DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

A HBYRID GENETIC ALGORITHM FOR MIXED-MODEL ASSEMBLY LINE BALANCING PROBLEM WITH PARALLEL WORKSTATION ASSIGNMENT

by
Şener AKPINAR

June, 2009
İZMİR

A HBYRID GENETIC ALGORITHM FOR MIXED-MODEL ASSEMBLY LINE BALANCING PROBLEM WITH PARALLEL WORKSTATION ASSIGNMENT

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

by
Şener AKPINAR

June,2009
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"A HYBRID GENETIC ALGORTIHM FOR MIXED-MODEL ASSEMBLY LINE BALANCING PROBLEM WITH PARALLEL WORKSTATION ASSIGNMENT"** completed by **ŞENER AKPINAR** under supervision of **PROF. DR. G. MİRAÇ BAYHAN** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Prof. Dr. G. Miraç BAYHAN

Supervisor

.....

Doç. Dr. Evren TOYGAR

(Jury Member)

.....

Yrd. Doç. Dr. Mehmet ÇAKMAKÇI

(Jury Member)

Prof.Dr. Cahit HELVACI

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

First, I would like to point out my gratitude to my advisor Prof. Dr. G. Miraç BAYHAN for her guidance, continuing support, encouragement and invaluable advice throughout the progress of this master thesis. Her advices always gave me the direction during the research.

I am truly grateful Dr.Ana Sofia Simaria from Departamento de Economia, Gestao e Engenharia Industrial, Universidade de Aveiro for providing the benchmark data and for her suggestions.

I would like to introduce my great thanks to my friends for their support, whenever I need, and listening to my complaints during this period.

Last, but the most, I would like to emphasize my thankfullnes to my family, especially to my mother, because of their love, confidence, encouragement and endless support in my whole life.

Şener AKPINAR

A HBYRID GENETIC ALGORITHM FOR MIXED-MODEL ASSEMBLY LINE BALANCING PROBLEM WITH PARALLEL WORKSTATION ASSIGNMENT

ABSTRACT

In this thesis, we deal with the mixed-model assembly line balancing problem (MMALBP) of type-1. This problem consists of finding the minimum number of stations for a predetermined cycle time. Various exact and approximation approaches have been developed to deal with MMALBP of type-1. Due to the NP-hard structure of the problem none of the optimum seeking methods has been proven to be practical to solve large scale problems. Moreover, approximation methods may lack the capability of exploring the solution space effectively. Over the last years, hybrid meta-heuristics which combine the various algorithmic ideas of meta-heuristics concerning overcome these shortages have been reported.

In this thesis, we propose an effective hybrid genetic algorithm (GA) that is able to address some particular features such as parallel workstations and zoning constraints of the MMALBP of type-1. The type of hybridization is sequential. For the hybridization of GA three well known heuristics, Kilbridge and Wester, Phase-I of Moodie and Young, and Ranked Positional Weight Technique are used. The original versions of first two methods only address the simple assembly line balancing problem, where one single model is assembled, no parallel workstations are allowed and zoning constraints are not considered. Therefore, we modified these first two methods for applying to MMALBP of type-1. Comparative experiments are carried out to evaluate the performances of the three heuristics, simulated annealing, pure GA, ANTBAL and the proposed hybrid GA on a benchmark data set including 20 MMALBPs of type-1. The proposed hybrid GA showed better performance than pure GA for large sized problems. Although the proposed hybrid GA explored the same performance with ANTBAL, it has an advantage of requiring less computational effort than ANTBAL.

Keywords: Mixed-model assembly line balancing problem, genetic algorithm, heuristic, hybridization, parallel workstation assignment, zoning constraints

PARALEL İSTASYON ATAMALI KARIŞIK TİPLİ MONTAJ HATTI DENGELEME PROBLEMİNİN MELEZ GENETİK ALGORİTMA İLE ÇÖZÜMÜ

ÖZ

Bu tezde, 1.tip karışık tipli montaj hattı dengeleme problemi ele alınmaktadır. Bu problem, maliyet veya kapasite tabanlı bir amaç fonksiyonunu optimize ederken, önceden belirlenen bir çevrim zamanına göre minimum istasyon sayısını bulma problemidir. Çeşitli kesin sonuç veren ve yaklaşım yöntemleri, bu probleme çözüm aramak amacıyla kullanılmışlardır. Problemin NP-Hard yapısından dolayı büyük ölçekli problemlerin çözümünde kesin sonuç veren algoritmalar etkili olamamakta ve arama algoritmaları da büyük ölçekli problemlerde çözüm uzayını etkili bir şekilde arama konusunda yetersiz kalabilmektedirler. Bu dezavantajın üstesinden gelebilmek için son yıllarda meta-sezgisellerin çeşitli algoritmalar ile kombinasyonu birçok çalışma tarafından ele alınmıştır. Bu yaklaşımlar melez meta-sezgiseller olarak adlandırılmaktadırlar.

