedule, is generally used, i.e., the rule will be used when a set number of generations have elapsed since the last time the rule was activated.

2. *Adaptive Parameter Control*: This method is used when there is some feedback from the search to determine the direction or magnitude of the change to the strategy parameter. The assignment of the value of the strategy parameter may be based on the quality of solutions discovered by different operators/parameters. It is important that the updating mechanism used to control parameter values is externally supplied, rather than being part of the algorithm.

3. *Self-Adaptive Parameter Control*: To implement the self-adaptation of parameters, the idea of the evolution of evolution can be used. The parameters are encoded into the chromosomes and undergo mutation and recombination. The better values of these encoded parameters lead to better individuals. In turn they are more likely to survive and produce offspring and hence propagate these better parameter values. An updating mechanism of different strategy parameters is entirely implicit, i.e., they are the selection and variation operators of the algorithm itself.

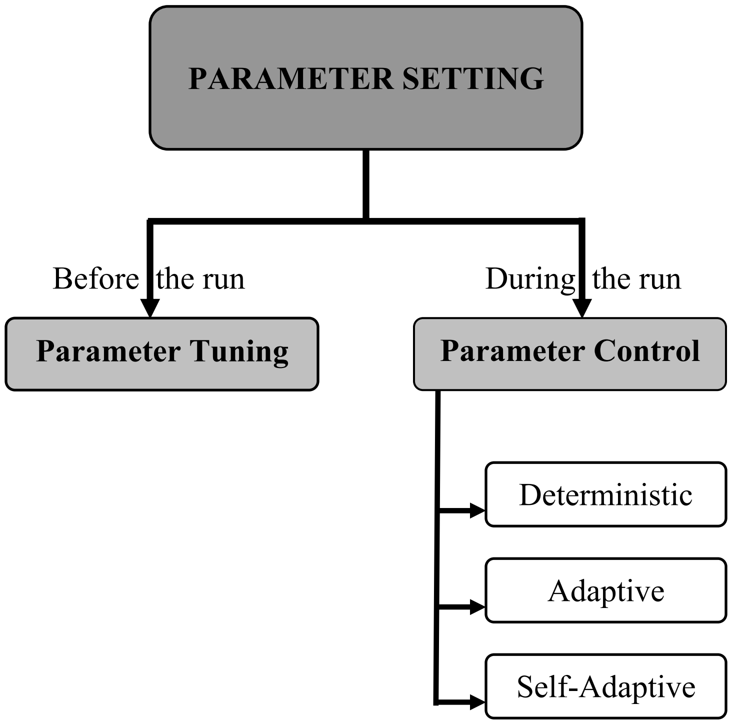


Figure 3.6 Taxonomy for parameter setting in GAs

3.3 Chapter Summary

In this chapter, main characteristics for GAs were presented in detail. The general procedure of GAs was given and their parameter setting is explained. Our study involves GAs for solving CCALBP; therefore the review of the literature for the application of GA approaches in line balancing will be given in the next chapter.

CHAPTER FOUR

LITERATURE REVIEW FOR APPLICATIONS OF GENETIC ALGORITHMS IN ASSEMBLY LINE BALANCING

4.1 Introduction

Genetic algorithms (GAs) are meta-heuristics that have been thoroughly used for solving ALBPs. Dimopoulos & Zalzala (2000) reviewed recent developments for the use of evolutionary computation in many manufacturing problems including assembly line balancing. Rekiek et al. (2002) presented a survey of the methods including exact methods, heuristics and meta-heuristics applied to ALBP. Aytug, Khouja, & Vergara (2003) reviewed the use of GAs to solve operations problems including assembly line balancing. Pierreval, Caux, Paris, & Viguier (2003) reviewed evolutionary approaches for solving several types of problems encountered in the area of manufacturing systems including ALBP. These studies have a very wide range of the application areas including supply chain management, facility layout design, assembly lines, etc.

Up-to-date analysis of the bibliography and available state of the art procedures for SALBP family of problems are given by Scholl & Becker (2006) and for GALBP by Becker & Scholl (2006). Tasan & Tunali (2008) reviewed the current applications of GAs in assembly line balancing with the focus on solving all types of ALBPs using GAs. Their study gave a structural framework to classify the reviewed papers according to the type of ALBP studied, the GA methodology and the performance specifications.

In this chapter, the published studies for applications of GAs in assembly line balancing are classified based on the structural framework given by Tasan & Tunali (2008). In Section 4.2, the literature is organized in chronological order. In Section 4.3, the conclusions about the literature review are given and in Section 4.4, the chapter is summarized.

4.2 Literature Review

Referring to the classification of ALBP by Baybars (1986), the published literature is reviewed under two types of ALBPs studied: the literature of GAs for SALBP is reviewed in Section 4.2.1 and the literature of GAs for GALBP is reviewed in Section 4.2.2. The reviewed studies based on the type of the problem and the objective functions are given in order in Table 4.1.

4.2.1 Research on SALBP

SALBP is the simplest version of ALBP which involves mass-production of only one homogeneous product, paced line with fixed cycle time, deterministic and independent processing times, no assignment restrictions besides the precedence constraints, serial line layout, one-sided and equally equipped workstations, and fixed rate launching

SALBP was first solved with a GA by Falkenauer & Delchembre (1992). To solve SALBP Type-1, the authors pointed out the weaknesses of a standard GA when applied to grouping problems, and introduced the Grouping Genetic Algorithm (GGA) which was presented by Falkenauer (1991). The GGA differed from the classic GA in two important aspects. First, a special encoding scheme was chosen in order to make the structure of chromosomes more group-oriented. Second, special genetic operators, which were suitable for the chromosomes, were used with the given encoding. Falkenauer & Delchembre (1992) presented efficient crossover and mutation operators for the bin packing problem, and modified them to solve SALBP Type-1. The authors tested the performance of the algorithm on randomly generated data.

Table 4.1 Chronological list of GA studies for assembly line balancing

PROBLEM SPECIFICATIONS			
Year	**Researcher(s)**	**Problem Type**	**Objective Function**
1992	Falkenauer & Delchambre	SALBP	Type-1
1994	Leu et al.	SALBP	Type-1
1994	Anderson & Ferris	SALBP	Type-2
1995	Rubinovitz & Levitin	SALBP	Type-2
1995	Tsujimura et al.	GALBP (SMALBP)	Type-1
1996	Kim et al.	SALBP	Type-1, 2, 3, 4, 5
1996	Suresh et al.	GALBP (SMALBP)	Type-1
1997	Falkenauer	GALBP (SMALBP)	Type-1
1998	Ajenblit & Wainwright	GALBP (SMALBP)	Type-1
1998	Chan et al.	GALBP (SMALBP)	Type-1
1998	Kim et al.	SALBP	Type-2
1999	Rekiek et al.	SALBP	Equal Piles
2000	Bautista et al.	SALBP	Type-1, Type-2
2000c	Kim et al.	GALBP (SMALBP)	Type-1
2000	Ponnambalam et al.	SALBP	Type-1, Type-3
2000	Sabuncuoglu et al.	SALBP	Type-1
2001	Carnahan et al.	SALBP	Type-2
2001a	Simaria & Vilarinho	GALBP (MMALBP)	Type-2
2002	Chen et al.	GALBP (Assembly Planning)	Type-2
2002	Goncalves & De Almedia	SALBP	Type-1
2002	Miltenburg	GALBP (MMALBP & sequencing simultaneously)	Type-1
2002	Valente et al.	GALBP (SMALBP)	Type-2
2004	Brudaru & Valmar	GALBP (SMALBP)	Type-1
2004	Martinez & Duff	GALBP (SMALBP)	Type-1
2004	Simaria & Vilarinho	GALBP (MMALBP)	Type-2
2004a, 2004b	Stockton et al.	SALBP	Type-1
2005	Brown & Sumichrast	SALBP	Type-1
2006	Haq et al.	GALBP (MMALBP)	Type-1
2006	Levitin et al.	GALBP (SMALBP)	Type-2
2007	Baykasoglu & Ozbakir	GALBP (SMALBP)	Type-1
2008	Guo et al.	GALBP (SMALBP)	
2008	Hwang et al.	GALBP (SMALBP)	Type-1
2009	Gao et al.	GALBP (SMALBP)	Type-2
2009	Hwang & Katayama	GALBP (MMALBP)	Type-1
2009	Moon et al.	GALBP (SMALBP)	Type-1
2009	Kim et al.	GALBP (SMALBP)	Type-2
2009	Yu &Yin	SALBP	Type-1, Type-3

Leu et al. (1994) showed how a GA was used to generate feasible line balances step by step to solve SALBP Type-1. The authors explained how to create feasible population in the initialization, and feasible children after crossover and mutation. As extensions, to improve GA solutions, Leu et al. (1994) used solutions of heuristic procedures in the initial population, and also demonstrated the possibility of balancing assembly lines with multiple criteria and side constraints such as allocating a task in a station by itself.

Anderson & Ferris (1994) were the first to solve SALBP Type-2 with a GA. The authors described a standard implementation of GA for SALBP and carried out extensive computational testing for it. Anderson & Ferris (1994) also introduced an alternative parallel version of the algorithm, and compared it with the serial implementation.

Rubinovitz & Levitin (1995) developed a GA to solve SALBP Type-2. The authors compared the proposed GA with MUST algorithm suggested by Dar-El & Rubinovitch (1979). Totally 36 problems with different flexibility ratios and with different number of stations were solved. The results showed that the proposed GA performed much faster than MUST for problems with large number of stations (more than 20) and high flexibility ratio.

Kim et al. (1996) presented a GA to solve SALBP with various objectives. The objectives were to minimize the number of workstations (Type-1), to minimize cycle time (Type-2), to maximize workload smoothness (Type-3), to maximize work relatedness (Type-4), and the multiple-objective with the objective of Type-3 and Type-4 (Type-5). The authors compared five standard crossover operators for the proposed GA to solve Type-1, Type-2, Type-3, and Type-4 problems. The proposed GA was also compared with the well-known heuristics in the literature. The results showed that in all of the four types, the proposed GA could provide much better solutions than the other heuristics on several test problems such as Kilbridge & Wester's 45-task and Tonge's

70-task problems. As an extension, Kim et al. (1996) implemented a multiple objective GA (MOGA) to solve SALBP Type-5. The authors concluded that for a multiple objective problem (Type-5), MOGA produced diverse Pareto optimal solutions. But according to Rekiek et al. (2002), the encoding scheme used by Kim et al. (1996) was not well suited for the grouping problem they dealt with.

Kim et al. (1998) proposed a heuristic-based GA to solve SALBP Type-2 with the objective of workload smoothness. The authors placed the emphasis on utilization of problem-specific information and heuristics in order to get high quality solutions. The experiments for the proposed GA were carried out on five test-bed problems: Kilbridge & Wester's 45-task, Tonge's 70-task, Arcus' 83-task, Arcus' 111-task, and Bartholdi's 148-task problems. The results of the experiments showed that the proposed GA outperformed the three existing heuristics and the standard GA in optimizing the workload smoothness.

Rekiek et al. (1999) presented a new algorithm using GGA, based on an Equal Piles approach. The proposed GGA was heavily modified to respect the precedence constraints to solve SALBP. Equal Piles approach for SALBP warranted to obtain the desired number of stations, and tried to equalize the station workloads (Rekiek et al., 2002). The authors applied the proposed GGA to Buxey's 29-task problem and presented this case study.

Bautista, Suarez, Mateo, & Companys (2000) developed a Greedy Randomized Adaptive Search Procedure (GRASP) and a GA to solve an extension of SALBP. The extension of SALBP, which the authors considered, had incompatibilities between groups of tasks. If two tasks were incompatible, they could not be assigned to the same workstation. The objectives were, first, to minimize the number of workstations (Type-1), and then, minimize the cycle time (Type-2) with the determined number of stations. GRASP was obtained from the application of some classic heuristics, based on priority rules. Bautista et al. (2000) used weights for revising GRASP and also proposed Greedy

Randomized Weighted Adaptive Search Procedure (GRWASP). The authors carried a comparative study and found that the proposed GA and GRWASP performed better than the greedy heuristics and GRASP.

Ponnambalam, Aravindan, & Naidu (2000) proposed a MOGA to solve SALBP with multi-objectives. The performance criteria, to optimize simultaneously, were the number of workstations (Type-1), line efficiency and the smoothness index (Type-3). The developed GA was compared with six popular heuristic algorithms from the literature. A set of 20 networks from the literature were used for comparison. Each network was solved for five different cycle times. The number of tasks varied from 7 to 50. The authors found that the proposed GA performed better than the heuristics in all of the performance measures; however, the execution times were longer because of more iterations for global optimal solutions.

Sabuncuoglu, Erel, & Tanyer (2000) proposed a GA with a special chromosome structure partitioned dynamically through the evolutionary process to solve SALBP. By using some concepts of simulated annealing (SA), a new elitism structure was implemented in the model to determine the survival of the individual solutions. The authors used a fitness function including two parts. The first part aimed at reducing the imbalance, and the second one at minimizing the number of stations (Type-1). The proposed GA was compared with the well-known heuristics in the literature. The results showed that the proposed GA outperformed the other heuristics on several test problems such as Kilbridge & Wester’s 45-task and Tonge’s 70-task problems.

Carnahan, Norman, & Redfern (2001) developed three heuristics to solve SALBP Type-2 with the objectives of minimizing the worker’s fatigue and the cycle time to explore the incorporation of physical demand criteria in line balancing. The developed heuristics were a multiple ranking heuristic, a combinatorial GA, and a problem space GA. Each heuristic was tested using a set of 100 Type-2 ALB problems. The literature problems were Buxey’s 29-task, Sawyer’s 30-task, Gunther's 35-task, Kilbridge &

Wester's 45-task, Hahn's 53-task, Warnecke's 58-task, Wee-Mag's 75-task, Arcus' 83-task, Lutz's 89-task, Lutz's 89-task, Muckherje's 94-task and Bartholdi's 148-task problems. The results showed that the problem space GA was found to be the best of all three heuristics.

Goncalves & De Almedia (2002) developed a hybrid GA that combined a heuristic priority rule, a local search procedure and a GA to solve SALBP Type-1. The proposed hybrid GA used a random key alphabet, an elitist selection and a parameterized uniform crossover. The authors presented computational experiments on 269 instances of three problem sets found in the literature: The Talbot set, the Hoffman set, and the Scholl set. The results showed that the algorithm performed remarkably well.

Stockton et al. (2004a, 2004b) researched the applications of GAs for solving problems in various areas such as designing and planning of manufacturing operations. These problems were assortment planning, aggregate planning, lot sizing in material requirement planning, line balancing and facilities layout. The authors applied a GA to solve SALBP Type-1 in Stockton et al. (2004a). The authors also performed computational experiments in Stockton et al. (2004b). With the help of these experiments, the relationships between problem characteristics and performance of individual operator types and parameter values were identified as a set of guidelines.

Brown & Sumichrast (2005) compared the performance of a GGA (Falkenauer, 1991) against the performance of a standard GA for solution quality and run time. The types of grouping problems selected to test were the bin packing problem, machine part cell formation problem and SALBP Type-1. Both GA and GGA obtained optimal solutions for all test problems, but the GA required much more time.

Yu & Yin (2009) developed an adaptive GA using adaptive crossover and mutation operators to solve SALBP. The authors considered to minimize the number of workstations (Type-1) and to maximize the workload smoothness (Type-3). In the proposed adaptive GA, the probability of crossover and mutation was dynamically adjusted according to the individual's fitness value. The individuals with higher fitness values were assigned to lower probabilities of genetic operators. Two computational examples demonstrated that the proposed approach was better than the Kilbridge-Wester algorithm and Monte-Carlo algorithm. The adaptive GA provided an effective convergence and efficient computation speed.

4.2.2 Research on GALBP

GALBP include all of the problems which are not SALBP. These are balancing problems with different additional characteristics such as single or mixed model production, cost functions, equipment selection, paralleling, U-shaped or two-sided line layouts with stochastic, fuzzy or dependent processing times.

GALBP was first solved with a GA by Tsujimura, Gen, & Kubota (1995). The authors proposed a GA to solve SMALBP Type-1. In order to treat the data of real world problems, the authors used fuzzy numbers by triangular membership functions to represent task times. Special mechanisms and operators ensured the feasibility of solutions. Tsujimura et al. (1995) illustrated the application of the proposed GA on an 80-task problem.

Suresh et al. (1996) proposed a general approach to solve SMALPB Type-1 with stochastic processing times using GAs. First, the authors presented a GA working with only feasible solutions. Then, an alternative version of the proposed GA working with two populations, one allowing only feasible solutions and the other allowing a certain amount of infeasible solutions, was presented. Suresh et al. (1996) claimed that the presence of infeasible solutions allowed a smoother search space and helped in escaping from certain local minima. The modified version of GA gave better results than the first one with only feasible solutions according to the results of the experiments.

Falkenauer (1997) developed a GA based on GGA proposed by Falkenauer & Delchembre (1992) and Branch & Bound (B&B) algorithm to solve SMALP Type-1. By GGA, tasks with resource dependent processing times were assigned to workstations along the line by minimizing the number of the stations; and then by B&B algorithm, the equipments were selected to carry out the operations by minimizing the cost of the line.

Ajenblit & Wainwright (1998) were the first to solve U-shaped SMALBP Type-1 by using a GA. The authors considered three possible definitions for fitness function. One was to minimize total idle time, one was to minimize mean-squared idle time for balancing workload among the stations, and the other was a combination of both. Ajenblit & Wainwright (1998) developed six assignment algorithms in order to determine how a particular order of tasks in a chromosome can be assigned to workstations. The proposed algorithm was tested with 61 test instances from Merten's 7-task, Bowman's 8-task, Jaeschke's 9-task, Jackson's 11-task, Dar-El's 11-task, Mitchell's 21-task, Heskiaoff's 28-task, Kilbridge & Webster's 45- task, Tonge's 70-task, Arcus' 83-task, and Arcus' 111-task problems. The proposed GA obtained the same results as previous researchers in 49 case, and superior results in 11 cases.

Chan, Hui, Yeung, & Ng (1998) applied a GA to solve SMALBP Type-1 in the clothing industry. In their study, line balance was achieved by the assignment of workers with varying skill levels to workstations. The objective was to smooth system's throughput while minimizing slack time in the apparel industry. The authors compared the proposed GA with a greedy algorithm by using a 41-task real case problem. The results showed that the proposed GA was much superior to the greedy algorithm.

Kim et al. (2000c) developed a new GA to solve a two-sided SMALBP Type-1. In a two-sided assembly line, different assembly tasks were performed on the same product item in parallel at both sides of the line for producing large-sized high-volume products. The positional constraints due to facility layout were also considered in their study. The objective was to minimize the number of workstations. The proposed GA was tested with five test-bed problems of 9, 12, 24, 65, and 148 tasks. The performance of GA was compared to integer programming (IP) and the first-fit rule heuristic, and the GA showed the best performance.

Simaria & Vilarinho (2001a) proposed a GA to solve MMALBP Type-2 with parallel workstations. The proposed GA was based on a model developed in Simaria & Vilarinho (2001b) to solve SMALBP Type-2 by using a simulated annealing (SA) approach. The authors proposed and illustrated an iterative search procedure that solved MMALBP Type-2 at first, and then the GA was employed to minimize the cycle time and the workload balance (Tasan & Tunali, 2008).

Chen, Lu, & Yu (2002) proposed a hybrid GA combined with a self-tuning mechanism, which changed the infeasible sequence of chromosomes as to prevent the violation of precedence relations, to solve SMALBP Type-2 involving various objectives. The objectives were minimizing cycle time, maximizing workload smoothness, minimizing the frequency of tool changes, minimizing the number of tools used, and minimizing the total penalty of assembly relations. The authors tried to find Pareto optimal solutions to this multiple objective assembly planning problem. The experiments showed that the proposed GA solved the multi-objective problem more quickly than conventional heuristics. The proposed GA found many feasible solutions which could help to choose a suitable alternative of the assembly plan for modeling a flexible assembly system.

Miltenburg (2002) solved two problems simultaneously with a GA: U-shaped MMALBP Type-1 and model sequencing. The author introduced the problem as mixed–model U-line balancing and scheduling (MMULB/S) problem and presented a mathematical model for it. The proposed GA produced good results in the computational experiments with 128 problem instances generated from Kilbridge & Webster’s 45- task and Arcus’ 83-task problems.

Valente, Lopes, & Arruda (2002) proposed a GA to solve two-sided SMALBP Type-2. The authors tried to solve a real world application in a car assembly facility. Valente et al. (2002) determined the parameters of GA after several experiments. The best solution of the proposed GA reduced the total assembly time of the current line.

Brudaru & Valmar (2004) developed a hybrid GA to solve SMALBP Type-1 with fuzzy processing times. The authors proposed embryonic chromosome representation, a special version of the task based chromosome. The only difference between the two was that the embryonic representation of a solution considered the subsets of solutions rather than the individual solutions. During the generations, the embryonic chromosome evolved through a full length solution, therefore the chromosome length varied throughout the generations. The hybrid method was found to take longer time with respect to the quality of solution.

Martinez & Duff (2004) proposed a GA based on the method proposed by Pannambalam et al. (2000) to solve U-shaped SMALBP Type-1. At first, the authors solved the problem by modifying 10 heuristic rules found in the literature to solve SALBP, such as maximum ranked positional weight, maximum total number of follower tasks or precedence tasks, and maximum processing time. Martinez & Duff (2004) tested the modified heuristic rules with 8 test-bed problems of 5, 8, 9, 11, 12, and 21 tasks. Later, the authors used the results of heuristic rules in the initialization of the proposed GA and illustrated it by using Jackson's 11-task problem. It was shown that the addition of a GA improved the solution.

Simaria & Vilarinho (2004) presented a mathematical model and developed an iterative GA-based procedure for the MMALBP Type-2 with parallel workstations and zoning constraints based on the studies of Simaria & Vilarinho (2001b) and Vilarinho & Simaria (2002). The objective was to minimize the cycle time while smoothing the workload balance within each workstation. The proposed procedure consisted of three stages: a constructive heuristic for finding initial solutions and two GA procedures working iteratively. The computational experiments were done by using the test problems developed by the authors. The algorithm reached the optimal or near-optimal solutions and the performance was found to be efficient.

Haq et al. (2006) introduced a hybrid GA to solve MMALBP Type-1. The objective was to minimize the number of workstations. Different models were transformed into an equivalent single model by using a combined precedence diagram. An illustrative example was used as a problem. First, the problem was solved by a classical GA method, and then by the modified ranked positional weight technique (RPWT) (Helgeson & Birnie, 1961). The results were compared and it was shown that the classical GA method gave superior results than modified RPWT. Next, the solutions of modified RPWT were randomly introduced into the initial population of GA. The hybrid GA gave better performance than the classical GA.

Levitin et al. (2006) suggested an algorithm based on a GA approach for solving large and complex robotic assembly line balancing problems (RALBP), a special kind of SMALBP Type-2. RALBP attempted to assign the most efficient type of robots to line stations optimally and balance the distribution of work among the stations. The objective was to minimize the cycle time of an assembly line with the given number of stations (Type-2). Two different procedures were used for adapting GA to the defined problem: a recursive and a consecutive procedure. To improve the quality of solutions, the local exchange procedure was applied. By testing with a set of randomly generated problems, the best combination of the procedures and GA parameters were reached. The developed GA was shown to be consistent and robust. Its comparison with a B&B algorithm achieved solutions of higher quality. It was concluded that GA gave better results in an efficient way for solving large and complex problems.

Baykasoglu & Ozbakir (2007) proposed a multiple-rule-based GA to solve U-shaped SMALBP Type-1 with stochastic processing times. The authors integrated COMSOAL method, task assignment heuristics from the literature and a GA. Each gene in a chromosome represented a task assignment rule. For deducting a solution from a chromosome, the COMSOAL procedure was used iteratively. The number of tasks assignment rules used was 10. The task times were assumed to be normally distributed. The proposed algorithm was tested with 7 categories of test problems from Merten's 7-task, Bowman's 8-task, Jaeschke's 9-task, Jackson's 11-task, Mitchell's 21-task, Heskiaoff's 28-task, Kilbridge & Webster's 45-task problems. The results were compared with the optimal solutions found by Urban & Chiang (2006). The proposed algorithm found optimal solutions for all problems, except one case within smaller computational times.

Guo, Wong, Leung, Fan, & Chan (2008) proposed a bi-level GA with multi-parent crossover for solving a kind of GALBP, a flexible assembly line balancing (FALB) problem with work sharing and workstation revisiting allowed. The objective function had two parts. One was to meet the desired cycle time of each order by using penalty weights and the other was to minimize the total idle time of the assembly line. The proposed bi-level multi-parent GA was composed of two GAs where the second-level GA was nested in the first-level GA. The first-level GA generated the optimal task assignment to workstations. The second-level GA determined the task proportion of the operation that was assigned to different workstations. Then, a heuristic operation routing rule was used to route the shared operation of each product to an appropriate workstation. Based on the industrial data, four experiments were conducted to validate the proposed optimization model. The experimental results demonstrated that the proposed GA solved the FALB problem effectively.

Hwang, Katayama, & Gen (2008) proposed a MOGA to solve U-shaped SMALBP Type-1. The performance criteria were the number of workstations (the line efficiency) and the variation of workload. Both the traditional straight line system and the U-shaped assembly line system were considered. The GA provided workable solutions whereas the optimal U-shaped assembly line solution had an improved line efficiency compared to the optimal straight line solution. The proposed approach produced good or even better line efficiency of workstation integration and improved the variation of workload.

Gao, Sun, Wang, & Gen (2009) proposed an innovative GA hybridized with local search to solve robotic assembly line balancing (rALB-II) problem, a special kind of SMALBP Type-2. They used a partial representation technique in GA. Five local search procedures were developed. They tested the performance of the hybrid GA on 32 generated rALB-II problems and compared with other methods. The results showed that the proposed approach was quite efficient to find out the optimal solutions for the problems.

Hwang & Katayama (2009) proposed a multi-decision amelioration procedure with GA to solve U-shaped MMALBP Type-1. The number of workstations (the line efficiency) and the variation of workload were considered simultaneously as the performance criteria. They tested the proposed approach by using three well-known problems and one case study. The results showed that the GA provided better solutions.

Kim, Song, & Kim (2009) presented a mathematical model and a neighborhood GA (n-GA) to solve two-sided SMALBP Type-2. The features of proposed GA were designed according to the specifications of two-sided SMALBP. To promote population diversity and search efficiency, the strategy of localized evolution and steady-state reproduction were adopted. This localized strategy was a structure of neighbor set, so the name of GA was called neighborhood GA. The performance of GA was compared with that of a heuristic and an existing GA with various experimental problems. The experimental results showed that the proposed GA outperformed the heuristic and the compared GA in terms of solution quality and convergence speed.

Moon, Logendran, & Lee (2009) developed a GA to solve SMALBP Type-1 in which multi-functional workers had different salaries depending on their skills. The objective was to minimize the total annual workstation costs and the annual salary of the assigned workers within a predetermined cycle time. The efficiency of the developed GA was demonstrated by numerical examples. For the small and medium-sized test problems, the GA found optimal solutions more rapidly than mathematical programming.

4.3 Conclusions for Literature Review

The summary of the literature review is given in Table 4.1 according to the type and the objective functions of the ALBP studied.