Bu tez çalışmasının temel amacı 1.tip karışık tipli montaj hattı dengeleme probleminin çözümü için kapasite tabanlı bir amaç fonksiyonunu optimize eden genetik algoritma tabanlı melez bir algoritma sunmaktır. Melez tipi sıralı olarak seçilmiştir. Önerilen metod, paralel istasyon ataması ve zoning kısıtları gibi gerçek karışık tipli montaj hatlarının bazı belirgin özelliklerini ele almaktadır. Melez genetik algoritmanın elde edilmesi için bilinen üç sezgisel algoritma, Kilbridge ve Wester Sezgiseli, Moodie ve Young Metodunun I. Aşaması ve RPWT, kullanılmıştır. İlk iki sezgiselin orjinal versiyonları tek bir ürün tipinin üretildiği basit montaj hattı dengeleme problemlerinin çözümünde paralel istasyon ataması ve zoning kısıtları göz önünde bulundurulmadan kullanılmaktadır. Bu yüzden, Moodie & Young Metodunun I. Aşaması ve Kilbridge ve Wester Sezgiseli karışık tipli montaj hatlarında kullanılabilecek şekilde modifiye edilmiştirr. Sonrasında modifiye edilmiş bu sezgisellerin, genetik algoritmanın, tavlama benzetiminin, ANTBAL'ın ve önerilen melez genetik algoritmanın performansını test etmek için karşılaştırmalı

deneyler 20 adet 1.tip problem kullanılarak yapılmıştır. Önerilen melez genetik algoritma genetik algoritmadan büyük problemlerin çözümünde daha iyi sonuç vermektedir. ANTBAL ile aynı performansı göstermesine rağmen daha az hesaplama gerektirmektedir.

Anahtar Kelimeler: Karışık tipli montaj hattı dengeleme problemi, genetik algoritma, sezgisel, melezleme, paralel istasyon ataması, zoning kısıtları

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

1.1 Importance of the Problem

Assembly lines were first introduced by Henry Ford in 1913. He was the first to introduce a moving belt in a factory. Before the moving belt, workers were able to build one piece of an item at a time instead of an item at a time. This changed type of manufacturing system and reduced the cost of production. Over the years an important problem type, design of efficient assembly lines, recieved much attention. A well-known assembly design problem is *assembly line balancing* problem(ALBP). ALBP is a decision problem of optimally partitioning (balancing) the assembly work among the stations with respect to some objective.

An assembly line is a flow-oriented production system where the productive units performing the operations, referred to as workstations, are aligned in a serial manner. The workpieces(jobs) visit stations successively as they are moved along the line usually by some kind of transportation system, usually by a conveyor belt. The workpieces are consecutively launched down the line and are moved from station to station. At each station, certain operations are repeatedly performed regarding the cycle time (maximum or average time available for each workcycle).

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

The role of assembly lines has been changing through time. Assembly lines were firstly created to produce a low variety of products in high volumes. They allow low production costs, reduced cycle times and accurate quality levels. These are important advantages from which companies can benefit if they want to remain competitive. However, single-model assembly lines, designed to carry out a single homogenous product, are the least suited production system for high variety demand scenarios. The current market is intensively competitive and consumer-centric. For example, in the automobile industry, most of the models have a number of features, and the customer can choose a model based on their desires and financial capability. Different features mean that different additional parts must be added on the basic model. Due to high cost to build and maintain an assembly line, the manufacturers produce one model with different features or several models on a single assembly line. Under these circumstances, the mixed model assembly line balancing problem arises to smooth the production and decrease the cost.

Formally, a mixed model assembly line balancing problem can be stated as follows (Gokcen and Erel,1997):

- ✓ Given *M* models,
- ✓ The set of operations associated with each model,
- ✓ The processing time of each operation (operation time),
- ✓ The set of precedence relations which specify the permissible orderings of the operations for each model.

The problem is to assign the operations to an ordered sequence of workstations such that precedence relations of each model are satisfied and some performance measures are optimized. Unlike the case of a single model line, different models of a product are assembled on a mixed model assembly line. The models are launched to the line one after another. Essentially, this problem is a sequencing problem with constraints: different sequences of operations being processed correspond to different allocation plans.

Nowadays, mixed model assembly lines, wherein different models of a standardized product are produced in an intermixed sequence, are the main focus of the research community. For a mixed model assembly system, finding a line balance whose station loads have the same station time whatever model is produced is almost impossible. This is because, the models differ from each other with respect to size, colour, used material or equipment and consequently their production requires different tasks, task times and/or precedence relations. The problem is more difficult than the single model case because the station times of the different models have to be smoothed for each station in order to avoid operating inefficiencies like work overload or idle time (Becker and Scholl 2006). The allocation of assembly times to workstations in a mixed-model assembly line balancing problem (MMALBP) is characterized by two types of variability or imbalances, vertical and horizontal;

- ✓ Vertical imbalance (model variability) results from the difficulty of reaching a perfect balance for each model separately, due to precedence and technological constraints.

- ✓ Non-identical total assembly time required by the different models on a station due to non-identical times for the same tasks on different models causes horizontal imbalance (station variability).

The two types of variability cause blockage and starvation, and as a consequence, high idle times within stations result in low line efficiency and/or throughput. Equal distribution (to the extent possible) of load on to the workstations considering all the models involved can reduce these types of variability. The classic objective of minimizing cycle time is not necessarily the same objective as load equalization (or smoothing). The aim of the latter usually translates into minimization of the squared differences between workstation loads, which means that a small increase in the maximum lead time may yield a substantial reduction in load imbalance, i.e. a better equalization of workload (Venkatesh and Dabade, 2008).

1.2 Framework of the Thesis

Various exact solution approaches, branch and bound, integer programming, dynamig programming etc., deal with MMALBP. However, due to the NP-Hard structure of the problem none of the optimum seeking methods have proven to be practical to solve large scale problems. Therefore, several heuristics and meta-heuristics (Genetic algorithm, Tabu Search, Simulated Annealing, Ant Colony Optimization, etc.) have been employed to solve the problem effectively.

Among the meta-heuristics, most widely used one is the genetic algorithm. Because, it provides an alternative to traditional optimization techniques by using directed random searches to locate optimum solutions in complex landscapes. It is also proven to be effective in various combinatorial optimization problems. However, as real life problems get larger and more complex, pure genetic algorithms may lack the capability of exploring the solution space effectively. As a remedy, over the last years, a number of studies have been reported combining the various algorithmic ideas of meta-heuristics. These approaches are commonly referred to as *hybrid meta-heuristics*.

The main objective of this study is to propose a hyrid algorithm based on genetic algorithm to tackle the MMALBP with parallel workstations assignment under the zoning constraints, and then to test the performance of the proposed hybrid genetic algorithm on an existing benchmark set of 20 problems.

For hybridization of the genetic algorithm three well known heuristics, Kilbridge and Wester (Kilbridge and Wester, 1961), Phase-I of Moodie and Young Method (Moodie and Young, 1965), and Ranked Positional Weight Tehnique (RPWT) (Helgeson and Birnie, 1961) are used. These approaches only address the simple assembly line balancing problem, where one single model is assembled, no parallel workstations are allowed and zoning constraints did not take into consideration. In order to apply these methods to MMALBP, modified versions are used.

In this thesis, we first solve the MMALBP with parallel workstations by employing the modified versions of three well known problem specific algorithms and pure genetic algorithm. Finally we propose a sequential hybrid genetic algorithm to tackle the problem. First a random population (a set of faesible solutions) is generated and then the solutions obtained by both Kilbridge and Wester Heuristic, Phase-I of Moodie and Young Method and RPWT inserted in the initial population

1.3 Outline of the Thesis

Rest of the thesis involves four chapters. The following chapter contains an overview of the assembly line balancing problem. In this chapter, a literature review about MMALBP which spans 12 years from 1997 through 2009 is also given.

Chapter three gives an overwiev on meta-heuristics, hybrid meta-heuristics, genetic algorithms and application of genetic algorithms for solving assembly line balancing problems

The fourth chapter mainly focuses on solving MMALBP with parallel workstations using the proposed hybrid genetic algorithm, pure genetic algorithm and the other heuristics combined with genetic algorithm. The solution results are presented in detail.

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

CHAPTER TWO
AN OVERWIEW ON ASSEMBLY LINE BALANCING PROBLEM

2.1 Assembly Lines

An *assembly line (AL)* is a manufacturing process consisting of various tasks in which interchangeable parts are added to a product in a sequential manner at a station to produce a finished product. Assembly lines are the most commonly used method 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.

The installation of an assembly line is a long-term decision and usually requires large capital investments. Therefore, it is important that an AL is designed and balanced so that it works as efficiently as possible. Most of the work related to the ALs concentrate on the *assembly line balancing* (ALB). The ALB model deals with the allocation of the tasks among stations so that the precedence relations are not violated and a given objective function is optimized. For a comprehensive review on ALB, see Boysen, Fliedner, and Scholl (2007).

2.1.1 Terminology of Assembly Line Production

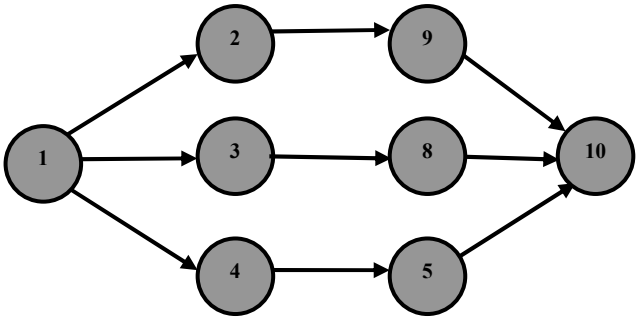
Assembly is the process of collecting and fitting together various parts in order to create a finished product. It is characterized by the used parts and the work necessary to combine them. The relationships of parts and the flow of material can be visualized by assembly charts. The unfinished units of the product are called workpieces.