Table 4.2 gives the summary of the studies with respect to proposed GA method, formation of initial population, the genetic representation of chromosomes, the evaluation of fitness, crossover and mutation operators, selection scheme, feasibility issues and the termination criteria used.

In most of the articles GAs are found to be superior to the well-known methods. Based on the published studies, researchers employed precedence graphs to summarize and visualize precedence constraints rather than using more effective tools.

More than half of the articles surveyed (22 out of 37) focused on GALBP, the general type of ALBP. A trend is noted for studying new kinds of problems or extensions of SALBP included in GALBP.

In some of the articles, genetic operators ensured feasibility of individuals for a certain representation. But in some of the articles, infeasibility was allowed in the population. It was claimed that the presence of infeasible solutions allowed a smoother search space and helped in escaping from certain local minima.

It is noted that parameters were optimized in the recent studies. But the information about parameter optimization was not given in detail. There is not enough information about how to optimize parameters when solving ALBPs with GAs.

It is noted that standardized benchmark problems were used for comparison. If the problem studied was new, new benchmark problems were generated using the literature problems.

Then, based on the findings and the insights gained above, this study proposes to develop a GA employing a different tool for precedence relations to solve a new kind of ALBP, called CCALBP. In order to balance a line by addressing alternative ways of assembling a product, the proposed GA employs a rule-base instead of a precedence graph. To balance a line based on the rule-based modeling of assembly constraints, the proposed GA is employed to solve CCALBP. Since CCALBP is a new problem, benchmark problems are generated for computational experiments.

The proposed GA allows infeasibility. Therefore, the objective function includes penalty function, and the genetic operators are not problem specific. But the chromosome representation is problem specific.

4.4 Chapter Summary

In this chapter, the summary of the main specifications with the objectives for the problems studied is reviewed to identify the recent research issues. The proposed GAs are given in chronological order. The summary of the specifications related to the proposed GAs such as chromosome representations, genetic operators and the fitness functions is reviewed and listed in Table 4.2.

Table 4.2 Chronological list of GA studies for assembly line balancing with respect to GA specifications

GENETIC ALGORITHM SPECIFICATIONS										
Year	**Researcher(s)**	**Method**	**Initial Population& Size of population**	**Chromosome representation**	**Crossover type & probability (Pc)**	**Mutation type & probability (Pm)**	**Selection type (for mating)**	**Survival type (replacement or reproduction)**	**Feasibility Force /Repair /Penalty**	**Termination Criteria**
1992	Falkenauer & Delchambre	Standard (GGA)	Random	Grouping based & variable length	Modified BPCX	Modified BPM	-	-	Force	Up to 10000 generation
1994	Leu et al.	Standard	Random+ Heuristics & popsize=20 (growing)	Task based & length=no. of tasks	OX & Pc=0.98	Scramble & Pm=0.02	Roulette wheel	Elitism	Force	Up to 500 generation + converge
1994	Anderson & Ferris	Standard & Parallel	Random+ Heuristics & popsize=64	Workstation based & length=no of tasks	One point crossover & Pc=0.6-0.7-0.8	One point & Pm=0.005-0.04	Stochastic universal sampling	Elitism	Penalty	Up to 350 generation + converge
1995	Rubinovitz & Levitin	Hybrid GA	Random	Task based & length=no of tasks	Fragment Reordering Crossover (FRG)	FRG mutation (FRGm)	Randomly	Elitism	Force	Up to T generation
1995	Tsujimura et al.	Standard	Random	Task based & length=no of tasks	PMX	Swap mutation	Elitism	Elitism	Repair	-
1996	Kim et al.	Standard	Random & popsize=100	Task based & length=no. of tasks	Standard and non standard crossover & Pc=0.4-0.6	Standard and non standard mutation & Pm=0.2-0.4	Tournament	Elitism	Repair	-
1996	Suresh et al.	Standard	Random & popsize=40-60 (with 2 population)	Workstation based & length=no. of tasks	One point crossover & Pc=0.5-0.7	Interchange mutation & Pm=0.01	Elitism	Elitism	Repair in std GA & penalty in 2 pop GA	-

Table 4.2 (cont) Chronological list of GA studies for assembly line balancing with respect to GA specifications

GENETIC ALGORITHM SPECIFICATIONS										
Year	**Researcher(s)**	**Method**	**Initial Population& Size of population**	**Chromosome representation**	**Crossover type & probability (Pc)**	**Mutation type & probability (Pm)**	**Selection type (for mating)**	**Survival type (replacement or reproduction)**	**Feasibility Force /Repair /Penalty**	**Termination Criteria**
1997	Falkenauer	GGA & Branch and Bound	Random	Grouping based & variable length	Modified BPCX	Modified BPM	-	-	Force	-
1998	Ajenblit & Wainwright	Standard	Random & popsize=100	Task based & length=no. of tasks	OX	Not used	-	-	Force	-
1998	Chan et al.	Standard	Random & popsize=50	Task based & length=no. of tasks	Uniform (uniform order- based) P_c=0.65	SSM Scramble & P_m=0.008	Roulette wheel	Survive children	Force	Terminate at 5000 s
1998	Kim et al.	Heuristic based GA	Random & popsize=100	Workstation based & length=no. of tasks	HSX & Pc=0.7-0.9	HSM & Pm=0.1-0.2	Tournament	Elitism	Force	Converge
1999	Rekiek et al.	Hybrid GGA	Random	Grouping based & variable length	Modified BPCX	-	-	-	Force	-
2000	Bautista et al.	Heuristic based GA	-	Heuristic based & length=no. of heuristics	-	-	-	-	No need	Up to T generation
2000c	Kim et al.	Standard	Random+ Heuristics	group number	Structured one point crossover (SOX) & Pc=50%	Random Pm=10%	Tournament	Elitism	-	Up to T generation
2000	Ponnambalam et al.	Standard	Random	Heuristic based & length=14 (no of heuristic)	Two point crossover	Random	Roulette wheel	Elitism	No need	-
2000	Sabuncuoglu et al.	Standard	Random	Task based & length=no. of tasks	Order crossover	Scramble	Roulette wheel	Elitism	Force	Up to T generation

Table 4.2 (cont) Chronological list of GA studies for assembly line balancing with respect to GA specifications

GENETIC ALGORITHM SPECIFICATIONS										
Year	**Researcher(s)**	**Method**	**Initial Population& Size of population**	**Chromosome representation**	**Crossover type & probability (Pc)**	**Mutation type & probability (Pm)**	**Selection type (for mating)**	**Survival type (replacement or reproduction)**	**Feasibility Force /Repair /Penalty**	**Termination Criteria**
2001	Carnahan et al.	Hybrid GA	Random & popsize=60	Task based & length=no. of tasks	FRG & Pc=0.6	FRGm	Roulette wheel	Elitism	Force	Up to T generation + converge
2001a	Simaria & Vilarinho	Two staged iterative GA	Random+ Heuristics	Workstation based & length=no. of tasks	SOX	One point	Tournament	Elitism	Repair	Up to T generation + converge
2002	Chen et al.	Standard	Random + Heuristics	Workstation based &	Order1-Order2-PMX-Cycle	Swap	Roulette wheel	Elitism	Repair using self-tuning	-
2002	Goncalves & De Almedia	Standard & hybrid with heuristic priority rules	Random+ Heuristics & popsize= no of tasks	Random key heuristic based & length=no. of tasks	Uniform crossover & Pc=0.7	Randomly generate & Pm=0.2	Copy 15%	Elitism	No need	Up to (3 X no. of tasks)
2002	Miltenburg	Standard	Random & popsize=50	Combination of task based and model sequence based & length= no of tasks+ model numbers	OX and Cycle	Swap	Rank selection with elitism	Elitism	Repair	Terminate at 300 s
2002	Valente et al.	Standard	Random & popsize=100	Workstation based & length=13 (no. of tasks)	One point crossover & Pc=0.8	Simple bit mutation & Pm=0.04	Stochastic universal sampling	Elitism	Penalty	Up to 200 generation

Table 4.2 (cont) Chronological list of GA studies for assembly line balancing with respect to GA specifications

GENETIC ALGORITHM SPECIFICATIONS										
Year	**Researcher(s)**	**Method**	**Initial Population& Size of population**	**Chromosome representation**	**Crossover type & probability (Pc)**	**Mutation type & probability (Pm)**	**Selection type (for mating)**	**Survival type (replacement or reproduction)**	**Feasibility Force /Repair /Penalty**	**Termination Criteria**
2004	Brudaru & Valmar	GA & Branch&Bound		Embryonic & variable length	-	-	-	-	-	-
2004	Martinez & Duff	Standard	Random & popsize=20	Heuristic based & length= 10 (no of heuristic)	-	-	-	-	No need	-
2004	Simaria & Vilarinho	Two staged iterative GA	Random+ Heuristics	Workstation based & length=no. of tasks	SOX	One point mutation	Tournament	Elitism	Repair	Up to T generation + converge
2004a 2004b	Stockton et al.	Standard		Binary	Two point crossover & Pc=0.6-0.65-0.7-0.75	Rm=0.05-0.025- 0.005	Roulette wheel	Elitism	Penalty	-
2005	Brown & Sumichrast	Standard (GGA)	-	Grouping based & variable length	Modified BPCX	-	-	-	Force	-
2006	Levitin et al.	Hybrid GA	Random& popsize=100	Task based & length=no of tasks	FRG	Swap & Pm=0.01	Randomly	Elitism	Force	Up to T generation
2006	Noorul Haq et al.	Hybrid GA	Random+ Heuristic	-	Pc=0.8	Pm=0.05	-	-	-	-
2007	Baykasoglu & Ozbakir	Hybrid with COMSOAL & task assignment rules	Random& popsize=50	E108Assignment rule based & length=no of tasks	One-point, two-point, uniform, mixed crossover & P_c=0.8-1	P_m=0.08	Roulette wheel	-	No need	Up to T generation

Table 4.2 (cont) Chronological list of GA studies for assembly line balancing with respect to GA specifications

GENETIC ALGORITHM SPECIFICATIONS										
Year	**Researcher(s)**	**Method**	**Initial Population& Size of population**	**Chromosome representation**	**Crossover type & probability (Pc)**	**Mutation type & probability (Pm)**	**Selection type (for mating)**	**Survival type (replacement or reproduction)**	**Feasibility Force /Repair /Penalty**	**Termination Criteria**
2008	Guo et al.	Two-staged two Gas (2) nested in (1)	(1) heuristic & (2) random	Work sharing & workstation revisiting based & length= no. of workstations	(1) modified fitness-based scanning (2)center of mass	(1) modified inversion (2)nonuniform	Tournament	-	Force	Up to T generation + converge
2008	Hwang et al.	Ttandard	Random & popsize=100	Priority-based & length=no of tasks	two point-based WMX P_c=0.7	Swap P_m=0.3	Roulette wheel	-	Repair	Up to T generation + converge
2009	Gao et al.	Hybrid with local search procedures	- popsize=100	2 chromosomes (1) task sequence (2) robot assignment based & (1) length=no of tasks (2) length=no. of workstations	Mixed crossover (1) OX (2) PMX & P_c=0.8	Allele-based (1) P_m=0.05 (2) P_m= 0.1	-	-	Repair	Up to T generation +
2009	Hwang & Katayama	Two-staged	Random & popsize=100	Priority-based & length=no of tasks	Two point-based WMX & P_c=0.7	Swap P_m=0.3	Roulette wheel	-	Repair	Up to T generation + converge

Table 4.2 (cont) Chronological list of GA studies for assembly line balancing with respect to GA specifications

GENETIC ALGORITHM SPECIFICATIONS										
Year	**Researcher(s)**	**Method**	**Initial Population& Size of population**	**Chromosome representation**	**Crossover type & probability (Pc)**	**Mutation type & probability (Pm)**	**Selection type (for mating)**	**Survival type (replacement or reproduction)**	**Feasibility Force /Repair /Penalty**	**Termination Criteria**
2009	Moon et al.	Standard	Heuristic & 2 populations popsize=100	2 chromosomes (1) task based (2) workers based (1) length=no of tasks (2) length=no of tasks no. of workstations	PMX & P_c=0.5	One cut point (1) P_m=0.3 (2) P_m= 0.4	By reordering in ascending order	-	Repair	Up to T generation + converge
2009	Kim et al.	Neighborhood GA	Heuristic popsize=100	Group number	Laszewski's heuristic & P_c=0.85	Alteration procedure P_m=0.15 & 0.05	Roulette wheel	-	Force	Up to T generation
2009	Yu &Yin	Adaptive	Heuristic	Task based & length= no. of tasks	PMX P_c=adaptive	Feasible insertion procedure P_m=adaptive	The fittest	Elitism	Repair	converge

CHAPTER FIVE

THE COMPLEX-CONSTRAINED ASSEMBLY LINE BALANCING PROBLEM

5.1 Introduction

This dissertation tackles a novel generalized ALBP, the complex-constrained assembly line balancing problem (CCALBP), introduced by Salum & Supciller (2007, 2008), Supciller & Salum (2009) and Topaloglu, Salum, & Supciller (2009). The chapter is organized as follows. CCALBP is defined in Section 5.2. In Section 5.3, the rule-base modeling of assembly constraints is discussed. In Section 5.4, its solution through constraint programming (CP) and integer programming (IP) is discussed. The context of this chapter is summarized in Section 5.5.

5.2 A Novel Line Balancing Problem: CCALBP

ALBP is the decision problem of optimally partitioning (balancing) the assembly work among the workstations (Scholl, 1999). ALBP is classified by researchers in various ways, e.g., based on the objective function (Kim et al., 1996; Scholl, 1999) and based on the problem structure (Baybars, 1986; Becker & Scholl, 2006; Scholl, 1999). Generally, ALBP is classified into two main categories: SALBP and GALBP (Baybars, 1986).

Any ALBP consists of at least three basic elements: a precedence graph which comprises all tasks and resources to be assigned, the stations which make up the line and to which those tasks are assigned, and some kind of objective to be optimized (Boysen et al., 2007).

The ordering in which tasks must be performed (technological requirements) are called precedence constraints. They are technological restrictions or physical sequencing requirements on the assembly line. Precedence constraints are generally represented graphically in the form of a precedence graph (diagram) that indicates the sequence in which the tasks must be performed. Nodes symbolize tasks, and arrows connecting the nodes indicate the precedence relations. The sequence proceeds from left to right (Groover, 2001). An example of a precedence graph is given in Figure 5.1.

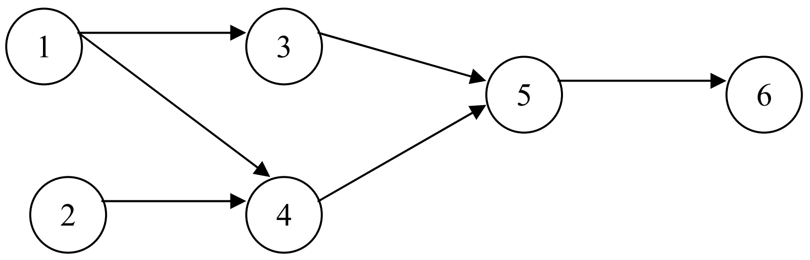


Figure 5.1 A precedence graph

There are some shortcomings of the precedence graphs. They usually fail to represent all the possible assembly sequences of a product in a single graph (Lambert, 2006), and exclude some logic statements, e.g., the precedence relation "(2 or 3) → 7" cannot be represented properly on a precedence graph (De Fazio & Whitney, 1987). Hence, they allow limited flexibility. One or more parts of a product's assembly process may admit alternative precedence sub-graphs, and because of the great difficulty of the problem and the impossibility of representing alternative sub-graphs in a precedence graph, a line designer selects, a priori, one of such alternative sub-graphs (Capacho & Pastor, 2008). According to Park et al. (1997), precedence graphs fail to describe some complicated constraints, e.g., constraints indicating that some pairs of tasks cannot be assigned into the same station because of incompatibility between them caused by some technological factors.

Despite their shortcomings, researchers continue to employ precedence graphs without questioning (Koc, Sabuncuoglu, & Erel, 2009). There are also some alternative representation methods, e.g., AND/OR graphs (Homem de Mello &

Sanderson, 1990), used in the line balancing problem. Koc et al. (2009) prove that using an AND/OR graph instead of a precedence diagram leads to better solutions of the traditional ALBPs. Capacho & Pastor (2008) employed some alternative assembly sub-graphs, in which processing times and/or precedence relations of certain tasks may vary, and solved the ALBP by simultaneously selecting an assembly sub-graph and balancing the line. Park et al. (1997) introduced two sub-problems to further consider some incompatibility constraints, range constraints, and partial precedence constraints.

ALBP has been extensively studied in the literature. However, the literature is relatively sparse in addressing alternative ways of assembling a product for ALBP, e.g., see Capacho & Pastor (2008), Capacho, Pastor, Dolgui, & Guschinskaya (2009), Koc et al. (2009), and Scholl, Becker, & Fliedner (2009) in this specific area. In other words, the literature tackles ALBP based on traditional precedence graphs in general, rather than investigating more effective modeling tools than precedence graphs to solve ALBP. This dissertation employs a well known tool, rule-bases, in modeling and solving ALBP.

Senin, Groppetti, & Wallace (2000) defined an assembly plan as a sequence of assembly operations to make a final product from a collection of individual parts. When there is more than one feasible way to combine subassemblies together, alternative assembly plans can be generated. Traditionally, ALB and determining the (near) optimum assembly plan have been considered two separate problems. Most studies consider line balancing after choosing the best plan for an assembly process. However, the overall optimal solution may not be obtained by solving these two problems separately. In other words, because pre-specifying the whole production process prior to balancing the line faces the risk of loss in efficiency (Scholl et al., 2009), alternative ways of assembling a product and their effects on task times, precedence relations and the line balance should be tackled simultaneously. In this regard, a rule-based assembly model addresses this issue. The rule-based modeling of assembly constraints will be discussed in Section 5.3.

This dissertation extends the rule-based assembly modeling (Salum & Supciller, 2007, 2008), and introduces the complex-constrained assembly line balancing problem (CCALBP) (Supciller & Salum, 2009; Topaloglu et al., 2009), which is of the general ALBPs (Baybars, 1986), in order to model all assembly constraints through a rule-base to overcome the aforementioned difficulties.

CCALBP can also be solved by various solution approaches, e.g. integer programming (IP) and constraint programming (CP) (Topaloglu et al., 2009), as discussed in Section 5.4.

ALBP is an NP-hard combinatorial optimization problem, so that the search for the optimal solution of problems in large sizes has high computational cost (Gutjahr & Nemhauser, 1964). Heuristic methods provide good results with more reasonable execution times but they do not guarantee optimal solutions. Genetic algorithms (GA) are meta-heuristics based on the mechanisms of natural selection (survival of the fittest) and genetics. Since the introduction of GAs by Holland (1975), they have been applied successfully to solve complex combinatorial problems in various research areas. Researches by Kim et al. (1996) and Ponnambalam et al. (2000) have shown that GAs improve the performance of heuristics developed for solving ALBPs. For that reason, a GA based on the rule-base is proposed by Supciller & Salum (2009) to solve CCALBP. The detailed study will be presented in Chapter 6.

5.3 Rule-based Modeling of Assembly Constraints

In this section, rule-based modeling of assembly constraints is discussed through an illustrative example (Salum & Supciller, 2007, 2008).

In practice, a precedence graph is not directly created, but derived from a table that shows precedence relations of an assembly, defined by workers carrying out the assembly process. This study also follows this convention to derive a rule-based model from such a table. Consider Table 5.1, which shows precedence relations of sewing a simple pant (jean). This table was created by workers in apparel industry.

For example, task 1, T_1, can be assigned to any workstation without any precedence constraint. T_5 can be assigned after T_9 and T_{10}, or T_3 and T_4 are assigned. T_6 can be assigned without any precedence constraint or after T_9 and T_{10} are assigned. Note that T_5, T_6, T_9, T_{10} and T_{11} involve a precedence relation that cannot be modeled through conventional precedence graphs easily. Redundancies or inconsistencies among the relations should then be discovered, which is the line designer's responsibility rather than the workers', as discussed below.

The If-then rules can then easily be derived from Table 5.1; the precedence relation of each task (in a row) is simply mapped to an If-then rule. Hence, there are as many rules as tasks with some precedence relations. Since T_1 and T_2 have no precedence relations, i.e., they are assignable initially, and T_6 can always be assigned to a workstation without considering any precedence relation, i.e., R10 in Figure 5.2a is in fact redundant, the number of the rules is nine, as defined in Figure 5.2a.

Note that such a rule-base does not grow rapidly in a harder, more realistic problem since the number of the rules is at most the number of the tasks, and the antecedent of a rule does not grow rapidly as technological flexibility is limited in practice, i.e., one does not encounter too many disjunctions in antecedents of rules in a realistic problem.

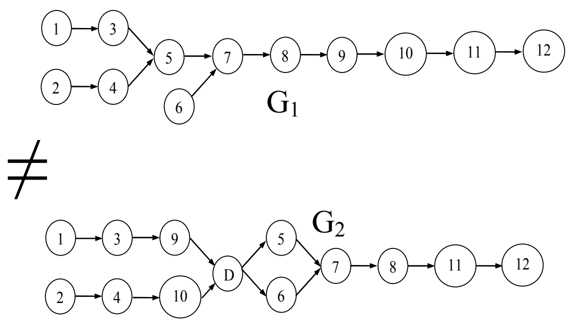
Table 5.1 Tasks of a simple pant assembly

Precedence Relation	Task T_i	Time (min)	Description
—	1	0.40	overlock stitch of parts of front right pocket
—	2	0.35	overlock stitch of parts of front left pocket
1	3	0.75	overlock stitch of front right pocket and front right part
2	4	0.80	overlock stitch of front left pocket and front left part
(9, 10) OR (3, 4)	5	0.60	overlock stitch of front left part and front right part
— OR (9, 10)	6	0.55	overlock stitch of back left part and back right part
5, 6	7	0.50	inside overlock stitch of back left part and back right part
7	8	0.45	inside overlock stitch of front left part and back left part
3 OR 8	9	0.70	outside overlock stitch of back left part and back right part
4 OR 9	10	0.60	outside overlock stitch of front left part and back left part
8 OR 10	11	0.80	stitch of waist
11	12	0.50	stitch of leg opening

One can suggest that precedence relations in Table 5.1 can also be modeled through some set of precedence graphs as in Figure 5.2b, instead of a unique rule-base as in Figure 5.2a. For example, as indicated by the sixth row in Table 5.1, T_6 has no precedence relation in G_1 in Figure 5.2b, and can be assigned after T_9 and T_{10} in G_2, while the sixth row directly corresponds to R10 in Figure 5.2a, which is a redundant rule and can be discarded from the rule base, as mentioned. However, it is difficult to derive precedence graphs if precedence relations are more complex. More importantly, it is not possible to model inclusiveness among tasks through graph-based models. That is, graphs G_1 and G_2 are mutually exclusive, although Table 5.1 indicates the inclusiveness among the tasks. In other words, Figure 5.2a, equivalent to Table 5.1, is not equivalent to Figure 5.2b. For example, if T_2, T_4 and T_{10} in G_2 are assigned to a station, then T_1, T_3 and T_5 in G_1 cannot be assigned to the next station as the tasks in G_1 and G_2 are mutually exclusive. The rule-based model in Figure 5.2a allows this assignment, as originally indicated by Table 5.1. Recall that some redundant, e.g., R10 in Figure 5.2a, or inconsistent rules may be declared by users who provide precedence relations as in Table 5.1. Therefore, some consistency check should be performed on the rule-base, which is one of the issues in rule-base modeling. Note that a dummy task, D, is required in G_2 to indicate the relevant precedence relation, which is not required in the rule-based model. In other words, each rule in Figure 5.2a directly corresponds to a row in Table 5.1.

R1: If T_1 then T_3
R2: If T_2 then T_4
R3: If (T_9 AND T_{10}) OR (T_3 AND T_4) then T_5
R4: If T_5 AND T_6 then T_7
R5: If T_7 then T_8
R6: If T_3 OR T_8 then T_9
R7: If T_4 OR T_9 then T_{10}
R8: If T_8 OR T_{10} then T_{11}
R9: If T_{11} then T_{12}
R10: If T_6 OR (T_9 AND T_{10}) then T_6

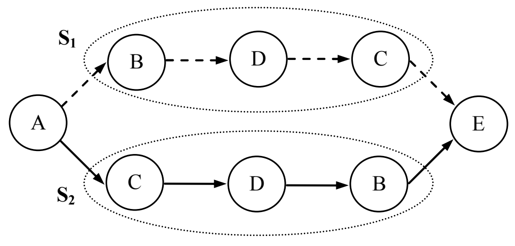
a) Rule-based model



b) Graph-based model

Figure 5.2 The rule-based and graph-based models derived by Table 5.1

More complicated relations can also be modeled easily through rule bases. For example, if a constraint indicates that certain tasks cannot be assigned into the same station (Park et al., 1997), a rule of the form "if $T_x \in S$ then $T_y \notin S$ OR if $T_y \in S$ then $T_x \notin S$" is used to mean that tasks T_x and T_y cannot be assigned into the same station, S. If some set of tasks are assembled in different sequences, e.g., see Capacho and Pastor (2008), these sequences can also be modeled easily through a rule-based model. For example, consider Figure 5.3a, where there are two alternative assembly sequences, also called assembly sub-graphs (Capacho and Pastor, 2008). Figure 5.3b gives the rules to represent the two sequences, in which XOR means exclusive OR. Note that some indices are used for the tasks to indicate the mutual exclusiveness of the sequences. This also makes it easy to consider sequence dependent task times. For example, $C_1 = 5$ and $C_2 = 6$ means that C takes 5 seconds if it is after D, and takes 6 seconds otherwise. Note also that even if the number of such sequences might be too many (n! at most), it is limited in practice due to technological constraints, e.g., the number of the sequences is two, not six in Figure 5.3a.



If A then B_1 XOR C_2
If B_1 then D_1
If D_1 then C_1
If C_2 then D_2
If D_2 then B_2
If C_1 XOR B_2 then E

a) Assembly sequences

b) The rules for the sequences

Figure 5.3 Modeling sequences through a rule-base

Consequently, the rule-based modeling is more effective than the graph-based modeling because a rule-based model can include precedence relations involving complex constraints, without the need for several precedence graphs as in Figure 5.2b. Moreover, some fuzzy rules can easily be employed in a rule-base to model vagueness in assembly constraints. As a result, CCALBP addresses a wide variety of assembly problems involving complex constraints.

As mentioned, CP is used to solve CCALBP since CP easily models logical assertions as discussed in Section 5.4.

Meta-heuristics are also commonly used solution techniques when analytic techniques like CP fail to find the optimum solution in a reasonable time. For example, if a GA is used, e.g., see Scholl & Becker (2006), the penalty term for precedence violations in the fitness function can be calculated easily, if necessary, by evaluating every rule. For the example above, if a chromosome decodes that the first station includes T_6, T_1, and T_3; the second station includes T_2, T_4, and $\boldsymbol{T_7}$; and the third $\boldsymbol{T_5}$, T_8, and T_9, then the number of the violated precedence relations (rules) is one, due to R5. That is, because T_6 and T_5 should be completed before T_7, T_7 in the second station and T_5 in the third violate R5. GA based solutions are discussed in more detail in Chapter 6.

5.4 Line Balancing through Rule-based Models and Constraint Programming

This section discusses the modeling capability of CP for CCALBPs (Topaloglu et al., 2009).

Mapping the rule-based model to the CP model is straightforward as CP can express a larger variety of constraints compared to IP such as those including logical operators $\vee$, $\Rightarrow$ and $\neq$, and global constraints that subsume a set of constraints (e.g., an “all different” relation on a set of variables replaces pairwise inequality constraints). For this reason, logic based assertions are easily modeled in CP.

For the recent years, CP has been used as an alternative solution method to IP for solving combinatorial optimization problems. An overview of CP and its main techniques can be found in Smith (1995) and Brailsford, Potts, & Smith (1999). Traditionally a CP model is composed of a set of variables (X), a set of domains (D), and a set of constraints (C) specifying which assignments of values in D to variable X are legal. The efficiency of CP lies in powerful constraint propagation algorithms which remove those values generating infeasible solutions from the domain of the variables. If constraint propagation is not sufficient to find a feasible solution, then a tree search is performed. Indeed, B&B tree developed for IP is the same as the search tree of CP in which each node represents a decision variable and each branch represents a value assignment of the variable. Typically, at each node of the search tree, the following steps are taken: first, a variable not yet fixed is selected and a remaining value of its domain is assigned to it. Then, constraint propagation occurs. If the domain of a variable becomes empty during the propagation, the solver has detected an inconsistency in the previously taken decisions, and the whole search process backtracks, typically by choosing another value for the variable. When the constraint propagation terminates while there are still some unfixed variables, the solver creates a new search node and goes on with the procedure just detailed.

The Constraint Programming Model:

Consider the line balancing problem in Figure 5.2. The proposed CP model, called CP_R, is derived from Figure 5.2a. The notation used in the formulation is as follows:

Indices & Sets

$i \in T = \{1, 2,\ldots, 12\}$ for tasks

$s \in S = \{1, 2,\ldots, n_{\max}\}$ for stations, where $n_{\max} = 5$ is set initially

S_i: set of stations to which task i can be assigned

Parameters

$c_{\max}$: upper bound on the cycle time, where $c_{\max} = 2$ minutes (the maximum time allowed at each workstation if the production rate is to be achieved)

t_i: time of task i

E_i: earliest station to which task i can be assigned

L_i: latest station to which task i can be assigned

Before a task is assigned, the total time of the tasks preceding this task must be assigned, and afterwards the total time of the tasks that follow it; as a result, the range of stations $[E_i, L_i]$ to which each task can be assigned is shown by the set S_i

$\in [E_i, L_i]$ (due to alternative precedence relations, these ranges are determined accordingly).

Variables

l_s: load of station s, i.e., the sum of the task times assigned to s

c: cycle time

n: number of stations ($n \in S$)

$I_{\{P\}} = 1$ if P is true, 0 otherwise

x_i: station number to which task i is assigned ($x_i \in S_i$)

$$\text{Min} \quad P_1 n + P_2 c \tag{5.1}$$

Subject to

$$n = \max_{i \in T}(x_i) \tag{5.2}$$

$$c = \max_{s \in \{1,2,\dots,n_{\max}\}}(l_s) \tag{5.3}$$

$$l_s = \sum_{\forall i | s \in S_i} t_i I_{\{x_i = s\}} \le c_{\max}, \forall s \in S \tag{5.4}$$

$$x_1 \le x_3 \tag{5.5}$$

$$x_2 \le x_4 \tag{5.6}$$

$$(x_9 \le x_5 \wedge x_{10} \le x_5) \vee (x_3 \le x_5 \wedge x_4 \le x_5) \tag{5.7}$$

$$x_5 \le x_7 \wedge x_6 \le x_7 \tag{5.8}$$

$$x_7 \le x_8 \tag{5.9}$$

$$x_3 \le x_9 \vee x_8 \le x_9 \tag{5.10}$$

$$x_4 \le x_{10} \vee x_9 \le x_{10} \tag{5.11}$$

$$x_{10} \le x_{11} \vee x_8 \le x_{11} \tag{5.12}$$

$$x_{11} \le x_{12} \tag{5.13}$$

The objective is to minimize the number of the stations in the first step, and the cycle time in the second. Minimizing the cycle time helps to find the best balance among the solutions that have the same number of stations. The objective function

(5.1) of CP_R employs P_1 and P_2 as the preemptive priority factors who serve only as a ranking symbol, and the ordering of objectives will be such that $P_1 >> P_2$. Thus, objective 1 is of the first priority level, and objective 2 is of the second priority level. The model is solved by optimizing the first priority objective initially. The solution obtained is added as a constraint to the original constraints and the model is solved again by optimizing the second priority objective. The description of CP_R is as follows: constraint (5.2) gives the maximum station number, thereby the number of stations required, and constraint (5.3) finds the maximum station load. Constraint (5.4) implies that the station load should not exceed the maximum time allowed. The constraints from (5.5) to (5.13) correspond to rules R1-R9, respectively, in Figure 5.2a.

The solution of CP_R is $n = 4$ and $c = 1.75$; hence the idle time, $nc - \Sigma t_i$, is zero; i.e., the balance efficiency, $(100 \times \Sigma t_i) / (n \times c)$, is 100% with $x_2 = x_4 = x_{10} = 1$, $x_1 = x_3 = x_5 = 2$, $x_6 = x_7 = x_9 = 3$, and $x_8 = x_{11} = x_{12} = 4$. In other words, the tasks are assigned to the stations as follows: (T_2, T_4, T_{10}), (T_1, T_3, T_5), (T_6, T_7, T_9) and (T_8, T_{11}, T_{12}).

The solution of the CP model of G_1 and G_2 in Figure 5.2b, denoted by CP_1 and CP_2, respectively, is given below.

Under CP_1, $n = 4$ and $c = 1.90$, hence the idle time is 0.6 (the balance efficiency is 92%) with the assignment (T_1, T_2, T_4), (T_3, T_5, T_6), (T_7, T_8, T_9) and (T_{10}, T_{11}, T_{12}). Under CP_2, $n = 4$ and $c = 1.85$, hence the idle time is 0.4 (the balance efficiency is 95%) with the assignment (T_1, T_3, T_9), (T_2, T_4, T_{10}), (T_5, T_6, T_7) and (T_8, T_{11}, T_{12}).

As CP_R outperforms CP_1 and CP_2, the rule-based modeling is effective not only in representing ALB problems with alternative precedence relations, but also in solving CP models of rule-bases, since the rule-based model contains the graph-based models, e.g., Figure 5.2a contains all the precedence constraints in G_1 and G_2 in Figure 5.2b.

Recall that more complex constraints can also be effectively modeled through rule bases. For example, assume that T_2 and T_4 cannot be assigned to the same station, i.e., If $T_2 \in S$ then $T_4 \notin S$ OR If $T_4 \in S$ then $T_2 \notin S$, and that T_2 and T_1 should be assigned to the same station, i.e., If $T_2 \in S$ then $T_1 \in S$ OR If $T_1 \in S$ then $T_2 \in S$, which are to be appended to Figure 5.2a. These constraints can easily be modeled by CP as $x_2 \neq x_4$, and $x_2 = x_1$, respectively. The assignment is then (T_1, T_2, T_3), (T_5, T_9, T_{10}), (T_4, T_6, T_7) and (T_8, T_{11}, T_{12}) with respective station times 1.5, 1.90, 1.85, and 1.75, i.e., c = 1.90. As seen, these extra constraints deteriorate the performance of CP_R.

The Integer Programming Model:

Another advantage of the rule-based modeling is that it also enables creation of an IP model, i.e., instead of a CP model, an IP model can be mapped from a rule-based model. The following gives the IP model of Figure 5.2a.

Additional Notation Required for the IP Model

$x_{i,s}$ = 1 if task i is assigned to station s, 0 otherwise

A_s = 1 if station s is required, 0 otherwise

$\delta_k \in$ [0, 1] and integer (an indicator variable for disjunctions)

$$\text{Min} \quad P_1(sA_s) + P_2 c \tag{5.14}$$

Subject to

$$\sum_{s \in S_i} x_{i,s} = 1, \forall i \in T \tag{5.15}$$

$$\sum_{\forall i | s \in S_i} t_i x_{i,s} \leq c_{\max} A_s, \forall s \in S \tag{5.16}$$

$$\sum_{\forall i | s \in S_i} t_i x_{i,s} \leq c, \forall s \in S \tag{5.17}$$

$$\sum_{s \in S_1} s x_{1,s} - \sum_{s \in S_3} s x_{3,s} \leq 0 \tag{5.18}$$

$$\sum_{s\in S_2} sx_{2,s} - \sum_{s\in S_4} sx_{4,s} \le 0 \tag{5.19}$$

$$\sum_{s\in S_9} sx_{9,s} - \sum_{s\in S_5} sx_{5,s} \le \mathrm{M}(1-\delta_1) \tag{5.20a}$$

$$\sum_{s\in S_{10}} sx_{10,s} - \sum_{s\in S_5} sx_{5,s} \le \mathrm{M}(1-\delta_2) \tag{5.20b}$$

$$\sum_{s\in S_3} sx_{3,s} - \sum_{s\in S_5} sx_{5,s} \le \mathrm{M}(1-\delta_3) \tag{5.20c}$$

$$\sum_{s\in S_4} sx_{4,s} - \sum_{s\in S_5} sx_{5,s} \le \mathrm{M}(1-\delta_4) \tag{5.20d}$$

$$\delta_1 + \delta_2 - 2\delta_5 \ge 0 \tag{5.20e}$$

$$\delta_1 + \delta_2 - \delta_5 \le 1 \tag{5.20f}$$

$$\delta_3 + \delta_4 - 2\delta_6 \ge 0 \tag{5.20g}$$

$$\delta_3 + \delta_4 - \delta_6 \le 1 \tag{5.20h}$$

$$\delta_5 + \delta_6 \ge 1 \tag{5.20i}$$

$$\sum_{s\in S_5} sx_{5,s} - \sum_{s\in S_7} sx_{7,s} \le 0 \tag{5.21a}$$

$$\sum_{s\in S_6} sx_{6,s} - \sum_{s\in S_7} sx_{7,s} \le 0 \tag{5.21b}$$

$$\sum_{s\in S_7} sx_{7,s} - \sum_{s\in S_8} sx_{8,s} \le 0 \tag{5.22}$$

$$\sum_{s\in S_3} sx_{3,s} - \sum_{s\in S_9} sx_{9,s} \le \mathrm{M}(1-\delta_7) \tag{5.23a}$$

$$\sum_{s\in S_8} sx_{8,s} - \sum_{s\in S_9} sx_{9,s} \le \mathrm{M}(1-\delta_8) \tag{5.23b}$$

$$\delta_7 + \delta_8 \ge 1 \tag{5.23c}$$

$$\sum_{s\in S_4} sx_{4,s} - \sum_{s\in S_{10}} sx_{10,s} \le \mathrm{M}(1-\delta_9) \tag{5.24a}$$

$$\sum_{s\in S_9} sx_{9,s} - \sum_{s\in S_{10}} sx_{10,s} \le \mathrm{M}(1-\delta_{10}) \tag{5.24b}$$

$$\delta_9 + \delta_{10} \ge 1 \tag{5.24c}$$

$$\sum_{s\in S_8} sx_{8,s} - \sum_{s\in S_{11}} sx_{11,s} \le \mathrm{M}(1-\delta_{11}) \tag{5.25a}$$

$$\sum_{s\in S_{10}} sx_{10,s} - \sum_{s\in S_{11}} sx_{11,s} \le \mathrm{M}(1-\delta_{12}) \tag{5.25b}$$

$$\delta_{11} + \delta_{12} \geq 1 \tag{5.25c}$$

$$\sum_{s \in S_{11}} sx_{11,s} - \sum_{s \in S_{12}} sx_{12,s} \leq 0 \tag{5.26}$$

The objective (5.14) is the same as in CP_R. Constraint (5.15) implies that every task must be assigned to only one station. Constraint (5.16) ensures that the station load in each station that is opened should be smaller than or equal to the maximum time allowed ($c_{\max}$), which is equivalent to constraint (5.4) in CP_R. Constraint (5.17) is required for the cycle time minimization. The constraints from (5.18) to (5.26) correspond to rules in Figure 5.2a. For example, constraints from (5.20a) through (5.20i) are developed only to model Rule 3. Here M can be taken as the maximum station number $n_{\max}$.

The IP model gives the same solution, n = 4 and c = 1.75, with that of CP_R, but contains more variables. Also, creation of IP models can be more difficult if the rules are more complex. Note that the correspondence between CP_R and Figure 5.2a is clearer than that of IP and Figure 5.2a because of the modeling capability of CP. Thus CP_R can be comprehended more easily.

The models are solved using ILOG OPL Studio 3.7 (2003) on a 1.8 GHz CPU, 3.5 GB memory PC. It provides access to ILOG CPLEX 9.0 (ILOG, 2005a) and ILOG Solver 6.0 (ILOG, 2005b) for solving the IP and CP models respectively. Computing time of 2000 CPU seconds is set. Since the problem is small, solution times of IP and CP are also small. However, a computational experiment should also be carried out to analyze the performances of CP and IP models with respect to modeling capability, solution quality and time; but this comparison is beyond the scope of this chapter.

5.5 Chapter Summary

The major drawback of precedence graphs is that they are not suitable to model complex assembly constraints. This dissertation introduced CCALBP to address this issue. It was shown how to model all assembly constraints through the well known If-then rules, and how to solve the problem through CP and IP models mapped from the rule-based model. It was also shown how to map a rule-based model to a CP or an IP model. This mapping can also be automated, which enables users to easily create the models. On the other hand, this mapping was not possible from graph-based models that address, though roughly, complex assembly constraints. Thus, CCALBP can be solved *only* through rule-based modeling, but not graph-based modeling. Some fuzzy rules can also be employed in a rule-base to model vagueness in assembly constraints. Rule-based and graph-based models were compared in terms of modeling capability. A GA based on the rule-base can also be developed to solve the CCALBP. The proposed GA will be discussed in detail in the next chapter.

CHAPTER SIX

A GENETIC ALGORITHM BASED APPROACH FOR SOLVING THE COMPLEX-CONSTRAINED ASSEMBLY LINE BALANCING PROBLEM

6.1 Introduction

This dissertation employs a well known tool, rule-bases, in modeling and solving ALBP for the first time and extends this literature in terms of modeling scope for assembly constraints in line balancing. This study extends the rule-based modeling of assembly constraints and introduces CCALBP (Salum & Supciller, 2007, 2008; Topaloglu et al., 2009). Its main advantage lies in its ability to simultaneously model the alternative ways of assembling a product. For the comprehension of CCALBP and to describe the rule-based modeling, the problem is modeled and solved by IP and CP in the last chapter. Due to the NP-hard nature of CCALBP, the use of a mathematical programming model to optimally solve CCALBP in large sizes has high computational cost. Therefore, heuristic or meta-heuristic procedures need to be developed. In order to search the solution space efficiently and to provide good solutions with more reasonable computation times, CCALBP is solved through GAs (Supciller & Salum, 2009). The main contribution of this dissertation is the integration of the rule-base approach for modeling assembly constraints through a GA solution.

In this chapter, a GA based on the rule-base is proposed to solve CCALBP. In Section 6.2, the proposed GA is discussed in detail. The specific characteristics of the proposed GA are devised with the inspiration taken from the current examples in the literature. These characteristics are explained on an example problem of sewing a simple pant. The control parameters of the GA are optimized to improve the performance in Section 6.3. In Section 6.4, the computational experiments are carried out on a set of generated problems by adapting the case problems in the literature. Finally, in section 6.5, the context of this chapter is summarized.

6.2 Line Balancing through Rule-based Models and GA

In this section, a GA based on the rule-base is proposed to solve CCALBP. The general GA specifications and details of objective function integrated with the proposed rule-base are given in the following sections.

To explain how the proposed GA works, the example given in Chapter 5 is used in this section. Table 6.1 shows precedence relations of this example.

Table 6.1 Tasks of a simple pant assembly and its rule base

Precedence Relation	Task T_i	Time (min)
—	1	0.40
—	2	0.35
1	3	0.75
2	4	0.80
(9, 10) OR (3, 4)	5	0.60
— OR (9, 10)	6	0.55
5, 6	7	0.50
7	8	0.45
3 OR 8	9	0.70
4 OR 9	10	0.60
8 OR 10	11	0.80
11	12	0.50

R1: If T_1 then T_3
R2: If T_2 then T_4
R3: If (T_9 AND T_{10}) OR (T_3 AND T_4) then T_5
R4: If T_6 OR (T_9 AND T_{10}) then T_6
R5: If T_5 AND T_6 then T_7
R6: If T_7 then T_8
R7: If T_3 OR T_8 then T_9
R8: If T_4 OR T_9 then T_{10}
R9: If T_8 OR T_{10} then T_{11}
R10: If T_{11} then T_{12}

In the proposed GA, the rule-base is represented by a matrix for coding in Matlab. The rule-base in Table 6.1 is given as a matrix in Figure 6.1, where each column represents a rule, e.g., the 3rd column is R3, i.e., IF (TASK 9 AND TASK 10) OR (TASK 3 AND TASK 4) THEN TASK 5.

R1	R2	**R3**	R4	R5	R6	R7	R8	R9	R10
3	4	**5**	6	7	8	9	10	11	12
1	2	**9 10**	6 0	5 6	7	3	4	10	11
0	0	**3 4**	9 10	0 0	0	8	9	8	0

Figure 6.1 Matrix representation of the rule-base in Table 6.1

6.2.1 Representation

Sequence-oriented representation, which is a kind of an order-based representation, is used for genetic representation because of its advantages. Especially, it can handle all types of ALB problems and provides flexibility in choosing genetic operators (Kim et al., 1996).

Each task is represented by a number that is placed on a string of numbers called chromosomes. The length of a chromosome is the number of the tasks. All tasks are sequentially listed in the order of their assignment to work stations.

For example, a chromosome for a 12 task problem is given below:

Tasks: 2 4 3   1 8 12   9 6 5   11 7 10, which indicates the following assignments.

Tasks of 1st station: 2, 4, 3
Tasks of 2nd station: 1, 8, 12
Tasks of 3rd station: 9, 6, 5
Tasks of 4th station: 11, 7, 10

6.2.2. Initialization

The initial population may be generated randomly or with the help of some heuristics (Sivanandam & Deepa, 2008). In this study, it is generated randomly for the small sized problems. Infeasible solutions, which violate some of the precedence constraints, are allowed in the population.

For the problems which have 45 and more tasks, largest candidate rule (Groover, 2001) is used to generate the first individual of the population. In the method of largest candidate rule, tasks are arranged in descending order according to their task times. Then they are assigned to a station by starting at the top of the list and selecting the first one which satisfies precedence relations and does not cause the total sum of the task times at that station to exceed the allowable cycle time (Groover, 2001). The rest of the population is generated randomly. Infeasible solutions are allowed in the population.

6.2.3 The Fitness Function

The objective of the example problem can be considered to minimize the number of work stations, subject to a given cycle time. On the other hand, one of the solutions with the same number of stations may be "better balanced" than the others. For example, an assembly line of three stations with the station times 30-50-40 is considered to be better balanced than the one with the times 50-50-20. Hence, a fitness function that consists of two objectives should be used; one minimizes the number of stations and the other obtains balanced station (Sabuncuoglu et al., 2000).

An infeasible solution in ALB problems is defined as the violation of some precedence relations. A population of feasible solutions may lead to a fragmented search space, which increases the probability of being trapped in local minima. Therefore, infeasible solutions are also allowed in a population as genetic operators can lead to feasible solutions from an infeasible population (Suresh et al., 1996).

When infeasible solutions are allowed in the population, the population is forced to feasibility by assigning high penalty costs to infeasible solutions as discussed in Anderson & Ferris (1994), Ruijun, Dingfang, Yong, Zhonghua, & Xinxin (2007) and Guo et al. (2008). This strategy increases the amount of variability in the population (Tasan & Tunali, 2008).

A simple method to penalize infeasible solutions is to apply a constant penalty to those solutions that violate feasibility in any way. The constrained problem is transformed into an unconstrained problem by penalizing infeasible solutions. The penalized objective function is then the sum of the unpenalized objective function and a penalty (for a minimization problem). A penalty term is added to the objective function for any violation of the constraints (Anderson & Ferris, 1994; Gen & Cheng, 1997; Michalewicz & Schoenauer, 1996). The penalty function with m constraints is then represented as below (for a minimization problem):

$$f_p(x) = f(x) + \sum_{i=1}^{m} C_i \delta_i \tag{6.1}$$

where $\begin{cases} \delta_i = 1, \text{ if constraint } i \text{ is violated} \\ \delta_i = 0, \text{ if constraint } i \text{ is satisfied} \end{cases}$

$f_p(x)$ is the penalized objective function, $f(x)$ is the unpenalized objective function and C_i is the constant imposed for the violation of the constraint (Smith & Coit, 1997).

In the solution, the fitness function combines the two objectives, i.e. minimizing the number of stations and finding the best balance among the solutions that have the same number of stations, and includes a penalty cost as described by Cilkin (2003):

$$\text{Fitness Function} = 2000 * N_v + 0.2\sqrt{\frac{\sum_{k=1}^{n}\left(S_{\max} - S_k\right)^2}{n}} + \frac{\sum_{k=1}^{n}\left(S_{\max} - S_k\right)}{n} \tag{6.2}$$

where N_v is the number of precedence violations, n is the number of stations, S_{max} is the maximum station time, and S_k is the k^{th} station time.

The maximum of the coefficients is 2000, and it is given to the number of precedence violations to force the algorithm to the feasible solutions in a faster way. The second and the third parts of the fitness function are used and explained by Leu et al. (1994) and Sabuncuoglu et al. (2000). The second part of the fitness function is taken to be the minimization of the mean-squared idle time. This part aims to find the best balance among the solutions that have the same number of stations (Leu et al., 1994). The minimum coefficient, 0.2, is given to the second part. The third part is to minimize mean idle time. The third part is assumed arbitrarily more critical than the second one. According to Sabuncuoglu et al. (2000), this part only minimizes the number of stations (Scholl & Becker, 2006). The coefficients of the fitness function are determined by experimenting with different values (tuning). A smaller fitness function means fewer workstations and more balanced workload between the workstations.

The second part of the fitness function is only used for feasible solutions. When there is a violation of precedence constraints, the work balance among the stations cannot be computed. The calculation of the fitness function in an infeasible solution is explained below.

<u>The calculation of the fitness function in an infeasible solution</u>

If the example problem given in Table 6.1 is considered, a chromosome in the population may be as in Figure 6.2. Let the order of the tasks be as below for an infeasible solution, where the cycle time, *C*, is 2.

2 4 3 1 8 12 9 6 5 11 7 10

Figure 6.2 The order of the tasks for an infeasible solution

1. <u>Assignment of tasks to the stations</u>:

The tasks are then assigned to the stations sequentially subject to the cycle time constraint. The assignment of tasks in Figure 6.2 yields Table 6.2.

Table 6.2 Assignment of tasks to the stations

Tasks	**Task time (min)**	**Station time (min)**	**Station**
2	0.35		
4	0.80		
3	0.75	1.90	1
1	0.40		
8	0.45		
12	0.50	1.35	2
9	0.70		
6	0.55		
5	0.60	1.85	3
11	0.80		
7	0.50		
10	0.60	1.90	4

2. <u>Calculating the number of precedence violations</u>.

There are then three violations in Table 6.2, i.e., N_v is 3.

1. Violation

T_1 is assigned to S_2 and T_3 is assigned to S_1, which violates R1, i.e.,

IF TASK 1 THEN TASK 3

2. Violation

T_7 is assigned to S_4 and T_8 is assigned to S_2, which violates R6, i.e.,

IF TASK 7 THEN TASK 8

3. Violation

T_{11} is assigned to S_4 and T_{12} is assigned to S_2, which violates R10, i.e.,
IF TASK 11 THEN TASK 12

3. Calculating the fitness function.

In this order of the example chromosome given in Figure 6.2, since there are some violations of precedence constraints, the second part of the fitness function which gives the work balance among the stations cannot be computed. Then the fitness function is computed according to Equation 6.2 without the second part as follows:

$$\text{Fitness Function} = 2000 \times N_v + 0.2\sqrt{\frac{\sum_{k=1}^{n}(S_{\max} - S_k)^2}{n}} + \frac{\sum_{k=1}^{n}(S_{\max} - S_k)}{n} \quad (6.2)$$

$$\text{Fitness Function} = 2000 \times 3 + \frac{(1.90-1.90)+(1.90-1.35)+(1.90-1.85)+(1.90-1.90)}{4}$$

$$\text{Fitness Function} = 2000 \times 3 + 0.15$$

$$\text{Fitness Function} = 6000.15$$

6.2.4 Selection

The selection process in genetic algorithms is based on the natural law of survival of the fittest. It is the process to determine which chromosomes are selected for the next generation in terms of their fitness (Mitchell, 1996).

In this study, the tournament selection is used. In its simplest form, tournament selection consists of picking two members of the population randomly, and then selecting the best one as a parent. After two parents are selected this way, the genetic operators take place as usual (Mitchell, 1996).

The procedure works as follows:

Step 1. A tournament size, *m*, is set.

Step 2. Randomly *m* individuals are selected from the population.

Step 3. With probability *r*, the best of the *m* individuals is selected and with probability *1-r*, a random individual among the other *m-1* is selected. *r* is referred to as the tournament selection parameter.

Two individuals are chosen at random from the population. A random number is then chosen between 0 and 1. If it is smaller than the tournament selection parameter, the fitter of the two individuals is selected to be a parent; otherwise the less fit one is selected. The two are then returned to the original population and can be selected again. Goldberg & Deb (1991) presented an analysis of this method.

In this study, the real world tournament selection is used (Lee, Soak, Kim, Park, & Jeon, 2008). Two groups of 8 ($8 = 2^3$) individuals are selected randomly from the same tournament level. In each group, each individual is sequentially paired with a neighbor from the same group. When all competitions in the present tournament level are completed, only the winners go on to the next tournament level. The competitions are completed on three levels. The winners of each group are selected this way and get ready for crossover.

6.2.5 Genetic Operators

To improve the adaptability of the population, two basic operators, crossover and mutation, are used to modify the chromosome. Crossover is the operation by which two parents in the current population create offspring for the next population. The mutation operator is used to randomly change the value of single genes within chromosomes. The *two-point order crossover* and *reciprocal exchange mutation* are used as genetic operators in the proposed genetic algorithm.

The two-point order crossover is the combination of the two-point crossover and the *order crossover* (OX) which was proposed by Davis (1985). The two-point order crossover randomly chooses two crossover points. The crossover operator copies the chromosome part between the crossover points of the two parents to the respective child chromosome while preserving the relative order of the sequence indicated by the other parent. There are different versions of the two-point order crossover (Ishibuchi & Murata, 1998; Murata & Ishibuchi, 1996).

The two-point order crossover procedure works as follows (Gen & Cheng, 1997):
Step 1. A substring between two crossover points is selected at random.
Step 2. A proto-child is produced by copying the selected substring into the corresponding positions.
Step 3. The tasks which are already in the substring are deleted from the second parent. The resulted sequence of tasks contains the tasks which the proto-child needs.
Step 4. The tasks are placed into the unfixed positions of the proto-child from left to right according to the order of the sequence to produce an offspring.