An operation (task) is a portion of the total work content in an assembly process. The time necessary to perform an operation is called operation (task) time.Operations are indivisible, because they can not be split into smaller work elements without creating unnecessary additional work.

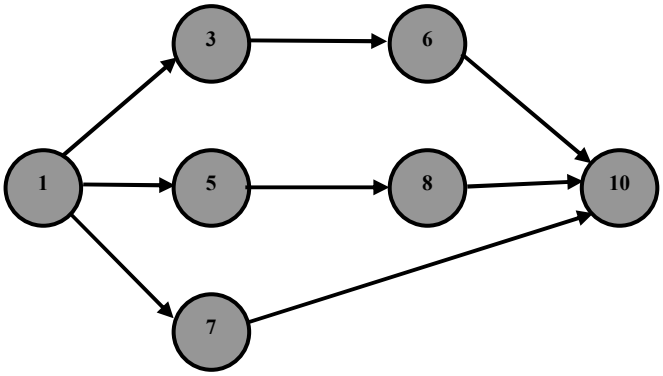
A (work) station is a segment of an assembly line where a certain amount of work (a number of operations) is performed. It is mainly characterized by its dimensions, the machinery and equipment as well as the kind of assigned work. To this effect, stations can be subdivided into manual or automated stations depending on the subjects performing the work.

The cycle time represents the maximal amount of time a workpiece can be processed by a station of a paced assembly line. Since tasks are indivisible work elements, the cycle time can not be smaller than the largest operation time. In unpaced flow-line production systems (including mixed-model lines), the cycle time serves as maximal possible average station time. The time interval during which a workpiece is accessible to a station is called tolerance time. The output rate or production rate of the line equals the reciprocal of the cycle time. A positive difference between the cycle time and the station time is called idle time. The sum of idle times for all stations of the line is called balance delay time.

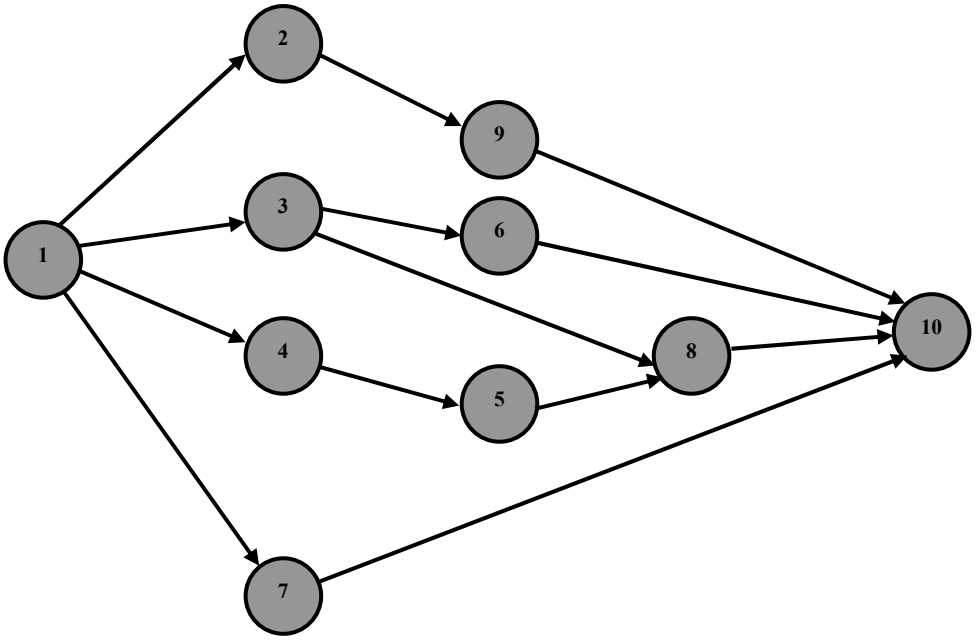
The ordering in which operations must be performed may partially be prespecified. This partial ordering of tasks can be illustrated by means of a ***precedence diagram*** which contains nodes for all operations and arcs (i,j) if an operation i must precede an operation j. In Figure 2.1 it can be seen the precedence diagrams of two models and ***joint precedence diagram*** after merging the precedence diagrams of these two models.



(a)



(b)



(c)

Figure 2.1 Precedence diagrams of (a) model 1, (b) model 2 and (c) combined.

A line balance (feasible task assignment) represents a feasible solution of a balancing problem. A feasible solution is characterized by the following properties: Because of its indivisibility each task is assigned to exactly one station. The precedence constraints are fulfilled, i.e., no task j which must succeed a task i is assigned to an earlier station than i. The station times of all stations of all stations (or the average station times) do not exceed the cycle time.