The procedure is illustrated in Figure 6.3. The second offspring can be produced with the same steps as [2 5 4 9 1 3 6 7 8 10 11 12] from the same parents. The rate of the crossover operation is defined by R_c.

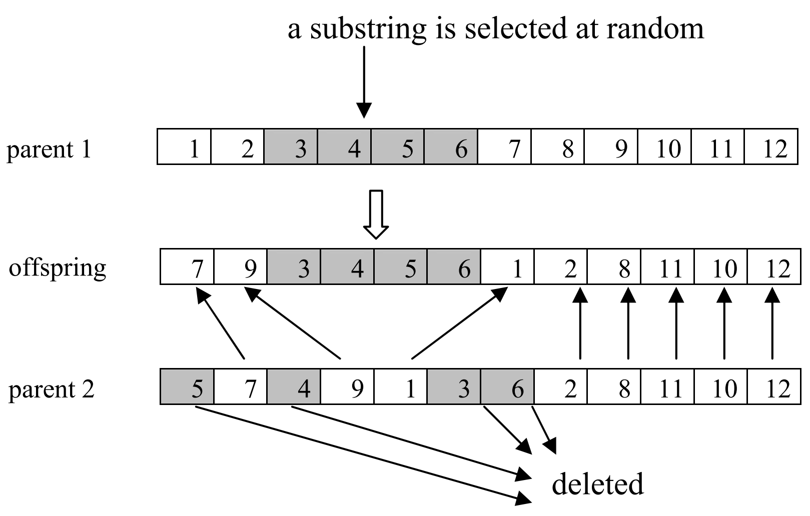


Figure 6.3 Illustration of the two-point crossover operator

In the reciprocal exchange mutation, two positions are selected at random and then the tasks are swapped on these positions (Gen & Cheng, 1997). The procedure is illustrated in Figure 6.4. The rate of the mutation operation is defined by R_m.

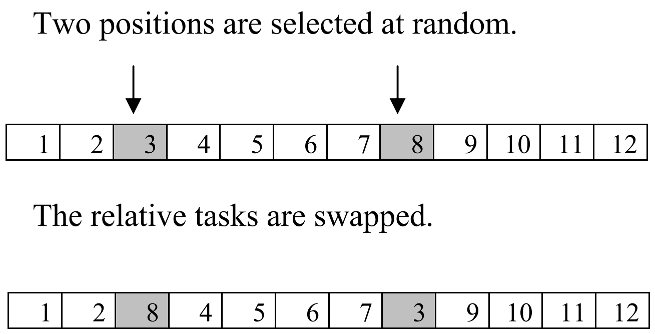

Two positions are selected at random.
1 2 3 4 5 6 7 8 9 10 11 12
The relative tasks are swapped.
1 2 8 4 5 6 7 3 9 10 11 12

Figure 6.4 Illustration of the reciprocal exchange mutation operator

6.2.6 Elitism

Survival is an essential process in GAs that removes individuals with a low fitness and drives the population towards better solutions. A part of the existing population survives to the next generation and forms a new population in the next generation. To ensure that the best solution of the previous generation is always present in the next population, a procedure known as *elitism* is used (Sivanandam & Deepa, 2008). In this study, two copies of the best individual are made. Then, the first two new individuals of the next generation are taken as exact copies of the best individual in the first generation.

6.2.7 Termination

There are many stopping conditions in GAs (Sivanandam & Deepa, 2008). While searching the solution space of the problem, the procedure can be stopped when one of the following is achieved (i) the fitness function of the best solution does not improve after a predetermined number of generations, e.g., T_G=100, or (ii) the total number of generations exceeds a maximum number, e.g. T_{max}=1,000.

In the proposed GA, the procedure stops when the fitness function is zero, or the total number of generations exceeds 1,000 generations.

6.2.8 Results of the Proposed GA

The proposed GA is coded in Matlab 7.0, and run three times for 1,000 generations for the example problem given in Table 6.1 in Section 6.2. The fitness value for each generation up to the end of the procedure for one of the runs is given in Figure 6.5. The minimum fitness value, zero, is reached at the 7th generation.

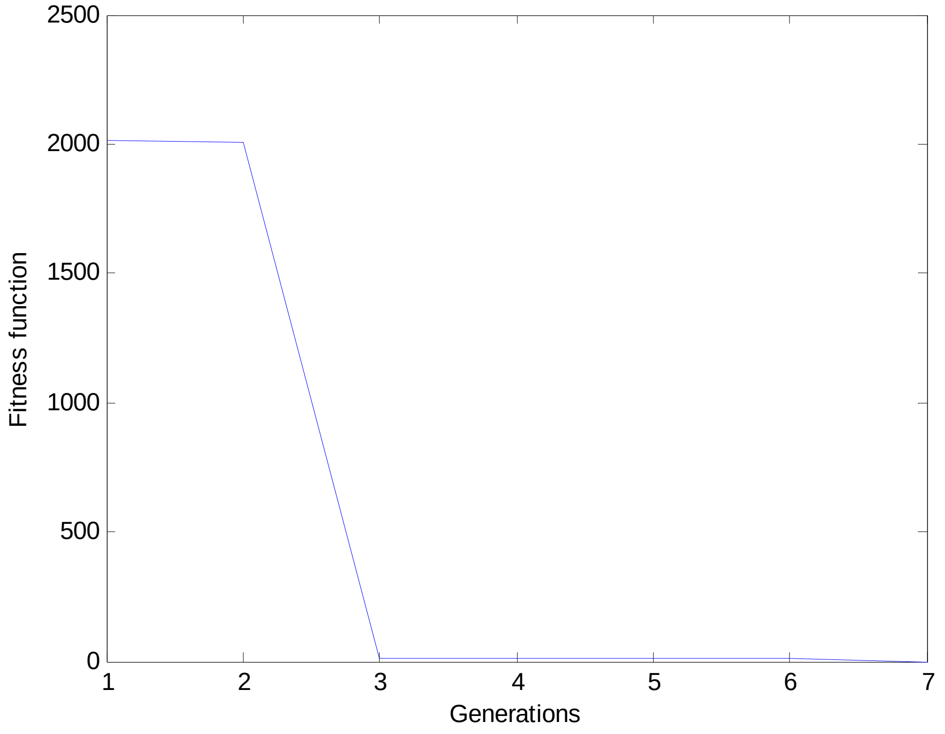


Figure 6.5 The fitness value versus the number of generations for the example problem

Table 6.3 shows the assignment for the example problem given in Table 6.1 solved by the proposed GA. The assignment has no violation of any precedence constraint.

Table 6.3 The result for the example problem

Tasks	**Task time (min)**	**Station time (min)**	**Station**
10	0.60		
4	0.80		
2	0.35	1.75	1
1	0.40		
3	0.75		
5	0.60	1.75	2
9	0.70		
6	0.55		
7	0.50	1.75	3
11	0.80		
8	0.45		
12	0.50	1.75	4

6.3 Parameter Optimization

The performance of GA depends on several parameters. The effects of three parameters considered as significant are studied for the performance of the proposed GA: the population size, the crossover rate and the mutation rate. Statistical design of experiments (Montgomery, 2001) is used to optimize the three parameters in Table 6.4.

Table 6.4 Levels of control parameters

Control Parameters			
Levels	**Population size**	**Crossover rate**	**Mutation rate**
1	100	0.50	0.05
2	500	0.70	0.10
3	1000	0.90	0.25

As a test problem, Mitchell's problem (Scholl, 1993) with 21 tasks from the literature is chosen for identifying the effect of different control parameters.

Since there are three distinct parameters with three levels, 3^3 full factorial experimental design given in Table 6.5 is used to detect the possible interactions of factor effects and to determine the optimal parameter setting.

For each design point, 5 independent GA runs are performed to determine the variations in the results, i.e., $135 = 3^3 \times 5$ runs are carried out.

Table 6.5 The 3^3 full factorial experimental design layout

Experiment no	**Population size**	**Crossover rate**	**Mutation rate**
1	100	0,50	0,05
2	100	0,70	0,05
2	100	0,70	0,05
3	100	0,90	0,05
4	100	0,50	0,10
5	100	0,70	0,10
6	100	0,90	0,10
7	100	0,50	0,25
8	100	0,70	0,25
9	100	0,90	0,25
10	500	0,50	0,05
11	500	0,70	0,05
12	500	0,90	0,05
13	500	0,50	0,10
14	500	0,70	0,10
15	500	0,90	0,10
16	500	0,50	0,25
17	500	0,70	0,25
18	500	0,90	0,25
19	1000	0,50	0,05
20	1000	0,70	0,05
21	1000	0,90	0,05
22	1000	0,50	0,10
23	1000	0,70	0,10
24	1000	0,90	0,10
25	1000	0,50	0,25
26	1000	0,70	0,25
27	1000	0,90	0,25

The scatter plots of fitness and computation time are given in Figure 6.6 and 6.7, respectively. As seen in the diagrams, the computation time increases with the size of the population.

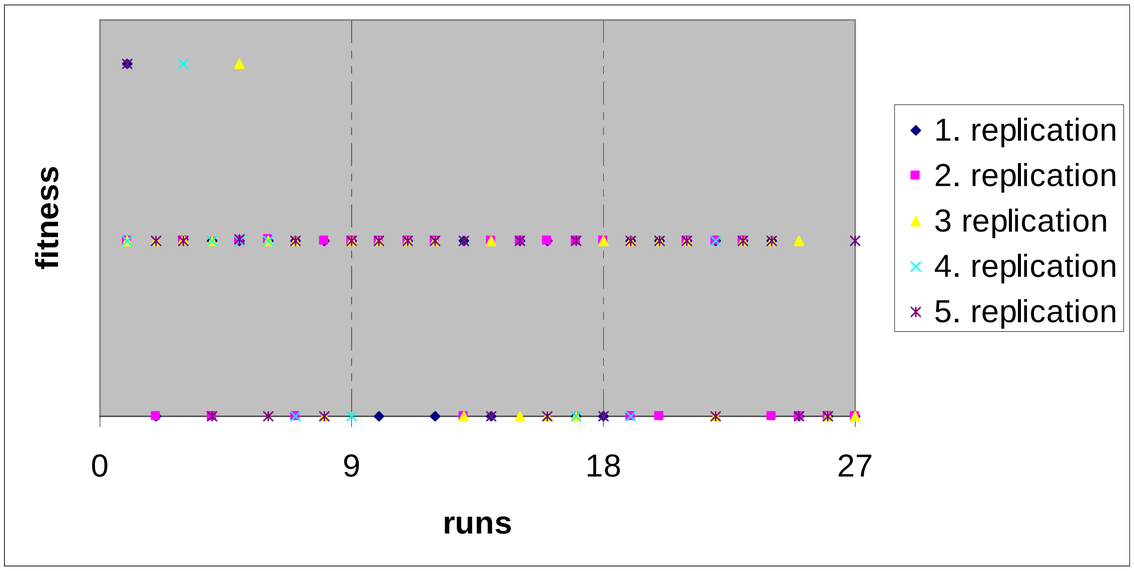


Figure 6.6 The scatter plot of fitness

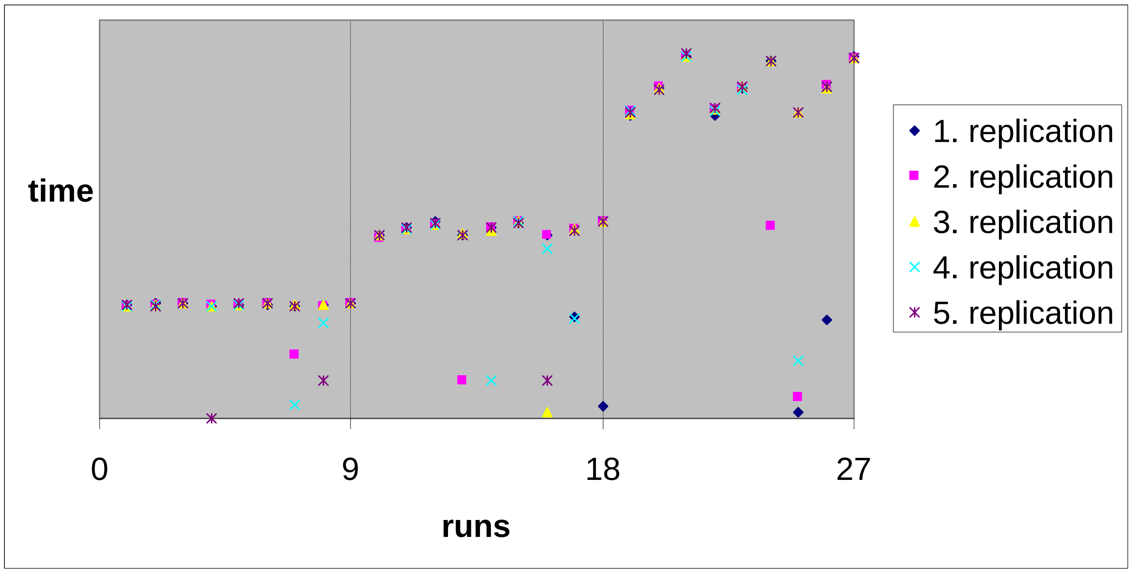


Figure 6.7 The scatter plot of computation time

To determine the significance of each parameter effect on the fitness, analysis of variance (ANOVA) is used. Given level of significance is equal to 0.05. According to Table 6.6, the values of “Prob>F” less than 0.05 indicate model terms are significant. The significant main factors with respect to the fitness response are the population size and the mutation rate. The crossover rate has an insignificant main effect. The interactions are also insignificant.

Table 6.6 ANOVA results for fitness values

Analysis of variance table for fitness			
Source	**F-value**	**P-value (Prob>F)**	
Model	2.35	0.0012	significant
A - crossover rate	1.22	0.2984	
B - mutation rate	14.14	0.0001	significant
C - population size	3.77	0.0262	significant
AB	1.02	0.4015	
AC	0.41	0.8033	
BC	0.81	0.5202	
ABC	1.73	0.0996	

From the ANOVA analysis, the main conclusions can be summarized in the main effects plot for fitness in Figure 6.8, and in the main effects plot for computation time in Figure 6.9. According to Figure 6.8, the fitness decreases when the population size and mutation rate is high.

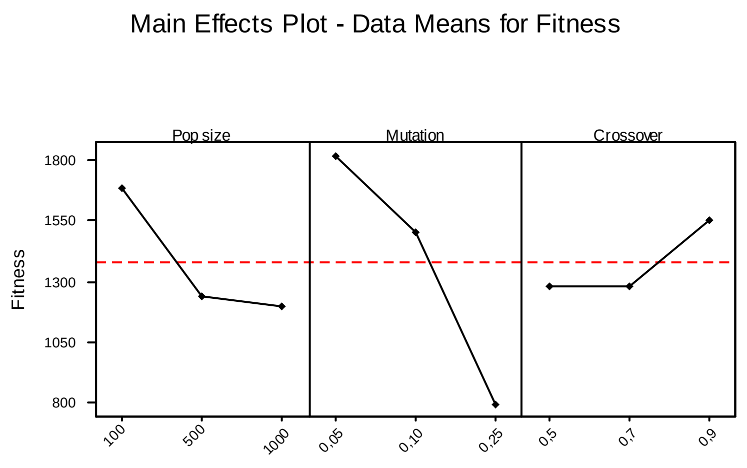


Figure 6.8 Main effects plot for fitness

In terms of the significance of each parameter effect on computation time, the significant main factors with respect to computation time response are the population size, the mutation rate and the crossover rate according to Table 6.7.

Table 6.7 ANOVA results for computation time

Analysis of variance table for computation time			
Source	**F-value**	**P-value (Prob>F)**	
Model	16.81	<0.0001	significant
A - crossover rate	11.04	<0.0001	significant
B - mutation rate	8.90	0.0003	significant
C - population size	178.04	<0.0001	significant
AB	2.68	0.0354	significant
AC	2.66	0.0367	significant
BC	1.36	0.2512	
ABC	1.79	0.0869	

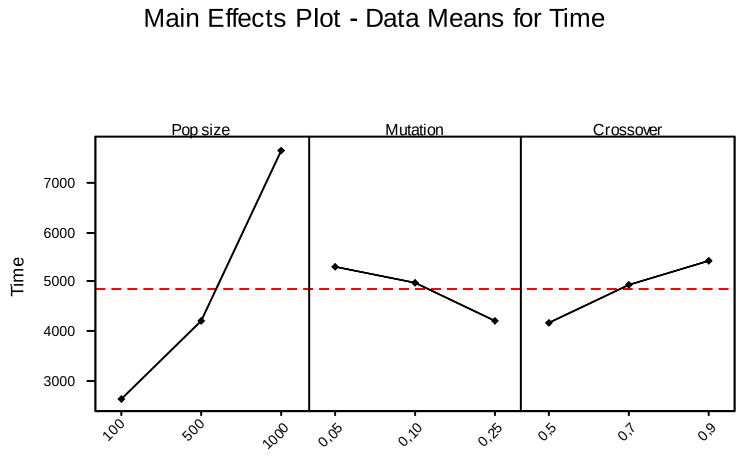


Figure 6.9 Main effects plot for computation time

As a result, even though the crossover rate is insignificant for the fitness response, it has a great effect on computation time. As seen in Figure 6.9, the computation time increases with the population size. Yet according to Figure 6.8, the fitness decreases with the high level of population size. Since the fitness is more important than the computation time, high level of population size can be used. The optimal levels of the three parameters can then be determined as 1,000 for population size, 0.50 for crossover rate, and 0.25 for mutation rate for the small sized problems. For the problems which have 45 tasks or more, crossover and mutation rates can be different.

6.4 Computational Experiments

Since CCALBP is a novel problem, there is no set of benchmark instances for testing. Therefore, self-made instances are generated by adapting well-known benchmark problems from the literature whose descriptions are given in Scholl (1993). Some alternative routes are created and added to these literature problems.

6.4.1 The Instances Generated from the Example Problem

For the example problem given in Table 6.1 in Section 6.2, three instances are generated considering different alternatives. New tasks are added to the problem for the alternatives, and the task times of the Mitchell's problem (Scholl, 1993) from the literature are used. The cycle time, C, is taken as 14. The alternatives of the example problem according to their complexities are given in Table 6.8.

Table 6.8 The instances generated for the example problem

Problems	**The number of ORs in one rule**	**Number of tasks**	**The number of ORs in all rules**
Example Problem 1	1	12	5
Example Problem 2	2	13	10
Example Problem 3	3	15	15

The problems and the results are given in the first three solutions below.

<u>1. Example: Problem 1</u>

It has one OR in one rule, 12 tasks and five ORs in total as given in Figure 6.10.

3	4	5		6		7		8	9	10	11	12
1	2	3	4	6	0	5	6	7	3	4	8	11
0	0	9	10	9	10	0	0	0	8	9	10	0

Figure 6.10 The matrix for the rule-base of example problem 1

The assignment of the tasks is given in Figure 6.11:

1 3 6 9 10 5 2 11 12 7 4 8

Figure 6.11 The assignment of the tasks for example problem 1

The fitness value is 1.5098.

<u>2. Example: Problem 2</u>

It has two ORs in one rule, 13 tasks and 10 ORs in total as given in Figure 6.12. The number of two ORs in all rules is three.

3	4	5		6		7		8	9	10	11	12	13
1	2	3	4	6	0	5	6	7	3	4	3	4	10
0	0	9	10	9	10	0	0	0	8	9	8	11	12
0	0	11	12	11	12	0	0	0	0	0	10	0	0

Figure 6.12 The matrix for the rule-base of example problem 2

The assignment of the tasks is given in Figure 6.13:

3 1 9 2 11 10 12 5 6 7 4 13 8

Figure 6.13 The assignment of the tasks for example problem 2

The fitness value is 0.2894.

3. Example: Problem 3

It has three ORs in one rule, 15 tasks and 15 ORs in total as given in Figure 6.14. The number of three ORs in all rules is two.

3	4	5		6		7		8	9	10	11	12	13	14	15
1	2	3	4	6	0	5	6	7	3	4	3	4	3	4	14
0	0	9	10	9	10	0	0	0	8	9	8	11	10	13	12
0	0	11	12	11	12	0	0	0	0	0	10	0	12	0	0
0	0	13	14	13	14	0	0	0	0	0	0	0	0	0	0

Figure 6.14 The matrix for the rule-base of example problem 3

The assignment of the tasks is given in Figure 6.15:

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

Figure 6.15 The assignment of the tasks for example problem 3

The fitness value is zero.

6.4.2 The Instances Generated from the Literature Problems

For the other instances, 10 well-known problems are selected whose descriptions are given in Scholl (1993). Table 6.9 shows the data for the benchmark problems, where the first and the second columns give the name of the problem and the number of tasks (n), respectively. The minimum and maximum cycle times are shown in the third and fourth columns. The fifth and sixth columns contain the minimum ($t_{\min}$) and maximum task times ($t_{\max}$) in each problem, respectively. In the seventh column, the sum of the task times for each problem is given. The eighth column gives the order strength ($OS = [number\ of\ all\ precedence\ relations / (n \times (n-1))]$), an indicator for complexity of problem instances (Scholl, 1999). The ninth column shows time variability ratio ($TV = t_{\max} / t_{\min}$) that measures the range of variation for the task times (Scholl, 1999).

Table 6.9 Data sets

Problem	**# of tasks**	**Min. cycle time**	**Max. cycle time**	**Min. task time**	**Max. task time**	**Sum of task times**	**Order strength**	**Time variability ratio**
Bowman	8	20	20	3	17	75	75.00	5.67
Jaeschke	9	6	18	1	6	37	83.33	6.00
Jackson	11	7	21	1	7	46	58.18	7.00
Mitchell	21	14	39	1	13	105	70.95	13.00
Roszieg	25	14	32	1	13	125	71.67	13.00
Heskiaoff	28	138	342	1	108	1024	22.49	108.00
Buxey	29	27	54	1	25	324	50.74	25.00
Sawyer	30	25	75	1	25	324	44.83	25.00
Kilbridge	45	56	184	3	55	552	44.55	18.33
Arcus1	83	3786	10816	233	3691	75707	59.09	15.84

Since CCALBP is a novel problem, the problem instances are generated by using the problems in Table 6.9. The original precedence relations and operation times are preserved, but new alternatives are added. Additionally, a real case problem from the apparel industry with 68 tasks is also used for experiments. For each problem, instances with one OR, two ORs and three ORs are generated, respectively. The rule-base for each problem instance is given in the appendices.

A brief computational experiment is carried out by using the given problems with different cycle time values. A total number of 208 problem instances are solved. For small sized problems, 24 problem instances are considered, having the number of tasks from 8 to 11, and with only one alternative (one OR) with different cycle time values. For medium sized problems, 148 problem instances are considered, having the number of tasks from 21 to 45, and with up to three ORs with different cycle time values. For large sized problems, 36 problem instances are considered, having the number of tasks from 68 to 83, and with up to three ORs with different cycle time values. For each problem, instances with one OR, two ORs and three ORs are solved by the proposed GA with the same parameters. The proposed GA is run three times for each instance.

The best results of the problem instances are detailed in Table 6.10. In the first column of Table 6.10, the problem source is reported. The second column reports the number of tasks. The third column lists the cycle time values. The fourth column reports the minimum number of stations found in the literature for the original problems given in Table 6.9. The other columns report the best solutions of the generated instances with and without alternatives solved by the proposed GA. The number of rules with logical ORs is also given in Table 6.10 to evaluate the effects of alternatives on the fitness function.

The computational experiments show that optimal solutions can only be obtained for small sized problem instances in a reasonable amount of time. Based on Table 6.10, the proposed GA is proven to perform better when new alternatives are added. As it can be seen in Table 6.10, a new alternative is added to the original problem in each step. The GA with the rule base solves each type of the problem with new alternatives simultaneously as well as the original problem. Table 6.10 shows that the fitness values are getting smaller while the number of alternatives is increasing. The fitness value consists of two objectives, minimizing the number of stations and obtaining balanced stations. When more alternatives are added, better balanced stations are obtained. As the number of ORs is increased, line balancing improves.

It should be noted that the minimum number of stations reported in the literature is 6 for Roszieg problem with the cycle time of 25 time units. When more alternatives are added, the number of stations decreases to 5, which is the lower bound for the cycle time of 25 time units. The same is also reported for the cycle time of 18.

Table 6.10 The results of the experiments as number of stations and fitness

Problem	**# of tasks**	**Cycle time**	**Optimal # of stations**	**No alternatives**			**With alternatives**											
								1 OR			**2 ORs**			**3 ORs**				
				# of all rules	**Min. # of stations**	**Fitness**	**# of all rules**	**# of rules with 1 OR**	**Min. # of stations**	**Fitness**	**# of rules with 2 ORs**	**Min. # of stations**	**Fitness**	**# of rules with 3 ORs**	**Min. # of stations**	**Fitness**		
Bowman	8	20	5	7	5	2.5933	7	2	5	2.645	-	-	-	-	-	-		
Jaeschke	9	6	8	8	8	1.699	8	5	8	1.6832	-	-	-	-	-	-		
		7	7		7	2.0997			8	1.6832	-	-	-	-	-	-		
		8	6		7	2.0997			8	1.699	-	-	-	-	-	-		
		10	4		4	0.9232			4	0.9232	-	-	-	-	-	-		
		18	3		3	5.9103			3	2.1428	-	-	-	-	-	-		
Jackson	11	7	8	10	8	1.5662	10	2	7	0.5976	-	-	-	-	-	-		
		9	6		6	1.6388			6	1.6388	-	-	-	-	-	-		
		10	5		5	1.0191			5	0.9789	-	-	-	-	-	-		
		13	4		4	0.6414			4	0.6414	-	-	-	-	-	-		
		14	4		4	0.6414			4	0.6414	-	-	-	-	-	-		
		21	3		3	0.83			3	0.83	-	-	-	-	-	-		

Table 6.10 (cont) The results of the experiments as number of stations and fitness

Problem	**# of tasks**	**Cycle time**	**Optimal # of stations**	**No alternatives**			**With alternatives**									
								1 OR			**2 ORs**			**3 ORs**		
				# of all rules	**Min. # of stations**	**Fitness**	**# of all rules**	**# of rules with 1 OR**	**Min. # of stations**	**Fitness**	**# of rules with 2 ORs**	**Min. # of stations**	**Fitness**	**# of rules with 3 ORs**	**Min. # of stations**	**Fitness**
Mitchell	21	14	8	20	9	2.8708	20	2	8	1.0871	2	8	1.1095	1	8	1.0871
		15	8		8	1.0871			8	1.0871		8	1.0871		8	1.0871
		21	5		6	0.6414			6	0.6414		6	0.6414		5	0
		26	5		5	2.5215			5	0		5	1.2366		5	1.2366
		35	3		3	0			3	0		3	0		3	0
		39	3		3	0			3	0		3	0		3	0
Roszieg	25	18	8	23	8	1.699	25	5	8	0.4975	2	8	0.4975	2	7	0.2185
		21	6		7	2.6269			6	0.2483		6	0.2483		6	0.2483
		25	6		6	0.2483			6	0.2483		5	0		5	0
		32	4		4	0.9736			4	0.9232		4	0.9232		4	0.9232
Heskiaoff	28	138	8	26	8	1.2449	26	3	8	1.2449	3	8	1.2449	3	8	1.2449
		205	5		6	1.6388			6	2.8871		6	1.6797		6	0.4966
		216	5		5	0.2894			5	1.4828		5	0.2894		5	0.2894
		256	4		5	3.9155			5	1.4828		5	0.2894		5	0.2894
		324	4		4	1.2449			4	1.2449		4	0		4	0
		342	3		3	0.83			4	1.2449		3	0.83		3	0.83

Table 6.10 (cont) The results of the experiments as number of stations and fitness

Problem	# of tasks	Cycle time	Optimal # of stations	No alternatives			With alternatives									
				# of all rules			# of all rules	1 OR			2 ORs			3 ORs		
					Min. # of stations	Fitness		# of rules with 1 OR	Min. # of stations	Fitness	# of rules with 2 ORs	Min. # of stations	Fitness	# of rules with 3 ORs	Min. # of stations	Fitness
Buxey	29	27	13	26	13	2.6507	26	3	14	2.3528	2	14	2.3352	2	13	2.6507
		30	12		12	2.5033			13	2.6666		14	3.5248		12	2.4899
		33	11		12	2.5292			11	3.1363		12	1.3162		11	2.0316
		36	10		10	2			10	1.9688		10	2.0099		10	1.9795
		41	8		9	2.5333			9	3.7149		9	3.7087		9	1.3944
		47	7		8	0.6732			8	0.6414		8	0.6732		8	0.6414
		54	7		7	3.4433			7	2.28		7	2.1283		7	2.28
Sawyer	30	30	12	26	14	2.3642	26	3	14	2.2914	3	12	2.5033	3	12	2.4690
		33	11		12	2.5292			11	3.2986		11	3.2889		11	3.3177
		36	10		11	3.1836			11	3.284		10	1.9899		10	1.9688
		41	8		9	2.5333			9	4.1294		9	3.7149		9	2.4899
		47	7		8	0.6732			8	0.6732		8	0.6414		8	0.6414
		54	7		7	2.1283			7	2.1143		7	2.1283		7	2.1143
		75	5		5	1.5347			5	0.2894		5	1.4828		5	0.2894
Kilbridge	45	69	8	40	9	2.0496	40	5	9	2.0496	5	9	0.83	5	9	0.83
		79	7		8	1.2236			8	1.2236		8	0		8	0
		92	6		7	1.4047			7	1.4047		7	0.2185		7	0.2185
		110	6		6	1.2309			6	1.2309		6	0		6	0
		111	5		6	0			6	0		5	0.7549		5	0.7549
		138	4		4	0			4	0		4	0		4	0
		184	3		3	0			3	0		3	0		3	0

Table 6.10 (cont) The results of the experiments as number of stations and fitness

Problem	**# of tasks**	**Cycle time**	**Optimal # of stations**	**No alternatives**			**With alternatives**									
								1 OR			**2 ORs**			**3 ORs**		
				# of all rules	**Min. # of stations**	**Fitness**	**# of all rules**	**# of rules with 1 OR**	**Min. # of stations**	**Fitness**	**# of rules with 2 ORs**	**Min. # of stations**	**Fitness**	**# of rules with 3 ORs**	**Min. # of stations**	**Fitness**
Real Case	68	70	-	67	29	12.0099	67	11	29	12.028	4	29	12.025	2	28	9.61
Arcus 1	83	5853	14	82	14	264.5	82	5	14	195.4	2	14	73	2	14	75.7
		6309	13		13	186.2			14	467.8		14	443.3		13	201.8
		6842	12		13	351.7			12	304.4		12	348.6		12	305
		6883	12		13	488.7			12	194.3		13	649.8		12	244.4
		7571	11		11	258.5			11	272.3		11	257.3		11	265.5
		8412	10		10	194.6			10	157.9		10	255.6		10	113.3
		8898	9		9	143.8			9	164.7		9	164.4		9	140.9
		10816	8		8	728			8	500.5		8	184.4		8	376.5

The results of the problem instances as efficiencies are detailed in Table 6.11. In the first column, the problem source is reported. The second column reports the number of tasks and the third column reports the sum of the task times. In the fourth column, the cycle times are listed. The fifth column reports the minimum number of stations found in the literature for the original problems given in Table 6.9. The other columns report the best solutions of the generated instances with and without alternatives solved by the proposed GA. The number of rules with logical ORs is also given in Table 6.11 to evaluate the effects of alternatives on efficiencies.

Table 6.11 shows that the number of stations is getting smaller when a new alternative is added to the problem. When the number of stations remains the same, the efficiency is higher. As a result, the solutions get better when the number of ORs is increased. These improvements are shown in the tables and graphics given in the appendices.

Table 6.11 The results of the experiments as number of stations and efficiency

					No alternatives				With alternatives												
										1 OR				2 ORs				3 ORs			
Problem	**# of tasks**	**Sum of task times**	**Cycle time**	**Optimal # of stations**	**# of all rules**	**Min. # of stations**	**Max. station time**	**Eff. (%)**	**# of all rules**	**# of rules with 1 OR**	**Min. # of stations**	**Max. station time**	**Eff. (%)**	**# of rules with 2 ORs**	**Min. # of stations**	**Max. station time**	**Eff. (%)**	**# of rules with 3 ORs**	**Min. # of stations**	**Max. station time**	**Eff. (%)**
Bowman	8	75	20	5	7	5	17	88	7	2	5	17	88	-	-	-	-	-	-	-	-
Jaeschke	9	37	6	8	8	8	6	77	8	5	8	6	77	-	-	-	-	-	-	-	-
			7	7		7	7	76			8	6	77	-	-	-	-	-	-	-	-
			8	6		7	7	76			8	6	77	-	-	-	-	-	-	-	-
			10	4		4	10	93			4	10	93	-	-	-	-	-	-	-	-
			18	3		3	17	73			3	14	88	-	-	-	-	-	-	-	-
Jackson	11	46	7	8	10	8	7	82	10	2	7	7	94	-	-	-	-	-	-	-	-
			9	6		6	9	85			6	9	85	-	-	-	-	-	-	-	-
			10	5		5	10	92			5	10	92	-	-	-	-	-	-	-	-
			13	4		4	12	96			4	12	96	-	-	-	-	-	-	-	-
			14	4		4	12	96			4	12	96	-	-	-	-	-	-	-	-
			21	3		3	16	96			3	16	96	-	-	-	-	-	-	-	-

Table 6.11 (cont) The results of the experiments as number of stations and efficiency

					No alternatives				With alternatives													
										1 OR				2 ORs				3 ORs				
Problem	**# of tasks**	**Sum of task times**	**Cycle time**	**Optimal # of stations**	**# of all rules**	**Min. # of stations**	**Max. station time**	**Eff. (%)**	**# of all rules**	**# of rules with 1 OR**	**Min. # of stations**	**Max. station time**	**Eff. (%)**	**# of rules with 2 ORs**	**Min. # of stations**	**Max. station time**	**Eff. (%)**	**# of rules with 3 ORs**	**Min. # of stations**	**Max. station time**	**Eff. (%)**
Mitchell	21	105	14	8	20	9	14	83	20	2	8	14	94	2	8	14	94	1	8	14	94
			15	8		8	14	94			8	14	94		8	14	94		8	14	94
			21	5		6	18	97			6	18	97		6	18	97		5	21	100
			26	5		5	23	91			5	21	100		5	22	95		5	22	95
			35	3		3	35	100			3	35	100		3	35	100		3	35	100
			39	3		3	35	100			3	35	100		3	35	100		3	35	100
Roszieg	25	125	18	8	23	8	17	92	25	5	8	16	98	2	8	16	98	2	7	18	99
			21	6		7	20	89			6	21	99		6	21	99		6	21	99
			25	6		6	21	99			6	21	99		5	25	100		5	25	100
			32	4		4	32	98			4	32	98		4	32	98		4	32	98
Heskiaoff	28	1024	138	8	26	8	129	99	26	3	8	129	99	3	8	129	99	3	8	129	99
			205	5		6	172	99			6	173	99		6	172	99		6	171	100
			216	5		5	205	100			5	206	99		5	205	100		5	205	100
			256	4		5	208	98			5	206	99		5	205	100		5	205	100
			324	4		4	257	100			4	257	100		4	256	100		4	256	100
			342	3		3	342	100			4	257	100		3	342	100		3	342	100

Table 6.11 (cont) The results of the experiments as number of stations and efficiency

					No alternatives				**With alternatives**												
										1 OR				**2 ORs**				**3 ORs**			
Problem	**# of tasks**	**Sum of task times**	**Cycle time**	**Optimal # of stations**	**# of all rules**	**Min. # of stations**	**Max. station time**	**Eff. (%)**	**# of all rules**	**# of rules with 1 OR**	**Min. # of stations**	**Max. station time**	**Eff. (%)**	**# of rules with 2 ORs**	**Min. # of stations**	**Max. station time**	**Eff. (%)**	**# of rules with 3 ORs**	**Min. # of stations**	**Max. station time**	**Eff. (%)**
			27	13		13	27	92			14	25	93		14	25	93		13	27	92
			30	12		12	29	93			13	27	92		14	26	89		12	29	93
			33	11		12	29	93			11	32	92		12	28	96		11	31	95
Buxey	29	324	36	10	26	10	34	95	26	3	10	34	95	2	10	34	95	2	10	34	95
			41	8		9	38	95			9	38	95		9	39	92		9	37	97
			47	7		8	41	99			8	41	99		8	41	99		8	41	99
			54	7		7	49	94			7	48	96		7	48	96		7	48	96
			30	12		14	25	93			14	25	93		12	29	93		12	29	93
			33	11		12	29	93			11	32	92		11	32	92		11	32	92
			36	10		11	32	92			11	32	92		10	34	95		10	34	95
Sawyer	30	324	41	8	26	9	37	97	26	3	9	39	92	3	9	39	92	3	9	38	95
			47	7		8	41	99			8	41	99		8	41	99		8	41	99
			54	7		7	48	96			7	48	96		7	48	96		7	48	96
			75	5		5	66	98			5	65	100		5	66	98		5	65	100
			69	8		9	63	97			9	63	97		9	62	99		9	62	99
			79	7		8	70	99			8	70	99		8	69	100		8	69	100
			92	6		7	80	99			7	80	99		7	79	100		7	79	100
Kilbridge	45	552	110	6	40	6	93	99	40	5	6	93	99	5	6	92	100	5	6	92	100
			111	5		6	92	100			6	92	100		5	111	99		5	111	99
			138	4		4	138	100			4	138	100		4	138	100		4	138	100
			184	3		3	184	100			3	184	100		3	184	100		3	184	100

Table 6.11 (cont) The results of the experiments as number of stations and efficiency

Problem	# of tasks	Sum of task times	Cycle time	Optimal # of stations	No alternatives				With alternatives														
										1 OR				2 ORs				3 ORs					
					# of all rules	Min. # of stations	Max. station time	Eff. (%)	# of all rules	# of rules with 1 OR	Min. # of stations	Max. station time	Eff. (%)	# of rules with 2 ORs	Min. # of stations	Max. station time	Eff. (%)	# of rules with 3 ORs	Min. # of stations	Max. station time	Eff. (%)
Real Case	68	1753	70	-	67	29	70	86		11	29	70	86	4	29	70	86	2	28	70	89
Arcus 1	83	75707	5853	14	82	14	5621	96	82	5	14	5561	97	2	14	5467	99	2	14	5469	99
			6309	13		13	5940	98			14	5766	94		14	5766	94		13	5977	97
			6842	12		13	6087	96			12	6491	97		12	6562	96		12	6542	96
			6883	12		13	6196	94			12	6443	98		13	6345	92		12	6488	97
			7571	11		11	7091	97			11	7091	97		11	7091	97		11	6928	99
			8412	10		10	7722	98			10	7698	98		10	7768	97		10	7660	99
			8898	9		9	8528	99			9	8545	98		9	8528	99		9	8528	99
			10816	8		8	10048	94			8	9862	96		8	9600	99		8	9766	97

The results in Table 6.10 and Table 6.11 show that CCALBP can be solved through the proposed GA efficiently. It is shown that balance efficiency improves when the number of alternatives increases. CCALBP allows alternative assembly processes. Therefore, precedence constraints are relaxed. As the number of alternatives increases, the number of stations and the fitness value get smaller. As a result, the proposed GA performs better when more alternatives are added to CCALBP.

6.5 Chapter Summary

In this chapter, a GA based on the rule-base was proposed to solve CCALBP. The specific characteristics of the proposed GA were also explained step by step on an example problem of sewing a simple pant.

The proposed GA was developed in Matlab 7.0. By experimenting different set of control parameters, the robustness of the suggested approach was tested. The computational experiments were carried out on a set of generated problems by adapting the case problems in the literature. The generated benchmark problems with different instances were solved for this novel problem.

Based on the computational experiments, it can be stated that the solution quality in terms of balance efficiency and the number of stations improves when the number of alternatives increases.

CHAPTER SEVEN

CONCLUSION

7.1 Summary and Concluding Remarks

From the earliest days, ancient man used assembly techniques to make tools, weapons, ships, machinery, furniture, and garment. Manufacturing evolved time by time. Assembly lines are the most commonly used methods in a mass production environment, because they allow the assembly of complex products by workers with limited training, by dedicated machines and/or by robots. Recently, mass production has been challenged by mass customization. Production systems and supply chains are designed to handle high variety of products. Today, assembly lines are still up to date.

The installation of an assembly line which is a long-term decision usually requires large capital investments. Therefore, it is important to design and balance an assembly line in a way that it should work as efficiently as possible. The assembly line balancing is the allocation of the tasks among stations so that the precedence relations are not violated and a given objective function is optimized. ALBP deals with balancing the assembly line with respect to the precedence constraints and objective function(s).

There are technological restrictions or/and physical sequencing requirements on the assembly line which are called precedence constraints. The sequence of tasks defined by the precedence constraints is represented by a precedence graph. But, there are some shortcomings of the precedence graphs. They cannot represent all the possible assembly sequences of a product in a single graph and cannot describe some complicated constraints. In order to overcome the aforementioned difficulties, a rule-based assembly model was proposed to model all assembly constraints.

The aim of this dissertation was to extend the rule-based assembly modeling and to introduce the *complex-constrained assembly line balancing problem* (CCALBP), which is of the general ALBPs, in order to model all assembly constraints through a rule-base to tackle alternative ways of assembling a product and their effects on task times, precedence relations and the line balance simultaneously.

It was shown how to model all assembly constraints through the well known If-then rules, and how to solve the problem through CP and IP models mapped from the rule-based model through a small real-life example. It was also shown how to map a rule-based model to a CP or an IP model. This mapping was not possible from graph-based models that address, though roughly, complex assembly constraints. Thus, CCALBP can be solved *only* through rule-based modeling, but not graph-based modeling. Rule-based and graph-based models were compared in terms of modeling capability.

In this dissertation, a GA integrated with the rule-base was proposed to solve CCALBP. The specific characteristics of the proposed GA were devised with the inspiration taken from the current examples in the literature. These characteristics were explained on an example problem of sewing a simple pant. The control parameters of the GA were optimized to improve the performance.

Since CCALBP is a novel problem, there is no set of benchmark instances for testing. Therefore, the computational experiments were carried out on a set of self-made instances generated by adapting well-known benchmark problems from the literature. Some alternative routes are created and added to these literature problems. A new alternative was added to the original problem in each step. The GA with the rule base solved each type of the problem with new alternatives simultaneously as well as the original problem. The proposed GA resulted better in the generated problems when new alternatives were added. The fitness value consisted of two objectives, minimizing the number of stations and obtaining balanced stations. When more alternatives were added, better balanced stations were obtained. It was shown that balance efficiency improved when the number of alternatives increased. As the number of alternatives increased, the number of stations and the fitness value got smaller. Based on the experiments, it is stated that the proposed GA performs better and the objectives improve when more alternatives are added to CCALBP.

7.2 Contributions

The research proposed in this dissertation provides several contributions. The contributions are presented in this section as follows.

Extensive literature review indicates that the researchers generally use precedence graphs to represent the precedence constraints and the sequence of tasks in an assembly line. But, there are some shortcomings of the precedence graphs:

- They cannot represent all the possible assembly sequences of a product in a single graph.
- They exclude some logic statements.
- They allow limited flexibility.
- They cannot describe some complicated constraints.

Despite their shortcomings, researchers continue to employ precedence graphs in ALBP. There are some alternative representation methods, but the literature is relatively sparse in addressing alternative ways of assembling a product for the ALBP. In other words, the literature tackles the ALBP based on traditional precedence graphs in general, rather than investigating more effective modeling tools than precedence graphs to solve the ALBP.

In this dissertation a well known tool, rule-base, is employed in modeling and solving the ALBP for the *first time* and extends this literature in terms of modeling scope for assembly constraints in line balancing.

This dissertation introduced *a novel line balancing problem*: *complex-constrained assembly line balancing problem* (CCALBP), which is of the general ALBPs, in order to model all assembly constraints through a rule-base to tackle alternative ways of assembling a product and their effects on task times, precedence relations and the line balance simultaneously.

This dissertation extends the rule-based modeling of assembly constraints (Salum & Supciller, 2007, 2008), and solves CCALBP (Topaloglu et al., 2009) through GAs (Supciller & Salum, 2009).

It was shown how to:

- model all assembly constraints through the well known If-then rules,
- map a rule-based model to a CP or an IP model,
- solve CCALBP through CP and IP models mapped from the rule-based model,
- solve CCALBP through a GA integrated with the rule-base,
- solve a real-life case of CCALBP.

Based on the experiments, it was also shown that the proposed GA performed better and the objectives improved when more alternatives were added to CCALBP.

7.3 Future Research Directions

Since assembly lines have many characteristics, any characteristic can be added to the new problem, CCALBP, to have a different CCALBP. Therefore many research directions can be added to future research topics stated in the following.

The experiments performed by the proposed GA can be performed by using IP, CP or another meta-heuristic such as simulated annealing (SA) or tabu search (TS) employing the rule-base for CCALBP. In other way, the proposed GA can be hybridized with another solution approach. Or, in order to improve the performance of the proposed GA, additional heuristic algorithm can be used. They can be compared in terms of solution efficiency.

Different types of the problem according to the objectives can be solved such as cost or profit oriented CCALP. Different methods such as pareto optimization can be used to solve multi-objective CCALBP.

Some fuzzy rules can also be employed in a rule-base to model vagueness in assembly constraints.

Since CCALBP addresses a wide variety of assembly problems involving complex constraints, many complicated constraints of real-life assembly lines can be modeled easily through a rule-base to solve the problems more realistically. They can include features such as parallel workstations, two-sided workstations, U-shaped line layout, workload constraints, assignment restrictions such as positive or negative zoning constraints, multi or mixed model, and stochastic processing times.

With the help of these further investigations, theoretical studies and practical applications can match and the gap between scientific research and industrial needs can be shortened.

REFERENCES

Agpak, K., & Gokcen, H. (2005). Assembly line balancing: Two resource constrained cases. *International Journal of Production Economics*, *96*, 129–140.

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

Amen, M. (2001). Heuristic methods for cost-oriented assembly line balancing: A comparison on solution quality and computing time. *International Journal of Production Economics*, *69*, 255-264.

Amen, M. (2006). Cost-oriented assembly line balancing: Model formulations, solution difficulty, upper and lower bounds. *European Journal of Operational Research*, *168*, 747-770.

Anderson, E. J., & Ferris, M. C. (1994). Genetic algorithms for combinatorial optimization: The assembly line balancing problem. *ORSA Journal on Computing*, *6*, 161-173.

Arcus, A. L. (1966). COMSOAL: A computer method of sequencing operations for assembly lines. *International Journal of Production Research, 4*, 259-277.

Aytug, H., Khouja, M., & Vergara, F. E. (2003). Use of genetic algorithms to solve production and operations management problems: a review. *International Journal of Production Research*, *41*(17), 3955-4009.

Bard, J. F. (1989). Assembly line balancing with parallel workstations and dead time. *International Journal of Production Research, 27*(6), 1005-1018.

Battini, D., Faccio, M., Ferrari, E., Persona, A., & Sgarbossa, F. (2007). Design configuration for a mixed-model assembly system in case of low product demand. *International Journal of Advanced Manufacturing Technology*, *34*(1-2), 188-200.

Bautista, J., & Pereira, J. (2002). Ant algorithms for assembly line balancing. *Lecture Notes in Computer Science*, *2463*, 65–75.

Bautista, J., & Pereira, J. (2007). Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research*, *177,* 2016-2032.

Bautista, J., & Pereira, J. (2009). Dynamic programming based heuristic for the assembly line balancing problem. *European Journal of Operational Research*, *194*, 787-794.

Bautista, J., Suarez, R., Mateo, M., & Companys, R. (2000). Local search heuristics for the assembly line balancing problem with incompatibilities between tasks. *In the Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, 2404-2409.

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

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

Baykasoglu, A., & Dereli, T. (2008). Two-sided assembly line balancing using an ant-colony-based heuristic. *International Journal of Advanced Manufacturing Technology*, *36*, 582-588.

Baykasoglu, A., & Dereli, T. (2009). Simple and U-type assembly line balancing by using ant colony based algorithm. *Mathematical and Computational Applications*, *14*(1), 1-12.

Baykasoglu, A., & Ozbakir, L. (2007). Stochastic U-line balancing using genetic algorithms. *International Journal of Advanced Manufacturing Technology*, *32*(1-2), 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.

Blum, C., Bautista, J., & Pereira, J. (2006). Beam-ACO applied to assembly line balancing. *ANTS 2006, LNCS, 4150*, 96-107.

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

Boctor, F. F. (1995). A multiple rule heuristic for assembly line balancing. *Journal of the Operational Research Society, 44*, 62-69.

Bowman, E. H. (1960). Assembly line balancing by linear programming. *Operations Research*, 8(3), 385-389.

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. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research*, 183, 674–693.

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

Brailsford, S. C., Potts, C. N., & Smith, B. M. (1999). Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119, 57-581.

Brown, E. C., & Sumichrast, R. T. (2005). Evaluating performance advantages of grouping genetic algorithms. *Engineering Applications of Artificial Intelligence, 18*, 1-12.

Brudaru, O., & Valmar, B. (2004). Genetic algorithm with embryonic chromosomes for assembly line balancing with fuzzy processing times. *The 8th International Research/Expert Conference Trends in the Development of Machinery and Associated Technology*, Neum, Bosnia and Herzegovina.

Bryton, B. (1954). Balancing of a Continuous Production Line, *Unpublished M.S. Thesis*, Northwestern University.

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.

Bukchin, J., & Rubinovitz, J. (2003). A weighted approach for assembly line design with station paralleling and equipment selection. *IIE Transactions, 35*, 73-85.

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

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

Capacho, L., & Pastor, R. (2008). ASALBP: the alternative subgraphs assembly line balancing problem. *International Journal of Production Research*, 46, 3503–3516.

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

Carnahan, B. J., Norman, B. A., & Redfern, M. S. (2001). Incorporating physical demand criteria into assembly line balancing. *IIE Transactions, 33*, 875-887.

Carraway R. L. (1989). A dynamic programming approach to stochastic assembly line balancing. Management Science, 35(4), 459-471.

Cevikcan, E., Durmusoglu, M. B., & Unal, M. E. (2009). A team-oriented design methodology for mixed model assembly systems. *Computers & Industrial Engineering, 56*, 576–599.

Chan, C. C. K., Hui, P. C. L., Yeung, K. W., & Ng, F. S. F. (1998). Handling the assembly line balancing problem in the clothing industry using a genetic algorithm. *International Journal of Clothing Science and Technology, 10*(1), 21-37.

Chen, R. S., Lu, K. Y., & Yu, S. C. (2002). A hybrid genetic algorithm approach on multi-objective of assembly planning problem. *Engineering Applications of Artificial Intelligence, 15*, 447–457.

Chiang, W. C. (1998). The application of a tabu search metaheuristic to the assembly line balancing problem. *Annals of Operations Research, 77*, 209–227.

Chiang, W. C., & Urban, T. L. (2006). The stochastic U-line balancing problem: A heuristic approach. *European Journal of Operational Research, 175*, 1767-1781.

Choi, G. (2009). A goal programming mixed-model line balancing for processing time and physical workload. *Computers & Industrial Engineering, 57*(1), 395-400.

Cilkin, S. (2003). Line balancing with genetic algorithms. *Unpublished Master Thesis*, The Graduate School of Natural and Applied Sciences, Gazi University, Ankara.

Coley, D. A. (1999). *An introduction to genetic algorithms for scientists and engineers*. Singapore: World Scientific.

Corominas, J. P. A. (1999). Modeling and solving the SALB-E problem. *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning*. Porto, Portugal, July, 356-360.

Corominas, A., Pastor, R., & Plans, J. (2008). Balancing assembly line with skilled and unskilled workers. *Omega, 36,* 1126-1132.

Dar-El, E. M., & Rubinovitch, Y. (1979). MUST-A multiple solutions technique for balancing single model assembly lines. *Management Science*, *25*, 1105-1114.

Davis, L. (1985). Applying adaptive algorithms to epistatic domains. *In the Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1, 162–164.

De Fazio, T. L., & Whitney, D. E. (1987). Simplified generation of all mechanical assembly sequences. *IEEE Journal of Robotics and Automation RA-3*, 6, 640–658.

Dimitriadis, S. G. (2006). Assembly line balancing and group working: A heuristic procedure for workers' groups operating on the same product and workstation. *Computers & Operations Research*, *33*, 2757–2774.

Dimopoulos, C., & Zalzala, A. M. S. (2000). Recent developments in evolutionary computation for manufacturing optimization: problems, solutions and comparisons. *IEEE Transactions on Evolutionary Computation*, *4*(2), 93-113.

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

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

Dowsland, K. A. (1996). Genetic Algorithms-A Tool for OR? *The Journal of the Operational Research Society, 47*(4), 550-561.

Dreo, J., Siarry, P., Petrowski, A., & Taillard, E. (2006). *Metaheuristics for Hard Optimization*. Berlin Heidelberg: Springer-Verlag.

Ege, Y., Azizoglu, M., & Ozdemirel, N. E. (2009) Assembly line balancing with station paralleling. *Computers & Industrial Engineering, 57*(4), 1218-1225.

Eiben, A. E., Michalewicz, Z., Schoenauer, M., & Smith, J.E. (2007). Parameter control in evolutionary algorithms. In *Studies in Computational Intelligence (SCI) 54*(19–46). Berlin Heidelberg: Springer-Verlag.

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

Falkenauer, E. (1991). A genetic algorithm for grouping. *In the Proceedings of the Fifth International Symposium on Applied Stochastic Models and Data Analysis*, Granada, Spain.

Falkenauer, E. (1997). A grouping genetic algorithm for line balancing with resource dependent task times. *In the Proceedings of the Fourth International Conference on Neural Information Processing*, Dunedin, New Zealand, 464-468.

Falkenauer, E., & Delchambre, A. (1992). A genetic algorithm for bin packing and line balancing. *In the Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, 1189-1192.

Fonseca, D. J., Guest, C.L., Elam, M., & Karr, C.L. (2005). A fuzzy logic approach to assembly line balancing. *Mathware & Soft Computing, 12*, 57-74.