2.1.2 Additional Features of Assembly Lines

Assembly lines can be distinguished with regard to the number and variety of products assembled in the line (Scholl, 1999) (see Figure 2.2). An assembly line can be treated as ***single-model line***, if only one product or several products with identical production process are assembled. However, in most of the modern manufacturing environments, several products or different models of the same base product often share the same assembly line. If several products are assembled in batches in an AL, it is called a ***multi-model line***. Another type of lines is a ***mixed-model line***, where different models of the same base product are assembled simultaneously in the same line (not in batches).

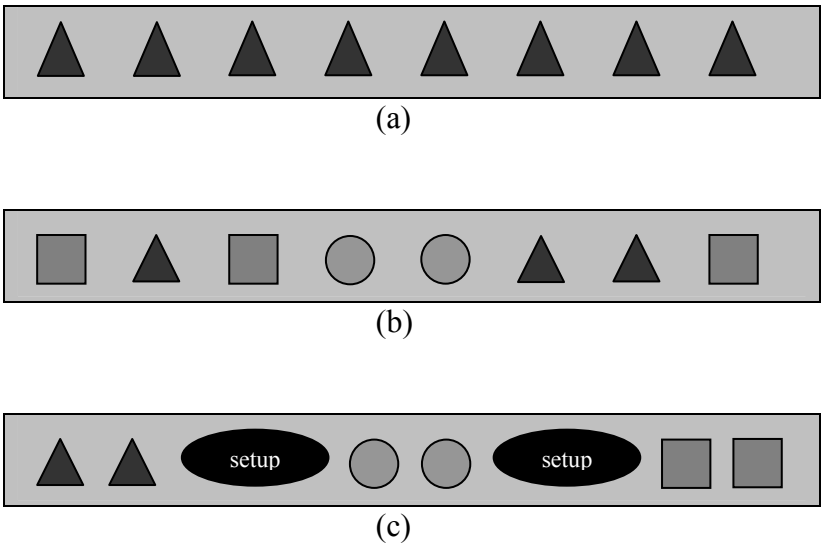


Figure 2.2 Assembly lines: (a) single-model, (b) mixed-model, and (c) multi-model.

Line control is another characteristic of the assembly lines. Assembly lines can be classified in three groups in dependency of line control: paced lines, unpaced asynchronous lines and unpaced synchronous lines. In a ***paced*** assembly production system typically a common cycle time is given which restricts process times at all stations. In ***unpaced*** lines, workpieces are transferred whenever the required operations are completed, rather than being bound to a given time span. Under ***asynchronous*** movement, a workpiece is always moved as soon as all required operations at a station are completed and the successive station is not blocked

anymore by another workpiece. In order to minimize waiting times, buffers are installed in-between stations, which can temporarily store workpieces. Under ***synchronous*** movement of workpieces, all stations wait for the slowest station to finish all operations before workpieces are transferred at the same point in time. In contrast to the asynchronous case, buffers are hence not necessary (Boysen, Fliedner and Scholl, 2008).

Assembly lines can also be distinguished with regard to the nature of task processing times which can be deterministic, stochastic, hidden or dynamic. In the case of manual ALs, the task time is constant only in the case of highly qualified and motivated workers. More advanced machines and robots are able to work permanently at a constant speed. At these cases the task processing times are assumed to be ***deterministic***. Likewise, the task processing times are accepted as ***stochastic***, if the human work rate, skill and motivation result in variations in processing times. In the case of automated stations, it is often difficult to determine the operating time of a complex task (two or more grouped tasks). Indeed, the process time of a station is not always the sum of the operating times of each equipment in the group because of the so-called ***hidden*** times. In the case of human workers, systematic reductions are possible due to the learning effects or successive improvements of the production process, the task processing times are assumed to be ***dynamic***.

In the plant layout problem, emphasis is often put on material flow between departments. Single stations are arranged in a ***straight line*** along a conveying system at the case of serial lines. As a consequence of introducing the JIT production principle, it has been recognised that arranging the stations in a ***U-line*** has several advantages over the traditional configuration. Workers are placed in the centre of the 'U' and can monitor each other's progress and collaborate easily whenever required. With high production rates, the longest task time sometimes exceeds the specified cycle time. A common remedy is to create ***stations with parallel or serial posts***, where two or more workers perform an identical set of tasks. It is common to duplicate the entire AL ***(parallel lines)*** when the demand is high enough. For complex products, the assembly system is most of the time decomposed into sub-

systems ***(workcentres)*** which are easier to manage than the entire system. Another special line layout arrangement is the ***feeder line***, which provides a main line with subassemblies.

Another issue, which must take into consideration, for assembly lines is assignment constraints. Assignment constraints reduce the set of workstations to which tasks can be assigned. This type of constraint can be classified into four groups:

(i) zoning constraints, (ii) workstation related constraints, (iii) position related constraints and (iv) operator related constraints.

Zoning constraints force or forbid the assignment of different tasks to the same workstation, being called positive or negative zoning constraints, respectively. Positive zoning constraints are normally related with the use of common equipment or tooling. Negative zoning constraints are usually imposed by technological issues.