Gamberini, R., Grassi, A., & Rimini, B. (2006). A new multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem. *International Journal of Production Economics*, *102*, 226–243.

Gao, J., Sun, L., Wang, L., & Gen, M. (2009) An efficient approach for type II robotic assembly line balancing problems. *Computers & Industrial Engineering, 56*, 1065–1080

Gen, M., & Cheng, R. (1997). *Genetic algorithms & engineering design*. New York: John Wiley & Sons.

Gen, M., Cheng, R., & Lin L. (2008). *Network models and optimization: multiobjective genetic algorithm approach.* London: Springer-Verlag.

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.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research, 13*(5), 533-549.

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., & Benzer, R. (2006). Balancing of parallel assembly lines. *International Journal of Production Economics*, *103*, 600–609.

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.

Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization & Machine Learning*. Boston: Addison-Wesley.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In G.J.E. Rawlins, (Ed.). (53-69). *Foundations of Genetic Algorithms.* San Francisco: Morgan Kaufmann.

Goncalves, J. F., & De Almedia, J. R. (2002). A hybrid genetic algorithm for assembly line balancing. *Journal of Heuristic, 8*, 629-642.

Groover, M. P. (2001). Automation, production systems, and computer-integrated *manufacturing* (2nd ed.). New Jersey: Prentice Hall.

Guerriero, F., & Miltenburg, J. (2002). The stochastic U-line balancing problem. *Naval Research Logistics, 50*(1), 31-57.

Guo, Z. X., Wong, W. K., Leung, S. Y. S., Fan, J. T., & Chan, S. F. (2008). A genetic-algorithm-based optimization model for solving the flexible assembly line balancing problem with work sharing and workstation revisiting. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, *38*(2), 218-228.

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

Haupt R. L., & Haupt S. E. (2004). *Practical genetic algorithms* (2nd ed.). New Jersey: John Wiley.

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

Held, M., & Karp, R. M., (1961) Dynamic programming approach to sequencing problems. Proceedings of the 16th ACM annual conference**,** 71.201 - 71.204.

Held, M., Karp, R. M., & Shareshian, R. (1963). Assembly-line balancing-Dynamic programming with precedence constraints, *Operations Research, 11*(3), 442-460.

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

Helgeson, W. B., Salveson, M. E., & Smith, W. W. (1954) How to balance an assembly line, *Technical Report*, No: 7, New Caraan, Conn: Carr Press.

Hoffmann T. R. (1963). Assembly line balancing with a precedence matrix. *Management Science*, *9*(4), 551–562.

Hoffmann T. R. (1992). EUREKA: A hybrid system for assembly line balancing. *Management Science, 38*, 39–47

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

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

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

Hu, S. J., Zhu, X., Wang, H., & Koren, Y. (2008). Product variety and manufacturing complexity in assembly systems and supply chains. *CIRP Annals - Manufacturing Technology*, *57*(1), 45-48.

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.

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.

ILOG OPL Studio 3.7 (2003). Language manual. ILOG SA: Gentilly.

Ishibuchi, M., & Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, *28*(3), 392-403.

Jiao, J., Kumar, A., & Martin, W. (2006). A web-based interactive advisor for assembly line balancing. *International Journal of Advanced Manufacturing Technology, 27*, 1192-1201.

Jin, M. & Wu, S. D. (2002). A new heuristic method for mixed model assembly line balancing problem. *Computers & Industrial Engineering, 44*, 159–169.

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

Johnson, R. V. (1981). Assembly line balancing algorithms: Computation comparisons. *International Journal of Production Research, 19*, 277- 287.

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

Johnson, R. V. (1988). Optimally balancing large assembly lines with "FABLE". *Management Science, 34*(2), 240–253.

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., Paksoy, T., & Chang C. (2009). A binary fuzzy goal programming approach to single model straight and U-shaped assembly line balancing. *European Journal of Operational Research*, 195, 335-347.

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.*

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

Kilbridge, M. D., & Wester, L. (1962). A Review of analytical systems of line balancing, *Operations Research, 10*(5), 626-638.

Kilincci, O. (2010). A Petri net-based heuristic for simple assembly line balancing problem of type-2. *International Journal of Advanced Manufacturing Technology, 46,* 329-338.

Kilincci, O., & Bayhan, G. M. (2006). A Petri net approach for simple assembly line balancing problems. *International Journal of Advanced Manufacturing Technology, 30,* 1165-1173.

Kilincci, O., & Bayhan, G. M. (2008). A P-invariant-based algorithm for simple assembly line balancing problem of type-1. *International Journal of Advanced Manufacturing Technology, 37,* 400-409.

Kim, J. Y., Kim, Y & Kim, Y. K. (2001). An endosymbiotic evolutionary algorithm for optimization. *Applied Intelligence, 15(2)*, 117–130.

Kim, Y. J., Kim, Y. K., & Cho, Y. (1998). A heuristic-based genetic algorithm for workload smoothing in assembly lines. *Computers and Operations Research, 25*(2), 99-111.

Kim, Y. K., Song, W. S., & Kim, J. H. (2009). A mathematical model and a genetic algorithm for two-sided assembly line balancing. *Computers and Operations Research 36*, 853 – 865.

Kim, Y. K., Kim, J. Y. & Kim, Y. (2000a). A co-evolutionary 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 Operations Research, 168*, 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.

Kim, Y. K., Kim, Y., & Kim, Y. J. (2000c). Two-sided assembly line balancing: a genetic algorithm approach. *Production Planning and Control, 11*(1), 44-53.

Kim, Y. K., Kim, Y. J., & Kim, Y. (1996). Genetic algorithms for assembly line balancing with various objectives. *Computers and Industrial Engineering*, *30*(3), 397-409.

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671-680.

Klein, M. (1963). On assembly line balancing. *Operations Research, 11*, 274-281.

Klein, R., & Scholl, A. (1996). Maximizing the production rate in simple assembly line balancing - A branch and bound procedure. *European Journal of Operations Research, 91*, 367–385.

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

Lambert, A. J. D. (2006). Generation of assembly graphs by systematic analysis of assembly structures. *European Journal of Operational Research*, 168, 932-951.

Lapierre, S. D. & Ruiz, A. (1999). Equilibrer une chaîne d'assemblage avec Microsoft ACCESS97. In Proceedings of 3rd International Industrial Engineering Conference, 357-364. Presses Internationales Polytechnique, Montreal.

Lapierre, S. D., Ruiz, A., & Soriano, P. (2006). Balancing assembly lines with tabu search. *European Journal of Operational Research, 168*, 826–837.

Lee, S., Soak, S., Kim, K., Park, H., & Jeon, M. (2008). Statistical properties analysis of real world tournament selection in genetic algorithms. *Applied Intelligence*, doi 10.1007/s10489-007-0062-2.

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

Levitin, G., Rubinovitz, J., & Shnits, B. (2006). A genetic algorithm for robotic assembly line balancing. *European Journal of Operational Research, 168*, 811–825.

Liu, S. B., Ng, K. M., & Ong, H. L. (2008). Branch-and-bound algorithms for simple assembly line balancing problem. *International Journal of Advanced Manufacturing Technology, 36*, 169-177.

Liu, S. B., Ong, H. L., & Huang, H. C. (2005). A bidirectional heuristic for stochastic assembly line balancing Type II problem. *International Journal of Advanced Manufacturing Technology, 25*, 71-77.

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

Margulis, L. (1980). Symbiosis in cell evolution. San Fransisco: WH Freeman.

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. (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., & Tarasewich, P. (2003). Using ant techniques to solve the assembly line balancing problem. *IIE Transactions. 35*, 605–617.

McMullen, P. R., & Tarasewich, P. (2006). Multi-objective assembly line balancing via a modified ant colony optimization technique. *International Journal of Production Research, 44*(1), 27-42.

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

Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation, 4*, 1-32.

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

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

Miralles, C., Garcia-Sabater, J. P., Andres, C. & Cardos, M. (2008). Branch and bound procedures for solving the assembly line worker assignment and balancing problem: Application to sheltered work stations for disabled. *Discrete Applied Mathematics*, 156, 352-367.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Cambridge: The MIT Press.

Montgomery, D.C. (2001). *Design and Analysis of Experiments*. New York: John Wiley & Sons.

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.

Moon, I., Logendran, R., & Lee, J. (2009). Integrated assembly line balancing with resource restrictions. *International Journal of Production Research, 47*(19), 5525–5541

Mosheiov, G. (1991). V-shaped policies for scheduling deteriorating jobs. *Operations Research, 39,* 979-991.

Mosheiov, G. (2001). Scheduling problems with a learning effect. *European Journal of Operational Research, 132,* 687-693.

Murata, T., & Ishibuchi, M. (1996). Positive and negative combination effects of crossover and mutation operators in sequencing problems. *In the Proceedings of IEEE International Conference on Evolutionary Computation*, 170-175.

Murata, T., Ishibuchi, M., & Tanaka, H. (1996). Multi-objective genetic algorithms and its application to flowshop scheduling. *Computers & Industrial Engineering, 30*(4), 957-968.

Nearchou, A. C. (2007). Balancing large assembly lines by a new heuristic based on differential evolution method. *International Journal of Advanced Manufacturing Technology, 34*, 1016–1029.

Nearchou, A. C. (2008). Multi-objective balancing of assembly lines by population heuristic. *International Journal of Production Research, 46*(8), 2275–2297.

Ozcan, U., & Toklu, B. (2009). Multiple-criteria decision–making in two-sided assembly line balancing: A goal programming and a fuzzy goal programming models, *Computers & Operations Research, 36*, 1955-1965.

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

Pastor, R, Andris, C., Duran, A., & Pirez, M. (2002). Tabu search algorithms for an industrial multi-product and multi-objective assembly line balancing problem, with reduction of the task dispersion. *Journal of the Operational Research Society, 53*(12), 1317-1323.

Patterson, J. H., & Albracht, J. J. (1975). Assembly-line balancing: Zero-one programming with Fibonacci Search. *Operations Research, 23*, 66–172.

Peeters, M., & Degraeve, Z. (2006). A linear programming based lower bound for the simple assembly line balancing problem. *European Journal of Operational Research, 168,* 716–731.

Peterson, C. (1993). A tabu search procedure for the simple assembly line balancing problem. *In the Proceedings of the Decision Science Institute Conference*, Washington, DC, 1502-1504.

Pierreval, H., Caux, C., Paris, J. L., & Viguier, F. (2003). Evolutionary approaches to the design and organization of manufacturing systems. *Computers & Industrial Engineering*, 44, 339-364.

Pirlot, M. (1996) General local search methods. *European Journal of Operational Research, 92*, 493-511.

Ponnambalam, S. G., Aravindan, P., & Naidu, G. M. (2000). A multi-objective genetic algorithm for solving assembly line balancing problem. *The International Journal of Advanced Manufacturing Technology*, 16, 341–352.

Potter, M. A., (1997). The design and analysis of a computational model of cooperative coevolution. *Ph.D. dissertation*, George Mason University, UA.

Reeves, C. R. (1997). Genetic algorithms for the operations researcher. *INFORMS Journal on Computing*, *9*(3), 231–250.

Reeves, C. R., & Rowe, J. E. (2003). *Genetic algorithms: Principles and perspectives*. Dordrecht: Kluwer Academic Publishers.

Rekiek, B., & Delchambre, A. (2006). *Assembly line design*. London: Springer-Verlag.

Rekiek, B., De Lit, P., & Delchambre, A. (2000). Designing Mixed-Product Assembly Lines. IEEE Transactions on robotics and Automation, 16(3), 268-280.

Rekiek, B., De Lit, P., Pellichero, F., Falkenauer, E., & Delchambre, A. (1999). Applying the equal piles problem to balance assembly lines. *In the Proceedings of the ISATP 1999*, Porto, Portugal, 399-404.

Rekiek, B., Dolgui, A., Delchambre, A., & Bratcu, A. (2002). State of art of optimization methods for assembly line design. *Annual Review in Control*, 26, 163-174.

Robert, S. D., & Villa, C. D. (1970). On a multi-product assembly line balancing problem. *AIIE Transactions, 2*, 361–364.

Robinson, L. W., McClain, J. O., & Thomas, L. J. (1990). The good, the bad and the ugly: Quality on an assembly line. *International Journal of Production Research, 28*, 963–980.

Rothlauf, F. (2006). *Representations for Genetic and Evolutionary Algorithms* (2nd Ed.), Berlin Heidelberg: Springer.

Rubinovitz, J., & Levitin, G. (1995). Genetic Algorithm for assembly line balancing. *International Journal of Production Economics, 41*, 343-354.

Ruijun, Z., Dingfang, C., Yong, W., Zhonghua, Y., & Xinxin, W. (2007). Study on line balancing problem based on improved genetic algorithms. *In the Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing, WiCom 2007,* 21-25 Sept. 2007, 2033 – 2036.

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.

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

Salum, L., & Supciller, A. A. (2007). Rule-based representation of precedence constraints for assembly line balancing. *In the proceedings of the 27th National Conference on Operations Research and Industrial Engineering*, in Turkish.

Salum, L., & Supciller, A. A. (2008). Rule-based modeling of assembly constraints for line balancing. *ICIC (2), LNAI, 5227*, 783-789.

Salveson, M. E. (1955). The assembly line balancing problem. *The Journal of Industrial Engineering*, 6 (3), 18–25.

Scholl, A. (1993). Data of assembly line balancing problems. *Working Paper*, TH Darmstadt. Retrieved December 7, 2008, from http://www.assembly-line-balancing.de/.

Scholl, A. (1999). *Balancing and sequencing of assembly lines* (2nd ed.). New York: Springer-Verlag.

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

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

Scholl, A., & Boysen, N. (2009). Designing parallel assembly lines with split workplaces: Model and optimization procedure. *International Journal of Production Economics, 119*(1), 90-100.

Scholl, A., Fliedner, M., & Boysen, N. (2010). Absalom: Balancing assembly lines with assignment restrictions. *European Journal of Operations Research, 200*(3), 688-701.

Scholl, A., & Klein, R. (1999). Balancing assembly lines effectively - a computational comparison. European Journal of Operational Research, 114, 50–58.

Scholl, A., & Voss, S. (1996). Simple assembly line balancing-Heuristic approaches. *Journal of Heuristics, 2*, 217–244.

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

Simaria, A. S., & Vilarinho, P. M. (2001a). A genetic algorithm approach for balancing mixed model assembly lines with parallel workstations. *In the Proceedings of the 6th Annual International Conference on Industrial Engineering Theory, Applications and Practice*, November 18-20, 2001, San Francisco, USA.

Simaria, A. S., & Vilarinho, P. M. (2001b). The simple assembly line balancing problem with parallel workstations- a simulated annealing approach. *International Journal of Industrial Engineering, 8*(3), 230-240.

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

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

Sivanandam, S. N., & Deepa, S. N. (2008). *Introduction to Genetic Algorithms*. New York: Springer Berlin Heidelberg.

Smith, B. (1995). A tutorial on constraint programming. *Research Report 95.14*, School of Computer Studies, University of Leeds.

Smith, A. E., & Coit, D. W. (1997). Constraint-handling techniques - Penalty functions. *In Handbook of Evolutionary Computation*, Chapter C 5.2. Bristol: Institute of Physics Publishing and Oxford University Press.

Sprecher, A. (1999). A competitive branch-and-bound algorithm for the simple assembly line balancing problem. *International Journal of Production Research, 37*, 1787–1816.

Stockton, D. J., Quinn, L., & Khalil, R. A. (2004a). Use of genetic algorithms in operations management Part 1: applications. *Proceeding of the Institution of Mechanical Engineers-Part B: Journal of Engineering Manufacture, 218*(3), 315-327.

Stockton, D. J., Quinn, L., & Khalil, R. A. (2004b). Use of genetic algorithms in operations management Part 2: results. *Proceeding of the Institution of Mechanical Engineers-Part B: Journal of Engineering Manufacture, 218*(3), 329-343.

Storn, R., & Price, K. (1997) Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341-354.

Supciller, A. A., & Salum, L. (2009). A genetic algorithm for the complex-constrained assembly line balancing problem. *23rd European Conference on Operational Research,* EURO 23, Bonn, Germany.

Suresh, G., & Sahu, S. (1994). Stochastic assembly line balancing using simulated annealing. *International Journal of Production Research, 32*(8), 1801-1810.

Suresh, G., Vinod, V. V., & Sahu, S. (1996). A genetic algorithm for assembly line balancing. *Production Planning and Control*, 7(1), 38-46.

Suwannarongsri, S., & Puangdownreong, D. (2008). Optimal assembly line balancing using tabu search with partial random permutation technique. *International Journal of Management Science and Engineering Management*, *3*(1), 3-18.

Talbot, F. B. & Patterson, J. H. (1984). An integer programming algorithm with network cuts for solving the assembly line balancing problem. *Management Science, 30*, 85-99.

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

Tasan, S. O., & Tunalı, S. (2008). A review of the current applications of genetic algorithms in assembly line balancing. *Journal of Intelligent Manufacturing*, 19, 49-69.

Thangavelu, S. R. & Shetty, C. M. (1971). Assembly line balancing by zero-one integer programming. *AIIE Transactions, 3*, 61–68.

Toksari, M. D., Isleyen, S. K., Guner, E., & Baykoc, O. F. (2008). Simple and U-type assembly line balancing problems with a learning effect. *Applied Mathematical Modelling, 32*, 2954-2961.

Toksari, M. D., Isleyen, S. K., Guner, E., & Baykoc, O. F. (2010). Assembly line balancing problem with deterioration tasks and learning effect. *Expert Systems with Applications, 37*(2), 1223-1228.

Topaloglu, S., Salum, L., & Supciller, A. A. (2009). Constraint programming for solving the complex-constrained assembly line balancing problem. *23rd European Conference on Operational Research*, EURO 23, Bonn, Germany.

Tseng, H. E., & Tang, C. E. (2006) A sequential consideration for assembly sequence planning and assembly line balancing using the connector concept. *International Journal of Production Research, 44*(1), 97–116.

Tsujimura, Y., Gen, M., & Kubota, E. (1995). Solving fuzzy assembly line balancing using genetic algorithms. *Computers & Industrial Engineering, 29*(1-4), 543-547.

Ugurdag, H. F., Rachamadugu, R., & Papachristou, C. A. (1997). Designing paced assembly lines with fixed number of stations. *European Journal of Operational Research, 102*(3), 488-501.

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

Valente, S. A., Lopes, H. S., & Arruda, L. V. R. (2002). Genetic algorithms for the assembly line balancing problem: a real-world automotive application. In: R. Roy, M. Köppen, S. Ovaska, T. Fukuhashi, F. Hoffman, (Ed.). *Soft Computing and Industry: Recent Applications* (319-328), Berlin: Springer-Verlag.

Van Assche, F., & Herroelen, W. S. (1979). An optimal procedure for the single model deterministic assembly line balancing problem. *European Journal of Operational Research, 3*, 142–149.

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 algorithm for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research, 44*(2), 291–303.

White, W. W. (1961) Comments on a paper by Bowman. *Operations Research*, *9*(2), 274-276.

Wong, W. K., Mok, P. Y., & Leung, S. Y. S. (2006) Developing a genetic optimisation approach to balance an apparel assembly line. *International Journal of Advanced Manufacturing Technology, 28*, 387–394.

Wu, E. F., Jin, Y., Bao, J. S., & Hu, X. F. (2008) A branch-and-bound algorithm for two-sided assembly line balancing. *International Journal of Advanced Manufacturing Technology, 39*, 1009-1015.

Yu, J., & Yin, Y. (2009) Assembly line balancing based on an adaptive genetic algorithm. *International Journal of Advanced Manufacturing Technology,* DOI 10.1007/s00170-009-2281-7.

Yu, J., Yin, Y., & Chen, Z. (2006). Scheduling of an assembly line with a multi-objective genetic algorithm. *International Journal of Advanced Manufacturing Technology, 28*, 551–555.

Zhao, X., Ohno, K. & Lau, H.-S. (2004). A balancing problem for mixed model assembly lines with a paced moving conveyor. *Naval Research Logistics, 51*(3), 446-464.

APPENDICES

APPENDIX A1. Matrix representation for the rule-base of problems Bowman, Jaeschke, and Jackson with 1 OR

BOWMAN 1 OR

2	**3**	**4**	**5**	**6**	**6**	**7**	**7**	**8**
1	2	2	3	3	4	5	6	6
0	0	0	4	0	0	5	0	0

JAESCHKE 1 OR

2	**2**	**3**	**4**	**4**	**5**	**6**	**7**	**8**	**9**	**9**	**9**
1	0	1	2	3	4	4	4	5	6	7	8
3	4	0	1	0	2	2	2	0	0	0	0

JACKSON 1 OR

2	**3**	**4**	**5**	**6**	**7**	**7**	**7**	**8**	**8**	**9**	**10**	**10**	**11**	**11**
1	1	1	1	2	3	4	5	6	7	7	8	5	9	10
0	0	0	0	0	0	0	0	6	0	0	8	0	0	0

APPENDIX A2. Matrix representation for the rule-base of problem Mitchell with 1 OR, 2ORs and 3 ORs

MITCHELL 1 OR

2	**3**	**4**	**4**	**5**	**6**	**7**	**8**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**15**	**15**	**16**	**17**	**17**	**18**	**18**	**19**	**19**	**20**	**21**	**21**
1	1	3	0	4	5	5	6	7	8	9	9	9	9	7	10	11	12	15	13	16	13	15	14	18	17	2	4
0	0	2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	20

MITCHELL 2 ORs

2	**3**	**4**	**4**	**5**	**6**	**7**	**8**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**15**	**15**	**16**	**17**	**17**	**18**	**18**	**19**	**19**	**20**	**21**	**21**	**21**
1	1	3	0	4	5	5	6	7	8	9	9	9	9	7	10	11	12	15	13	16	13	15	14	18	17	2	4	0
0	0	2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	4	20

MITCHELL 3 ORs

2	**3**	**4**	**4**	**5**	**6**	**7**	**8**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**15**	**15**	**16**	**17**	**17**	**18**	**18**	**19**	**19**	**20**	**21**	**21**	**21**
1	1	3	0	4	5	5	6	7	8	9	9	9	9	7	10	11	12	15	13	16	13	15	14	18	19	2	4	0
0	0	2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	4	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	2	4	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0

APPENDIX A3. Matrix representation for the rule-base of problem Roszieg with 1 OR, 2ORs and 3 ORs

ROSZIEG 1 OR

1	**1**	**2**	**2**	**3**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**10**	**11**	**11**	**11**	**12**	**13**	**13**	**14**	**15**	**16**	**17**	**18**	**18**	**19**	**20**	**21**	**22**	**22**	**22**	**23**	**24**	**25**	**25**	**25**
1	0	2	0	1	2	3	4	5	6	4	8	6	9	7	8	10	7	9	11	13	12	14	15	16	17	14	14	20	15	19	21	17	21	18	20	23
4	8	4	8	0	0	4	3	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROSZIEG 2 ORs

1	**1**	**2**	**2**	**3**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**10**	**11**	**11**	**11**	**12**	**13**	**13**	**14**	**15**	**16**	**17**	**18**	**18**	**19**	**20**	**21**	**22**	**22**	**22**	**23**	**23**	**24**	**25**	**25**	**25**	**25**
1	0	2	0	1	2	3	4	5	6	4	8	6	9	7	8	10	7	9	11	13	12	14	15	16	17	14	14	20	15	19	21	17	0	21	18	20	23	0
4	8	4	8	0	0	4	3	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	22	0	18	20	23	22
0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	20	23	24

ROSZIEG 3 ORs

1	**1**	**2**	**2**	**3**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**10**	**11**	**11**	**11**	**12**	**13**	**13**	**14**	**15**	**16**	**17**	**18**	**18**	**19**	**20**	**21**	**22**	**22**	**22**	**23**	**23**	**24**	**25**	**25**	**25**	**25**
1	0	2	0	1	2	3	4	5	6	4	8	6	9	7	8	10	7	9	11	13	12	14	15	16	17	14	14	20	15	19	21	17	0	21	18	20	23	0
4	8	4	8	0	0	4	3	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	22	0	18	20	23	22
0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	20	23	24
0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	20	23	21

APPENDIX A4. Matrix representation for the rule-base of problem Heskia with 1 OR, 2ORs and 3 ORs

HESKIA 1 OR

3	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**10**	**11**	**12**	**13**	**14**	**15**	**15**	**16**	**17**	**18**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
1	1	1	2	6	1	8	9	0	10	10	12	13	11	0	13	2	7	0	1	19	1	1	1	1	24	1	26	3	4	5	14	15	16	17	18	20	21	22	23	25	27
0	0	0	0	0	0	0	9	6	0	0	0	0	11	13	0	0	7	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HESKIA 2 ORs

3	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**10**	**11**	**12**	**13**	**14**	**15**	**15**	**16**	**17**	**18**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
1	1	1	2	6	1	8	9	0	10	10	12	13	11	0	13	2	7	0	1	19	1	1	1	1	24	1	26	3	4	5	14	15	16	17	18	20	21	22	23	25	27
0	0	0	0	0	0	0	9	6	0	0	0	0	11	13	0	0	7	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	9	7	0	0	0	0	11	12	0	0	7	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HESKIA 3 ORs

3	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**10**	**11**	**12**	**13**	**14**	**15**	**15**	**16**	**17**	**18**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
1	1	1	2	6	1	8	9	0	10	10	12	13	11	0	13	2	7	0	1	19	1	1	1	1	24	1	26	3	4	5	14	15	16	17	18	20	21	22	23	25	27
0	0	0	0	0	0	0	9	6	0	0	0	0	11	13	0	0	7	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	9	7	0	0	0	0	11	12	0	0	7	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	9	0	0	0	0	0	11	19	0	0	7	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APPENDIX A5. Matrix representation for the rule-base of problem Buxey with 1 OR, 2ORs and 3 ORs

BUXEY 1 OR

3	**4**	**5**	**6**	**8**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**15**	**16**	**16**	**17**	**17**	**18**	**19**	**20**	**21**	**22**	**22**	**23**	**23**	**24**	**25**	**25**	**25**	**26**	**27**	**28**	**29**	**29**	**29**	**29**
1	3	4	2	5	6	7	9	8	7	5	10	10	12	8	14	11	13	16	15	17	19	18	21	20	22	23	1	7	0	2	26	23	24	25	27	28
0	0	0	0	4	6	0	0	0	0	0	0	7	12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	10	12	0	0	0	0	0	0	0

BUXEY 2 ORs

3	**4**	**5**	**6**	**8**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**15**	**16**	**16**	**17**	**17**	**18**	**19**	**20**	**21**	**22**	**22**	**23**	**23**	**24**	**25**	**25**	**25**	**26**	**27**	**28**	**29**	**29**	**29**	**29**
1	3	4	2	5	6	7	9	8	7	5	10	10	12	8	14	11	13	16	15	17	19	18	21	20	22	23	1	7	0	2	26	23	24	25	27	28
0	0	0	0	4	6	0	0	0	0	0	0	8	12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	10	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	7	12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8	0	0	0	0	0	0	0	0

BUXEY 3 ORs

3	**4**	**5**	**6**	**8**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**15**	**16**	**16**	**17**	**17**	**18**	**19**	**20**	**21**	**22**	**22**	**23**	**23**	**24**	**25**	**25**	**25**	**26**	**27**	**28**	**29**	**29**	**29**	**29**
1	3	4	2	5	6	7	9	8	7	5	10	10	12	8	14	11	13	16	15	17	19	18	21	20	22	23	1	7	0	2	26	23	24	25	27	28
0	0	0	0	4	6	0	0	0	0	0	0	8	12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	10	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	7	12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	11	12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	15	0	0	0	0	0	0	0	0

APPENDIX A6. Matrix representation for the rule-base of problem Sawyer with 1 OR, 2ORs and 3 ORs

SAWYER 1 OR

4	**5**	**6**	**7**	**7**	**8**	**9**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**20**	**21**	**22**	**22**	**23**	**24**	**24**	**25**	**26**	**26**	**26**	**27**	**27**	**28**	**28**	**29**	**30**
1	1	5	4	6	7	8	2	2	12	13	14	3	3	17	18	14	16	20	15	21	22	10	20	24	9	25	0	23	26	27	0	27	29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	16	0	0	0	0	0	0	0	9	25	11	0	0	27	19	0	0

SAWYER 2 ORs

4	**5**	**6**	**7**	**7**	**8**	**9**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**20**	**21**	**22**	**22**	**23**	**24**	**24**	**25**	**26**	**26**	**26**	**27**	**27**	**28**	**28**	**29**	**30**
1	1	5	4	6	7	8	2	2	12	13	14	3	3	17	18	14	16	20	15	21	22	10	20	24	9	25	0	23	26	27	0	27	29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	16	0	0	0	0	0	0	0	9	25	11	0	0	27	19	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	16	0	0	0	0	0	0	0	9	25	19	0	0	27	11	0	0

SAWYER 3 ORs

4	**5**	**6**	**7**	**7**	**8**	**9**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**20**	**21**	**22**	**22**	**23**	**24**	**24**	**25**	**26**	**26**	**26**	**27**	**27**	**28**	**28**	**29**	**30**
1	1	5	4	6	7	8	2	2	12	13	14	3	3	17	18	14	16	20	15	21	22	10	20	24	9	25	0	23	26	27	0	27	29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	16	0	0	0	0	0	0	0	9	25	11	0	0	27	19	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	16	0	0	0	0	0	0	0	9	25	19	0	0	27	11	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	16	0	0	0	0	0	0	0	9	25	23	0	0	27	26	0	0

APPENDIX A7. Matrix representation for the rule-base of problem Kilbridge with 1 OR, 2ORs and 3 ORs

KILBRIDGE 1 OR

3	**4**	**5**	**6**	**7**	**8**	**9**	**9**	**10**	**10**	**13**	**13**	**14**	**14**	**14**	**15**	**16**	**17**	**17**	**18**	**19**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**26**	**26**	**27**	**28**	**28**	**29**	**30**	**31**	**32**
1	2	3	4	1	2	5	7	6	8	11	12	7	8	13	13	15	14	0	15	16	18	19	20	21	15	15	14	17	25	0	17	22	27	14	14	14	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	14	0	0	0	0	0	0	0	0	0	10	17	25	0	0	0	0	0	0	0

33	**33**	**33**	**33**	**33**	**34**	**35**	**36**	**37**	**38**	**38**	**38**	**38**	**40**	**40**	**40**	**41**	**41**	**41**	**41**	**41**	**41**	**41**	**41**	**42**	**42**	**43**	**44**	**45**
19	23	24	27	0	33	33	33	12	26	28	34	36	35	38	0	9	10	29	30	31	32	39	40	41	0	37	42	42
19	23	24	27	43	0	0	0	0	0	0	0	0	35	38	43	0	0	0	0	0	0	0	0	41	43	0	0	0

KILBRIDGE 2 ORs

3	**4**	**5**	**6**	**7**	**8**	**9**	**9**	**10**	**10**	**13**	**13**	**14**	**14**	**14**	**15**	**16**	**17**	**17**	**18**	**19**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**26**	**26**	**27**	**28**	**28**	**29**	**30**	**31**	**32**
1	2	3	4	1	2	5	7	6	8	11	12	7	8	13	13	15	14	0	15	16	18	19	20	21	15	15	14	17	25	0	17	22	27	14	14	14	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	14	0	0	0	0	0	0	0	0	0	10	17	25	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	14	0	0	0	0	0	0	0	0	0	9	17	25	0	0	0	0	0	0	0

33	**33**	**33**	**33**	**33**	**34**	**35**	**36**	**37**	**38**	**38**	**38**	**38**	**40**	**40**	**40**	**41**	**41**	**41**	**41**	**41**	**41**	**41**	**41**	**42**	**42**	**43**	**44**	**45**
19	23	24	27	0	33	33	33	12	26	28	34	36	35	38	0	9	10	29	30	31	32	39	40	41	0	37	42	42
19	23	24	27	43	0	0	0	0	0	0	0	0	35	38	43	0	0	0	0	0	0	0	0	41	43	0	0	0
19	23	24	27	37	0	0	0	0	0	0	0	0	35	38	37	0	0	0	0	0	0	0	0	41	37	0	0	0

KILBRIDGE 3 ORs

3	**4**	**5**	**6**	**7**	**8**	**9**	**9**	**10**	**10**	**13**	**13**	**14**	**14**	**14**	**15**	**16**	**17**	**17**	**18**	**19**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**26**	**26**	**27**	**28**	**28**	**29**	**30**	**31**	**32**
1	2	3	4	1	2	5	7	6	8	11	12	7	8	13	13	15	14	0	15	16	18	19	20	21	15	15	14	17	25	0	17	22	27	14	14	14	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	14	0	0	0	0	0	0	0	0	0	10	17	25	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	14	0	0	0	0	0	0	0	0	0	9	17	25	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	14	0	0	0	0	0	0	0	0	0	15	17	25	0	0	0	0	0	0	0

33	**33**	**33**	**33**	**33**	**34**	**35**	**36**	**37**	**38**	**38**	**38**	**38**	**40**	**40**	**40**	**41**	**41**	**41**	**41**	**41**	**41**	**41**	**41**	**42**	**42**	**43**	**44**	**45**
19	23	24	27	0	33	33	33	12	26	28	34	36	35	38	0	9	10	29	30	31	32	39	40	41	0	37	42	42
19	23	24	27	43	0	0	0	0	0	0	0	0	35	38	43	0	0	0	0	0	0	0	0	41	43	0	0	0
19	23	24	27	37	0	0	0	0	0	0	0	0	35	38	37	0	0	0	0	0	0	0	0	41	37	0	0	0
19	23	24	27	9	0	0	0	0	0	0	0	0	35	38	22	0	0	0	0	0	0	0	0	41	38	0	0	0

APPENDIX A8. Matrix representation for the rule-base of problem Arcus 83 with 1 OR, 2ORs and 3 ORs

ARCUS 1 OR

2	**3**	**4**	**5**	**6**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**27**	**28**	**28**	**29**	**30**	**31**	**32**	**32**	**33**	**34**	**35**
1	2	2	2	3	4	4	5	6	6	7	9	10	10	11	12	13	13	14	13	15	17	17	18	19	10	20	21	22	17	24	24	29	30	25	28	32	32	32
0	0	0	0	12	16	0	0	0	0	0	3	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

36	**37**	**38**	**39**	**39**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**69**	**70**
32	33	36	15	31	36	37	38	39	39	39	41	44	45	46	47	48	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	27	49	69
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

71	**72**	**73**	**73**	**74**	**74**	**74**	**74**	**75**	**75**	**75**	**76**	**76**	**77**	**77**	**77**	**77**	**77**	**77**	**77**	**77**	**78**	**78**	**78**	**78**	**78**	**78**	**78**	**78**	**79**	**79**	**80**	**81**	**82**	**83**	**83**
69	70	71	72	23	26	68	73	39	68	73	74	75	8	16	34	35	40	42	43	76	8	16	34	35	40	42	43	76	77	78	79	79	80	81	82
0	0	0	0	0	0	0	0	0	0	0	0	0	8	9	34	35	40	42	43	76	8	9	34	35	40	42	43	76	0	0	0	0	0	0	0

ARCUS 2 ORs

2	**3**	**4**	**5**	**6**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**27**	**28**	**28**	**29**	**30**	**31**	**32**	**32**	**33**	**34**	**35**
1	2	2	2	3	4	4	5	6	6	7	9	10	10	11	12	13	13	14	13	15	17	17	18	19	10	20	21	22	17	24	24	29	30	25	28	32	32	32
0	0	0	0	12	16	0	0	0	0	9	3	12	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

36	**37**	**38**	**39**	**39**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**69**	**70**
32	33	36	15	31	36	37	38	39	39	39	41	44	45	46	47	48	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	27	49	69
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

71	**72**	**73**	**73**	**74**	**74**	**74**	**74**	**75**	**75**	**75**	**76**	**76**	**77**	**77**	**77**	**77**	**77**	**77**	**77**	**77**	**78**	**78**	**78**	**78**	**78**	**78**	**78**	**78**	**79**	**79**	**80**	**81**	**82**	**83**	**83**
69	70	71	72	23	26	68	73	39	68	73	74	75	8	16	34	35	40	42	43	76	8	16	34	35	40	42	43	76	77	78	79	79	80	81	82
0	0	0	0	0	0	0	0	0	0	0	0	0	8	9	34	35	40	42	43	76	8	9	34	35	40	42	43	76	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ARCUS 3 ORs

2	**3**	**4**	**5**	**6**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**27**	**28**	**28**	**29**	**30**	**31**	**32**	**32**	**33**	**34**	**35**
1	2	2	2	3	4	4	5	6	6	7	9	10	10	11	12	13	13	14	13	15	17	17	18	19	10	20	21	22	17	24	24	29	30	25	28	32	32	32
0	0	0	0	12	16	0	0	0	0	9	3	12	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

36	**37**	**38**	**39**	**39**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**69**	**70**
32	33	36	15	31	36	37	38	39	39	39	41	44	45	46	47	48	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	27	49	69
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

71	**72**	**73**	**73**	**74**	**74**	**74**	**74**	**75**	**75**	**75**	**76**	**76**	**77**	**77**	**77**	**77**	**77**	**77**	**77**	**77**	**78**	**78**	**78**	**78**	**78**	**78**	**78**	**78**	**79**	**79**	**80**	**81**	**82**	**83**	**83**
69	70	71	72	23	26	68	73	39	68	73	74	75	8	16	34	35	40	42	43	76	8	16	34	35	40	42	43	76	77	78	79	79	80	81	82
0	0	0	0	0	0	0	0	0	0	0	0	0	8	9	34	35	40	42	43	76	8	9	34	35	40	42	43	76	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APPENDIX A9. Matrix representation for the rule-base of real-case problem without OR and with 1 OR, 2ORs and 3 ORs

REAL-CASE WITHOUT OR

2	**3**	**4**	**5**	**6**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**22**	**23**	**24**	**24**	**24**	**25**	**26**	**26**	**27**	**27**	**28**	**29**	**30**	**31**
1	2	1	1	4	5	6	1	8	9	1	11	12	3	1	1	16	1	18	19	20	10	21	22	60	62	14	24	68	23	25	26	27	1	29	1

32	**32**	**32**	**33**	**34**	**34**	**35**	**36**	**36**	**37**	**38**	**39**	**40**	**41**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**
13	30	31	32	33	67	34	28	35	36	37	38	7	39	40	41	42	43	44	45	46	47	48	17	49	50	51	52	1	54	55	1	57	58	1

61	**62**	**63**	**64**	**65**	**66**	**66**	**66**	**67**	**68**
56	61	1	63	1	59	64	65	66	15

REAL-CASE 1 OR

2	**3**	**4**	**5**	**6**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**22**	**23**	**24**	**24**	**24**	**25**	**26**	**26**	**27**	**27**	**27**	**28**	**29**	**30**
1	2	1	1	4	5	6	1	8	9	1	11	12	3	1	1	16	1	18	19	20	10	21	22	60	62	14	24	68	23	25	26	0	27	1	29
0	0	0	0	0	0	0	10	0	1	0	0	0	0	0	0	0	0	0	0	0	9	21	0	0	0	0	0	15	23	25	26	68	0	0	0

31	**32**	**32**	**32**	**33**	**34**	**34**	**35**	**36**	**36**	**37**	**38**	**39**	**40**	**41**	**41**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**
1	13	30	31	32	33	67	34	28	35	36	37	36	7	39	40	0	41	42	43	44	45	46	47	48	17	49	50	51	52	1	54	55	1	57
0	0	0	0	0	0	0	0	0	0	39	36	38	0	38	40	37	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	1	0	0

59	**60**	**61**	**62**	**63**	**64**	**65**	**66**	**66**	**66**	**67**	**68**
58	1	56	61	1	63	1	59	64	65	66	15
0	0	55	0	0	0	0	0	0	0	0	0

REAL-CASE 2 ORs

2	**3**	**4**	**5**	**6**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**22**	**23**	**24**	**24**	**24**	**25**	**26**	**26**	**27**	**27**	**27**	**28**	**29**	**30**
1	2	1	1	4	5	6	1	8	9	1	11	12	3	1	1	16	1	18	19	20	10	21	22	60	62	14	24	68	23	25	26	0	27	1	29
0	0	0	0	0	0	0	10	54	1	0	0	0	0	0	0	0	0	0	0	0	9	21	0	0	0	0	0	15	23	25	26	68	0	0	0
0	0	0	0	0	0	0	0	0	55	0	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	0	0	0	0	0	0	0

31	**32**	**32**	**32**	**33**	**34**	**34**	**35**	**36**	**36**	**37**	**38**	**39**	**40**	**41**	**41**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**
1	13	30	31	32	33	67	34	28	35	36	37	36	7	39	40	0	41	42	43	44	45	46	47	48	17	49	50	51	52	1	54	55	1	57
0	0	0	0	0	0	0	0	0	0	39	36	38	0	38	40	37	0	0	0	0	0	0	0	0	0	0	0	0	0	56	9	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	10	0	0

59	**60**	**61**	**62**	**63**	**64**	**65**	**66**	**66**	**66**	**67**	**68**
58	1	56	61	1	63	1	59	64	65	66	15
0	0	55	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

REAL-CASE 3 ORs

2	**3**	**4**	**5**	**6**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**22**	**23**	**24**	**24**	**24**	**25**	**26**	**26**	**27**	**27**	**27**	**28**	**29**	**30**
1	2	1	1	4	5	6	1	8	9	1	11	12	3	1	1	16	1	18	19	20	10	21	22	60	62	14	24	68	23	25	26	0	27	1	29
0	0	0	0	0	0	0	10	54	1	0	0	0	0	0	0	0	0	0	0	0	9	21	0	0	0	0	0	15	23	25	26	68	0	0	0
0	0	0	0	0	0	0	54	55	55	0	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31	**32**	**32**	**32**	**33**	**34**	**34**	**35**	**36**	**36**	**37**	**38**	**39**	**40**	**41**	**41**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**
1	13	30	31	32	33	67	34	28	35	36	37	36	7	39	40	0	41	42	43	44	45	46	47	48	17	49	50	51	52	1	54	55	1	57
0	0	0	0	0	0	0	0	0	0	39	36	38	0	38	40	37	0	0	0	0	0	0	0	0	0	0	0	0	0	56	9	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	10	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0

59	**60**	**61**	**62**	**63**	**64**	**65**	**66**	**66**	**66**	**67**	**68**
58	1	56	61	1	63	1	59	64	65	66	15
0	0	55	0	0	0	0	0	0	0	0	0
0	0	10	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

APPENDIX A10. The results of the experiments as efficiency.

		Efficiencies (%)	
Problem	**Cycle time**	**WITHOUT OR**	**WITH 1 OR**
Bowman	20	88	88
Jaeschke	6	77	77
	7	76	77
	8	76	77
	10	93	93
	18	73	88
	Average Efficiency	**79**	**82**
	Standard Deviation	**8**	**7**
Jackson	7	82	94
	9	85	85
	10	92	92
	13	96	96
	14	96	96
	21	96	96
	Average Efficiency	**91**	**93**
	Standard Deviation	**6**	**4**

		Efficiencies (%)			
Problem	Cycle time	WITHOUT OR	WITH 1 OR	WITH 2 ORs	WITH 3 ORs
Mitchell	14	83	94	94	94
	15	94	94	94	94
	21	97	97	97	100
	26	91	100	95	95
	35	100	100	100	100
	39	100	100	100	100
	Average Efficiency	**94**	**97**	**97**	**97**
	Standard Deviation	**6**	**3**	**3**	**3**
Roszieg	18	92	98	98	99
	21	89	99	99	99
	25	99	99	100	100
	32	98	98	98	98
	Average Efficiency	**95**	**98**	**99**	**99**
	Standard Deviation	**5**	**1**	**1**	**1**
Heskiaoff	138	99	99	99	99
	205	99	99	99	100
	216	100	99	100	100
	256	98	99	100	100
	324	100	100	100	100
	342	100	100	100	100
	Average Efficiency	**99**	**99**	**100**	**100**
	Standard Deviation	**1**	**0**	**0**	**0**
Buxey	27	92	93	93	92
	30	93	92	89	93
	33	93	92	96	95
	36	95	95	95	95
	41	95	95	92	97
	47	99	99	99	99
	54	94	96	96	96
	Average Efficiency	**95**	**95**	**94**	**95**
	Standard Deviation	**2**	**2**	**3**	**2**

Problem	Cycle time	Efficiencies (%) WITHOUT OR	WITH 1 OR	WITH 2 ORs	WITH 3 ORs
Sawyer	30	93	93	93	93
	33	93	92	92	92
	36	92	92	95	95
	41	97	92	92	95
	47	99	99	99	99
	54	96	96	96	96
	75	98	100	98	100
	Average Efficiency	**95**	**95**	**95**	**96**
	Standard Deviation	**3**	**3**	**3**	**3**
Kilbridge	69	97	97	99	99
	79	99	99	100	100
	92	99	99	100	100
	110	99	99	100	100
	111	100	100	99	99
	138	100	100	100	100
	184	100	100	100	100
	Average Efficiency	**99**	**99**	**100**	**100**
	Standard Deviation	**1**	**1**	**0**	**0**
Arcus	5853	96	97	99	99
	6309	98	94	94	97
	6842	96	97	96	96
	6883	94	98	92	97
	7571	97	97	97	99
	8412	98	98	97	99
	8898	99	98	99	99
	10816	94	96	99	97
	Average Efficiency	**96**	**97**	**97**	**98**
	Standard Deviation	**2**	**2**	**3**	**1**

APPENDIX A11. The results of the experiments as number of stations.

		Number of stations	
Problem	**Cycle time**	**WITHOUT OR**	**WITH 1 OR**
Bowman	20	5	5
Jaeschke	6	8	8
	7	7	8
	8	7	8
	10	4	4
	18	3	3
Jackson	7	8	7
	9	6	6
	10	5	5
	13	4	4
	14	4	4
	21	3	3
	number of better solutions		**1**

Problem	Cycle time	Number of stations WITHOUT OR	WITH 1 OR	WITH 2 ORs	WITH 3 ORs
Mitchell	14	9	8	8	8
	15	8	8	8	8
	21	6	6	6	5
	26	5	5	5	5
	35	3	3	3	3
	39	3	3	3	3
	number of better solutions		**1**	**1**	**2**
Roszieg	18	8	8	8	7
	21	7	6	6	6
	25	6	6	5	5
	32	4	4	4	4
	number of better solutions		**1**	**2**	**3**
Heskiaoff	138	8	8	8	8
	205	6	6	6	6
	216	5	5	5	5
	256	5	5	5	5
	324	4	4	4	4
	342	3	4	3	3
	number of better solutions		**-**	**-**	**-**
Buxey	27	13	14	14	13
	30	12	13	14	12
	33	12	11	12	11
	36	10	10	10	10
	41	9	9	9	9
	47	8	8	8	8
	54	7	7	7	7
	number of better solutions		**-**	**-**	**1**

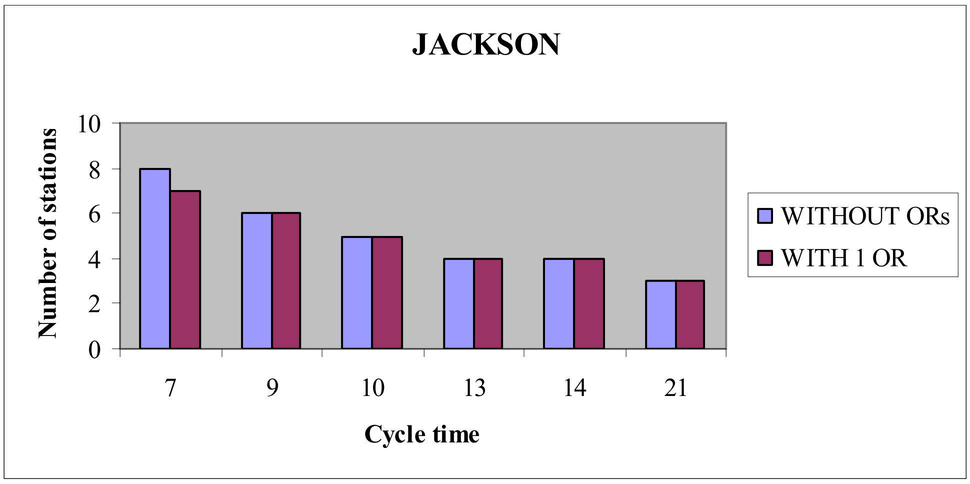
		Number of stations			
Problem	Cycle time	WITHOUT OR	WITH 1 OR	WITH 2 ORs	WITH 3 ORs
Sawyer	30	14	14	12	12
	33	12	11	11	11
	36	11	11	10	10
	41	9	9	9	9
	47	8	8	8	8
	54	7	7	7	7
	75	5	5	5	5
	number of better solutions		**1**	**3**	**3**
Kilbridge	69	9	9	9	9
	79	8	8	8	8
	92	7	7	7	7
	110	6	6	6	6
	111	6	6	5	5
	138	4	4	4	4
	184	3	3	3	3
	number of better solutions		-	-	-
Arcus	5853	14	14	14	14
	6309	13	14	14	13
	6842	13	12	12	12
	6883	13	12	13	12
	7571	11	11	11	11
	8412	10	10	10	10
	8898	9	9	9	9
	10816	8	8	8	8
	number of better solutions		**1**	**1**	**2**

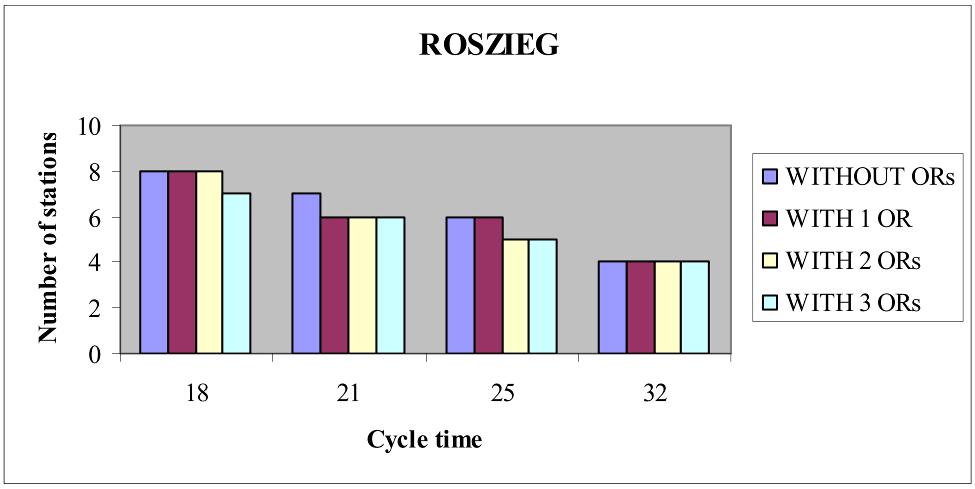
APPENDIX A12. The results of the experiments as fitness value.