Workstation related constraints are needed if special equipment is only available at a determined workstation. Then the tasks that need that equipment must be assigned to that workstation.

In the case of large and heavy products the workpieces have a fixed position and cannot be turned. So, it may be necessary to perform tasks, for example, at both sides of the line. In this case a 2-sided line is used. It is, therefore, convenient to include ***position related constraints*** that group tasks according to the position in which they are performed.

When tasks require different levels of skills, depending on their complexity, ***operator related constraints*** are needed to ensure that a sufficiently qualified operator is assigned to a determined task. The qualification of an operator is determined by the most complex task assigned to its workstation. For ergonomic reasons, more monotonous tasks and more variable tasks should be combined in the same workstation in order to induce higher levels of job satisfaction and motivation.

2.1.3 Performance Measures of Assembly Lines

The implementation of an assembly line requires high capital investments, hence it is the most important issue that designing and balancing the line in order to produce as efficiently as possible. Also, re-balancing an existing assembly line is required when changes in the production process or demand structure occur. To assess the performance of the line, several criteria of technical and economical nature such as number of workstations, workload variance, idle time and the line efficiency can be included in assembly line balancing problems.

The goals may to minimize the number of workstations, to minimize the workload variance, to minimize the idle time, and to maximize the line efficiency as shown in (2.1)-(2.4), respectively, where n is the number of workstations, $n_{\max}$ is the maximum number of workstation allowance, W is the total processing time, ct is the cycle time, ct_r is the actual cycle time, T_i is the processing time of the i^{th} workstation, L_{eff} is the line efficiency, w_v is the workload variance, and T_{id_T} is the total idle time (Suwannarongsri et al., 2007).

$$\frac{W}{ct} \le min\ n \le n_{max} \tag{2.1}$$

$$min\ T_{id\_T} = min \sum_{t=1}^{n} (ct - T_i) \tag{2.2}$$

$$min\ w_V = min \sum_{t=1}^{n} \left[T_i - \left(\frac{W}{n} \right) \right]^2 /n \tag{2.3}$$

$$max\ L_{eff} = max \frac{\sum_{i=1}^{n} T_i}{(n.ct\_r)} 100 \tag{2.4}$$

The economical nature criteria deals with minimising the total costs of the line, including long-term investment costs and short-term operating costs. Both

installation and operation costs depend mainly on the cycle time and the number of workstations. According to Scholl (1999), the most important cost categories are

- ✓ costs of machinery and tools,
- ✓ labour costs,
- ✓ materials costs,
- ✓ idle time costs,
- ✓ penalties for not meeting the demand,
- ✓ incompletion costs,
- ✓ setup costs and
- ✓ inventory costs.

Recent studies deals more with multi objective approaches, which are consider simultaneously two or more performance measures, than the approaches aim at optimizing only one performance measure. Multi objective approaches provide better line balances when it compared with the methods dealing with the optimization only one performance measure.

Nonetheless, social goals may be important to fulfil, such as

- ✓ job enrichment, avoiding the assignment of many monotonous tasks to an operator and

- ✓ job enlargement, increasing the number of tasks performed by an operator.

2.2 Mixed Model Assembly Line

Mixed-model production systems are mainly used due to the following advantages. They provide a continuous flow of materials, reduce the inventory levels of final items, and are very flexible with respect to model changes. However, this flexibility requires expensive equipment which reduce or even eliminates delays due to set-up activities

2.2.1 Mixed Model Assembly Line Balancing Problem

Several decision problems with different planning horizons arise in managing mixed-model lines. Medium-term (or long-term) decisions concern the installation of the line and the division of work among the stations. This scope includes the determination of the line length (number of stations, lengths of the stations), the production rate or, equivalently, the cycle time as well as the work loads of the stations. Most of these decisions are part of the mixed-model assembly line balancing problem. As in the case of single-model production, it is the problem of finding a number of stations and a cycle time as well as a respective assignment of tasks to the stations such that certain objective is optimized.

Based on the model structure, ALB models can be classified into two groups as seen in Figure 2.3. While, the first group includes single-model assembly line balancing (smALB), multi-model assembly line balancing (muALB), and mixed-model assembly line balancing (mALB); the second group includes simple assembly line balancing (sALB) and general assembly line balancing (gALB). The smALB model involves only one product. The muALB model involves more than one product produced in batches. The mALB refers to assembly lines which are capable of producing a variety of similar product models simultaneously and continuously (not in batches). Additionally, sALB, the simplest version of the ALB model and the special version of the smALB model, involves production of only one product with features such as paced line with fixed cycle time, deterministic independent processing times, no assignment restrictions, serial layout, one sided stations, equally equipped stations and fixed rate launching. The gALB model includes all of the models that are not sALB, such as balancing of mixed-model, parallel, u-shaped and two sided lines with stochastic dependent processing times; thereby more realistic ALB models can be formulated by gALB (Gen, Cheng and Lin, 2008). In this study we deal with one of the model depended assembly line balancing problem, type-I of the mixed-model assembly line balancing problem with parallel workstation assignment under the zoning constraints.

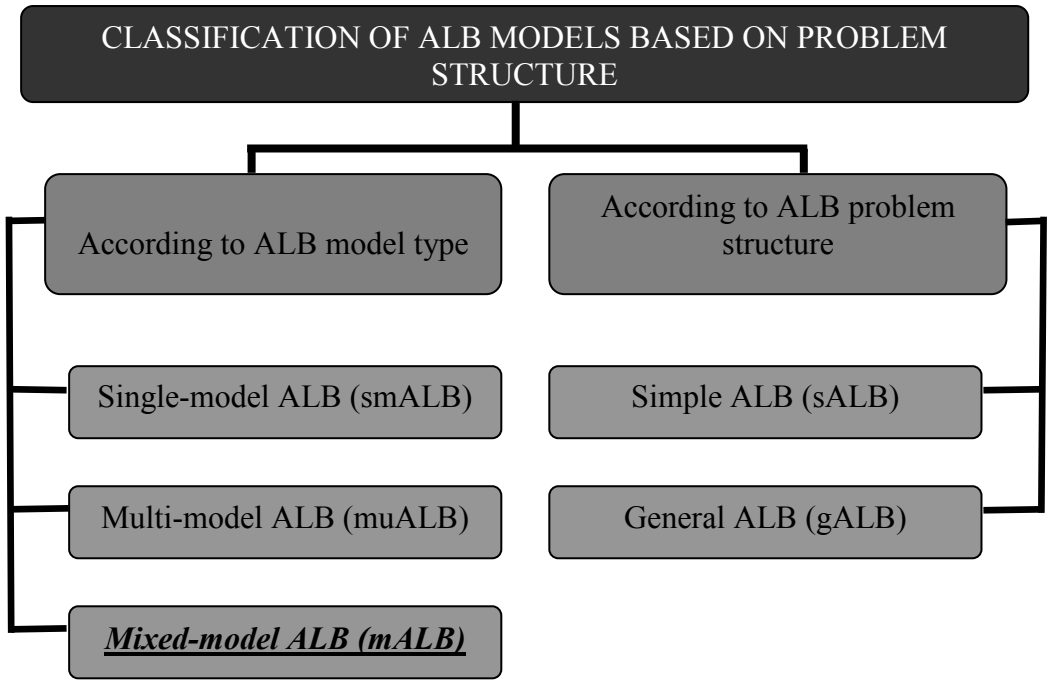


Figure 2.3 Classification of assembly line balancing models

2.2.1.1 Description of the Mixed-Model Assembly Line Balancing Problem

Mixed-model assembly line balancing problem relies on the basic assumptions of deterministic operation times, no assignment restrictions, serial line-layout, fixed rate launching, as single assemby line balancing problem. Unlike the case of a single model line, different models of a product are assembled on a mixed-model assembly line. The models are launched to the line one after another. Table 2.1 contains all the notations for MMALBP. Additional and modified characteristics which results from the joint assembly of several products are:

- ✓ The assembly line is capable of producing more than one type of product simultaneously, not in batches
- ✓ The assembly of each model requires performing a set of tasks which are connected by precedence relations (precedence graph for each model).
- ✓ A subset of tasks is common to all models; the precedence graphs of all models can be combined to a non-cyclical joint precedence graph.

Table 2.1 The notations for mixed-model assembly line balancing problem

M	Number of models, $index: m = 1, ..., M$
d_m	Expected demand for model mduring the planning period
D	Total number of units required during the planning period $(=\sum_m d_m)$
J	Number of operations (tasks), $index: j = 1, ..., J$
PT	Total time available for production during the planning period
c	(average) cycle time, launch interval $(c \le [PT / D])$
t_{jm}	Operation time of task j for one unit of model m
t'_j	Cumulated time of task j for all required units $(=\sum_m d_m t_{jm})$
t_j	Average operation time of task j per unit (t'_j / D)
K	Number of stations, $index: k = 1, ..., K$
S_k	Station load, set of tasks assigned to station k
τ_{mk}	Processing time per unit of model m in station k $(= \sum_{j \in S_k} t_{jm})$
τ'_{mk}	Total operation time of model m in station k $(=d_m \tau_{mk})$
$\bar{t}_m$	Average operation time per station and unit of model m $(=\sum_k \tau_{mk} / K)$
$\bar{t}'_m$	Average total operation time per station for model m $(=d_m \bar{t}_m)$
τ'_k	Total operation time in station k $(=\sum_m \tau'_{mk})$
τ_k	Average operation time of station k per unit $(=\tau'_k / D)$

- ✓ Tasks which are common to several models are performed by the same station but may have different operation times; zero operation times indicate that a task is not required for a model.
- ✓ Fixed total time available for the production during the planning period (given by the number of shifts and the shifts durations) known.
- ✓ Expected demands for all models (expected model mix) during the planning period are known.