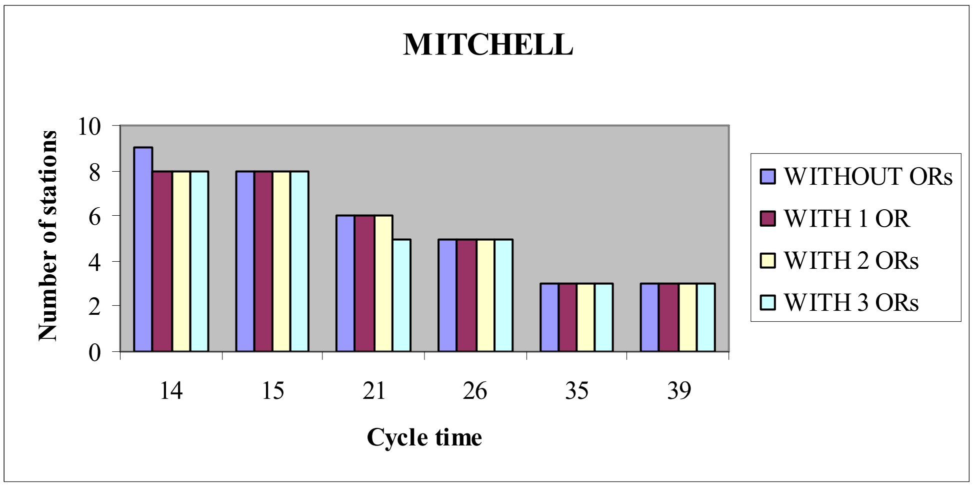
		Fitness values			
		WITHOUT	**WITH**	**WITH**	**WITH**
Problem	**Cycle time**	**OR**	**1 OR**	**2 ORs**	**3 ORs**
Mitchell	14	2.8708	1.0871	1.1095	1.0871
	15	1.0871	1.0871	1.0871	1.0871
	21	0.6414	0.6414	0.6414	0
	26	2.5215	0	1.2366	1.2366
	35	0	0	0	0
	39	0	0	0	0
	number of better solutions		**2**	**2**	**3**
Roszieg	18	1.699	0.4975	0.4975	0.2185
	21	2.6269	0.2483	0.2483	0.2483
	25	0.2483	0.2483	0	0
	32	0.9736	0.9232	0.9232	0.9232
	number of better solutions		**3**	**4**	**4**
Heskiaoff	138	1.2449	1.2449	1.2449	1.2449
	205	1.6388	2.8871	1.6797	0.4966
	216	0.2894	1.4828	0.2894	0.2894
	256	3.9155	1.4828	0.2894	0.2894
	324	1.2449	1.2449	0	0
	342	0.83	1.2449	0.83	0.83
	number of better solutions		**1**	**2**	**3**
Buxey	27	2.6507	2.3528	2.3352	2.6507
	30	2.5033	2.6666	3.5248	2.4899
	33	2.5292	3.1363	1.3162	2.0316
	36	2	1.9688	2.0099	1.9795
	41	2.5333	3.7149	3.7087	1.3944
	47	0.6732	0.6414	0.6732	0.6414
	54	3.4433	2.28	2.1283	2.28
	number of better solutions		**3**	**1**	**6**

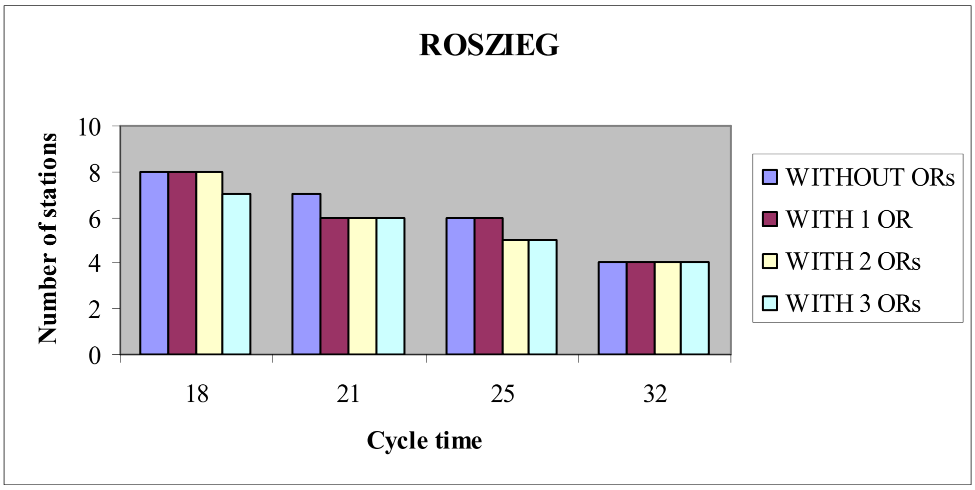
		Fitness values			
		WITHOUT	WITH	WITH	WITH
Problem	**Cycle time**	**OR**	**1 OR**	**2 ORs**	**3 ORs**
Sawyer	30	2.3642	2.2914	2.5033	2.4690
	33	2.5292	3.2986	3.2889	3.3177
	36	3.1836	3.284	1.9899	1.9688
	41	2.5333	4.1294	3.7149	2.4899
	47	0.6732	0.6732	0.6414	0.6414
	54	2.1283	2.1143	2.1283	2.1143
	75	1.5347	0.2894	1.4828	0.2894
	number of better solutions		**1**	**3**	**5**
Kilbridge	69	2.0496	2.0496	0.83	0.83
	79	1.2236	1.2236	0	0
	92	1.4047	1.4047	0.2185	0.2185
	110	1.2309	1.2309	0	0
	111	0	0	0.7549	0.7549
	138	0	0	0	0
	184	0	0	0	0
	number of better solutions		**-**	**4**	**4**
Arcus	5853	264.5	195.4	73	75.7
	6309	186.2	467.8	443.3	201.8
	6842	351.7	304.4	348.6	305
	6883	488.7	194.3	649.8	244.4
	7571	258.5	272.3	257.3	265.5
	8412	194.6	157.9	255.6	113.3
	8898	143.8	164.7	164.4	140.9
	10816	728	500.5	184.4	376.5
	number of better solutions		**4**	**2**	**6**

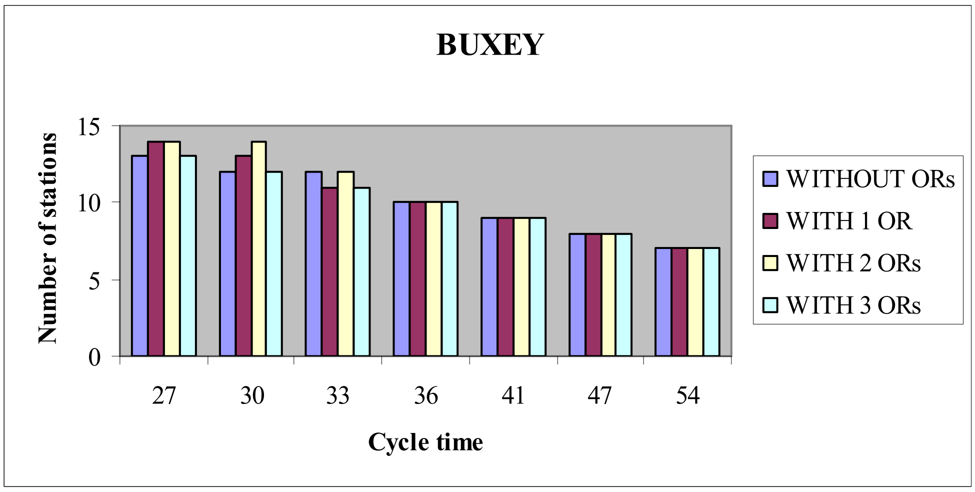
APPENDIX A13. Graphical representation for the results of the experiments as number of stations.

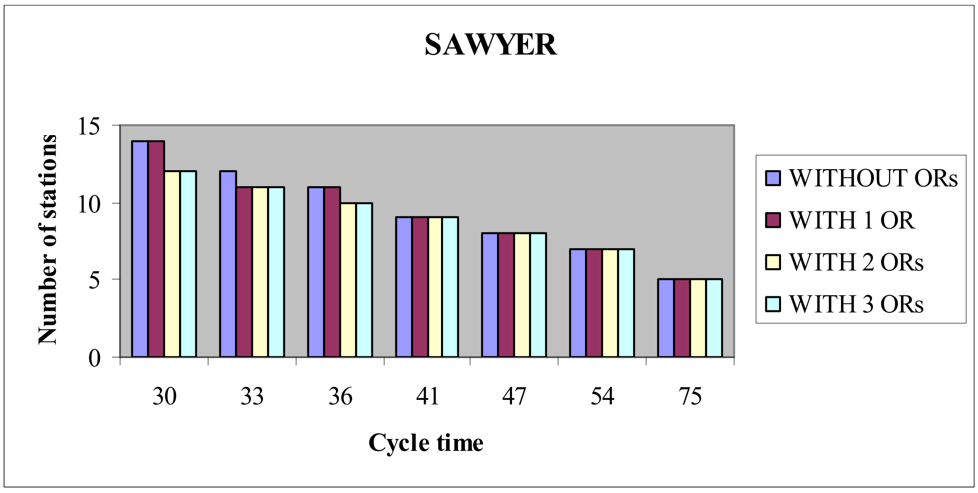


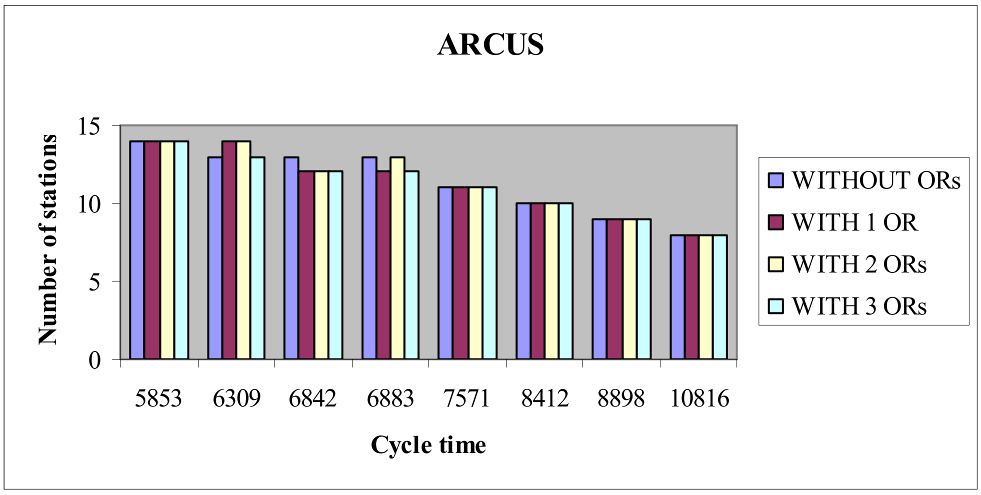




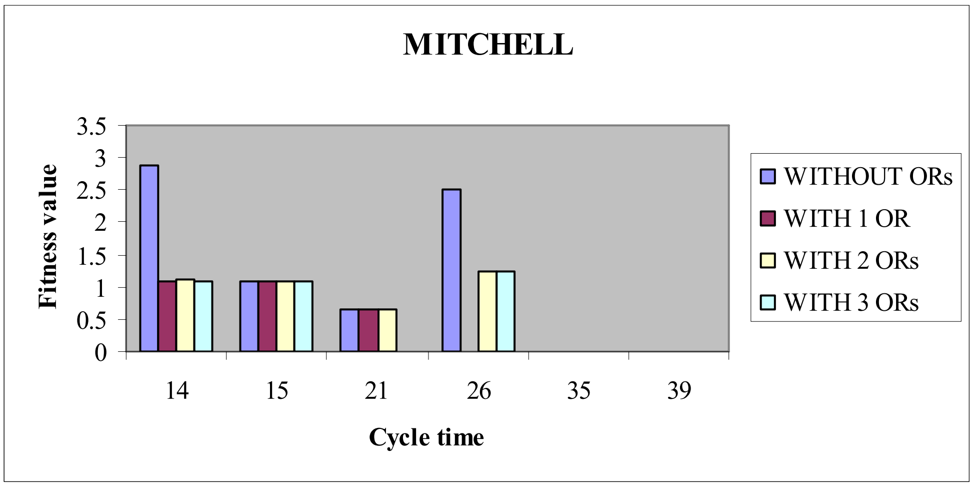

ROSZIEG
Number of stations
10
8
6
4
2
0
18
21
25
32
Cycle time
WITHOUT ORs
WITH 1 OR
WITH 2 ORs
WITH 3 ORs

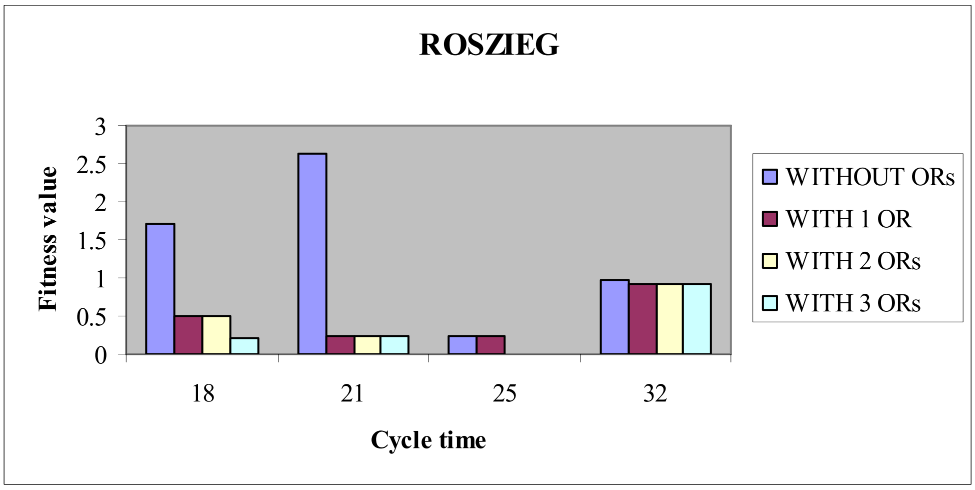

BUXEY
Number of stations
15
10
5
0
27
30
33
36
41
47
54
Cycle time
WITHOUT ORs
WITH 1 OR
WITH 2 ORs
WITH 3 ORs

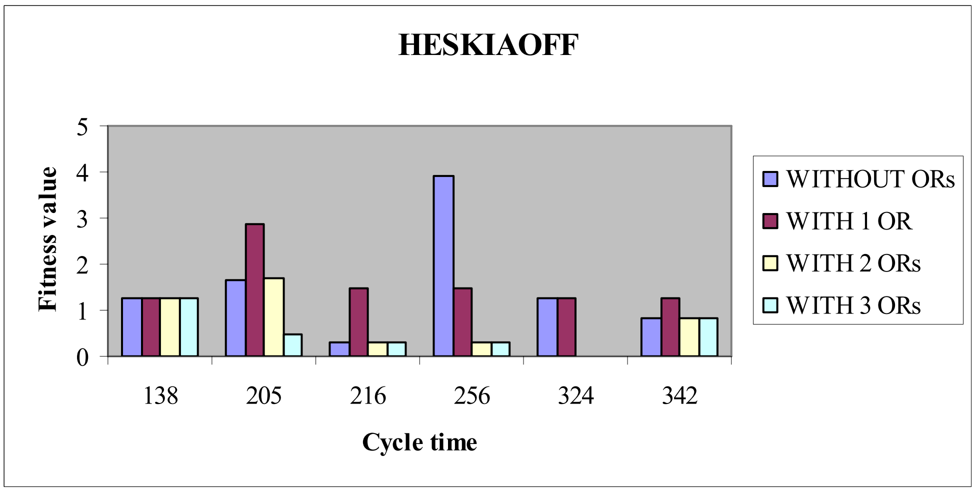

SAWYER
Number of stations
15
10
5
0
30
33
36
41
47
54
75
Cycle time
WITHOUT ORs
WITH 1 OR
WITH 2 ORs
WITH 3 ORs

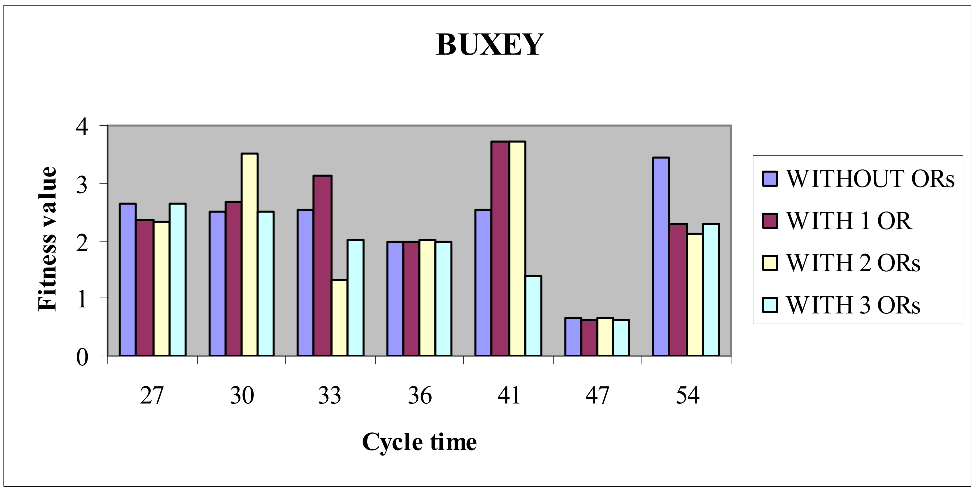

ARCUS
Number of stations
15
10
5
0
5853
6309
6842
6883
7571
8412
8898
10816
Cycle time
WITHOUT ORs
WITH 1 OR
WITH 2 ORs
WITH 3 ORs

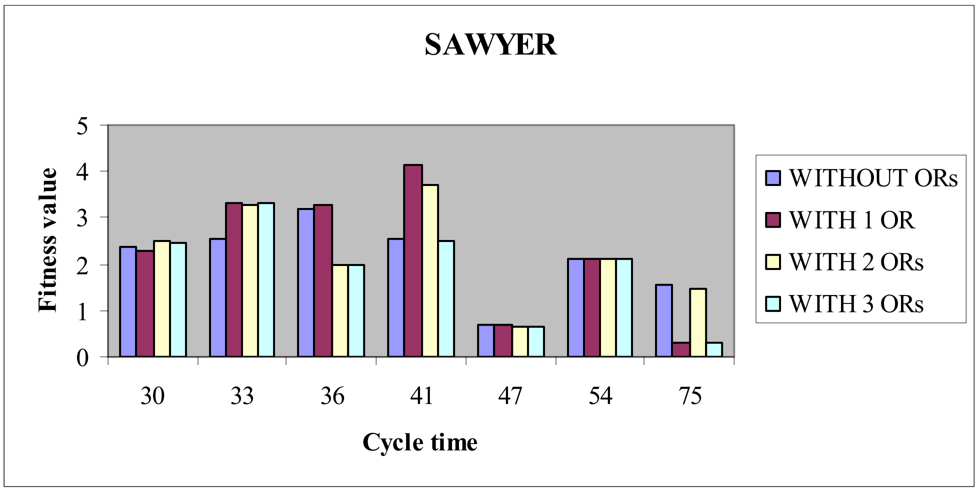
APPENDIX A14. Graphical representation for the results of the experiments as fitness value.

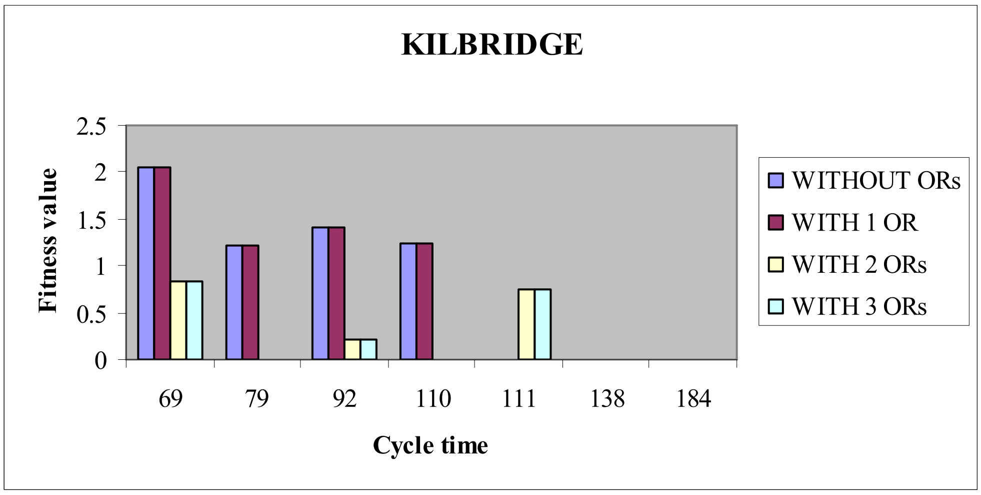


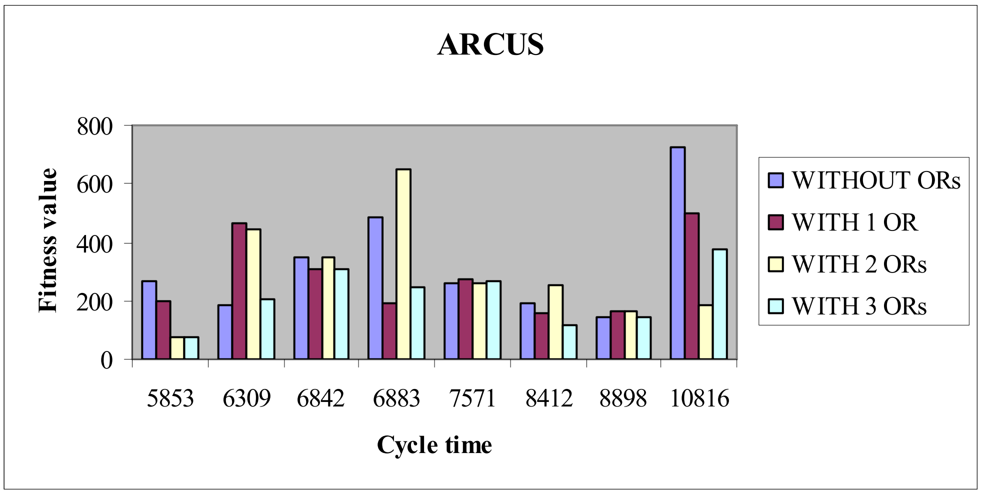




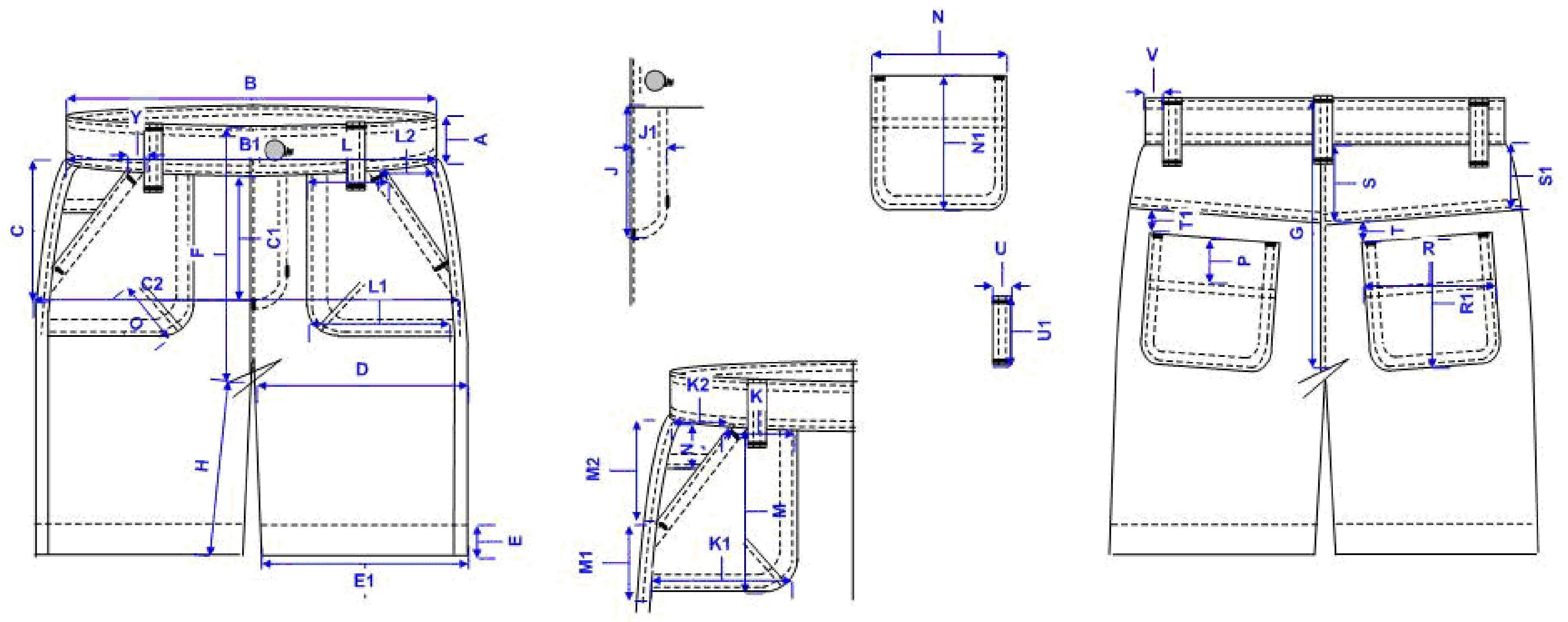

BUXEY
Fitness value
0
1
2
3
4
27
30
33
36
41
47
54
Cycle time
WITHOUT ORs
WITH 1 OR
WITH 2 ORs
WITH 3 ORs


SAWYER
Fitness value
0
1
2
3
4
5
30
33
36
41
47
54
75
Cycle time
WITHOUT ORs
WITH 1 OR
WITH 2 ORs
WITH 3 ORs


KILBRIDGE
Fitness value
0
0.5
1
1.5
2
2.5
69
79
92
110
111
138
184
Cycle time
WITHOUT ORs
WITH 1 OR
WITH 2 ORs
WITH 3 ORs


ARCUS
Fitness value
800
600
400
200
0
5853
6309
6842
6883
7571
8412
8898
10816
Cycle time
WITHOUT ORs
WITH 1 OR
WITH 2 ORs
WITH 3 ORs

APPENDIX A15. Short model for the real-case problem



APPENDIX A16. Data of short model for the real-case problem

NO	OPERATIONS	MACHINE TYPES	STANDARD TIMES (Heim)
1	Preparation	Hand made	40
2	Left fly interlining	Fusing press machine	6
3	Pick left fly	Hand made	5
4	Waistband interlining attachment	Fusing press machine	12
5	Waistband lining interlining attachment	Fusing press machine	12
6	Assembling waistband and waistband lining	Lock-stitch sewing machine	30
7	Waistband ironing	Hand made	30
8	Right front pocket sason stitching and top stitching	Lock-stitch sewing machine	18
9	Right front pocket ironing	Hand made	35
10	Right front pocket edge top stitching	Two needle sewing machine	22
11	Right back pocket edge overlock	Three thread overlock machine	5
12	Right back pocket ironing	Hand made	30
13	Right back pocket twin needle seam	Two needle sewing machine	15
14	Left fly overlock	Three thread overlock machine	10
15	Right fly edge overlock	Three thread overlock machine	10
16	Belt loop preparation	Two needle sewing machine	15
17	Belt loop cutting	Hand made	15
18	Right front part crotch overlock	Three thread overlock machine	13
19	Attach right front pocket to right front part	Lock-stitch sewing machine	25
20	Assembling right front pocket and right front part	Three thread overlock machine	15
21	Right front pocket with right front part top stitching	Two needle sewing machine	20
22	Attach right front pocket to right front part	Lock-stitch sewing machine	22
23	Assembling right front pocket to right front part	Two needle sewing machine	35
24	Assembling left fly to left front part and top stitching	Lock-stitch sewing machine	18
25	Zipper assembling to left fly	Two needle sewing machine	30
26	Zipper assembling to right fly and right front part	Lock-stitch sewing machine	45
27	Fly top stitching	Two needle sewing machine	30
28	Assembling front center	Lock-stitch sewing machine	50
29	Assembling right back parts	Five thread overlock machine	18
30	Right back parts top stitching	Two needle sewing machine	20
31	Marking the place of right back pocket	Hand made	20
32	Assembling right back pocket to right back part	Two needle sewing machine	40
33	Right back pocket bartacking	Bartack machine	14
34	Back center overlock	Five thread overlock machine	30
35	Back center top stitching	Lock-stitch sewing machine	30
36	Matching front part and back part	Hand made	20
37	Side seam	Five thread overlock machine	45
38	Side top stitching	Lock-stitch sewing machine	35
39	Inside leg stitch	Five thread overlock machine	35
40	Cutting waistband	Hand made	3
41	Waistband attaching	Lock-stitch sewing machine	67
42	Waistband edge stitching	Lock-stitch sewing machine	30
43	Inside out waistband	Hand made	20
44	Washing label assembling waistband	Lock-stitch sewing machine	25
45	Branded label assembling above the right back pocket	Lock-stitch sewing machine	35
46	Waistband top stitching	Two needle sewing machine	70

NO	OPERATIONS	MACHINE TYPES	STANDARD TIMES (Heim)
47	Closing waistband	Two needle sewing machine	50
48	Inside out trouser	Hand made	20
49	Leg hem stitch	Lock-stitch sewing machine	50
50	Belt loop bartacking	Bartack machine	63
51	Button hole	Button hole machine	10
52	Button sewing	Button sewing machine	20
53	Pocket, fly and side bartacking	Bartack machine	50
54	Left front pocket sason stitching and top stitching	Lock-stitch sewing machine	18
55	Left front pocket ironing	Hand made	35
56	Left front pocket edge top stitching	Two needle sewing machine	22
57	Left back pocket edge overlock	Three thread overlock machine	5
58	Left back pocket ironing	Hand made	30
59	Left back pocket twin needle seam	Two needle sewing machine	15
60	Left front part crotch overlock	Three thread overlock machine	13
61	Attach left front pocket to left front part	Lock-stitch sewing machine	22
62	Assembling left front pocket to left front part	Two needle sewing machine	35
63	Assembling left back parts	Five thread overlock machine	18
64	Left back parts top stitching	Two needle sewing machine	20
65	Marking the place of left back pocket	Hand made	20
66	Assembling left back pocket to left back part	Two needle sewing machine	40
67	Left back pocket bartacking	Bartack machine	14
68	Right fly overlock	Three thread overlock machine	13
		Total	1753

(1 Heim=0.01 minute)