2.2.1.2 Mathematical Formulation of MMALBP

The MMALBP can be formulated as a binary integer programming model, as presented in Figure 2.4. The notations belonging to this formulation are:

- ✓ N is the number of tasks of the combined precedence diagram
- ✓ M is thenumber of models assembled on the line
- ✓ D_m is the demandof the model m over the planning horizon P
- ✓ q_m is the overall proportion of the number of units of model m being assembled,
- ✓ given by $D_m / \sum_{p=1}^{M} D_p$
- ✓ S is the number of workstations
- ✓ C is the cycle time computed by $P / \sum_{m=1}^{M} D_m$
- ✓ t_{im} is the processing timeof task i for model m
- ✓ Suc_i is the set of tasks that can not be performed before task i is completed (successors of task i), derived from the combined precedence diagram

- ✓ $x_{ik} = \begin{cases} 1, & \text{if task } i \text{ assigned to workstation } k \\ 0, & \text{otherwise} \end{cases}$

The objective function (1) aims at minimising the weighted idle time of the assembly line, taking into consideration each model's (assembled on the line) production share. This goal is equivalent to minimise the number of workstations for a predefined cycle time in MMALBP-1 (Type-1 problem, see subsection 2.2.1.3) and to minimise the cycle time for a given number of workstations in MMALBP-2 (Type-2 problem).

$$\text{Minimise} \quad \sum_{k=1}^{S}\left(C - \sum_{m=1}^{M} q_m \sum_{i=1}^{N} t_{im} x_{ik}\right) \tag{1}$$

subject to :

$$\sum_{k=1}^{S} x_{ik} = 1 \qquad i = 1,\ldots..,N \tag{2}$$

$$\sum_{k=1}^{S} kx_{ik} - \sum_{k=1}^{S} kx_{jk} \le 0 \qquad i \in N, j \in Suc_i \tag{3}$$

$$\sum_{i=1}^{N} t_{im} x_{ik} \le C \qquad k = 1,\ldots..,S; m = 1,\ldots..,M \tag{4}$$

$$x_{ik} \in \{0,1\} \qquad i = 1,\ldots..,N; k = 1,\ldots..,S \tag{5}$$

Figure 2.4 Binary integer programming model for mixed-model assembly line balancing problem

The set of constraints (2) ensures that each task is assigned to only one workstation of the station interval and consequently tasks that are common to several models are performed on the same workstation.

The precedence constraints are handled by the set of constraints (3) which guarantees that no successor of a task is assigned to an earlier station than that task.

Constraints (4) are called capacity constraints and ensure that the workload of a workstation does not exceed the cycle time, regardless of the model being assembled.

Finally the set of constraints (5) defines the domain of the decision variables.

In this study we deal with mixed-model assembly line balancing problem with parallel workstations under the zoning constraints. This type of MMALBP is formulated by Vilarinho and Simaria (2002). Their mathematical model and its explanation will be given in section four in details.

2.2.1.3 Type-I of MMALBP

MMALBP–1 deals with minimizing the total number of workstations for a given cycle time C. Usually, the cycle time is derived from the total available time, PT, and desired output volume D, i.e., C= [PT/D]. In type I problems, the cycle time, and, consequently the production rate, has to be pre-specified, so it is more frequently used in the design of a new assembly line for which the demand can be easily forecasted.

2.2.1.4 Variations of Station Utilization

The objectives included in the types of MMALBP are based on considering the balancing problem with respect to average station utulizations. Even in the case of an optimal solution for the average model, considerable inefficiencies may occur when operating the line. This is due to the variations in the station times of the models. Next it is discussed that the influences of work load variations on the performance of mixed-model assembly lines.

Work overload occurs, whenever the operator of a station is not able to complete the assigned operations on a workpiece. It is measured in terms of the remaining operation time to complete the operations. Work overload is inefficient and expensive and should be minimized. Unfortunately, the amount of work overload

which really occurs can not be computed for a solution of the balancing problem directly, because it depends on the unknown short-term production programs and corresponding production sequences. Therefore, it is essential to obtain balances in which potential work overload situations are minimized.

The interval of time during which a workpiece is accessible to a station is called ***tolerance time***. It is an upper bound on the time available for performing operations in that station. The tolerance time should be smaller than the cycle time in order to avoid unnecessary idle times. In the case of a continuously moving conveyor belt, the tolerance times are determined by the physical station lengths and the speed of the belt.

It is illustrated the influence of tolerance times by assuming that they are equal to the cycle time in all stations. All station times of any model which are in excess of the cycle time result in incomplete operations and a corresponding work overload. If the tolerance times exceed the cycle time, the sequence in which model units are launched down the line influences the amount of work overload.

Idle time occurs when a station has completed its work on a unit and has to wait for the next unit arriving at the station. The idle times per unit are constant if only one model is produced. If several models are assembled, the idle times differ and depend on the sequence. Only for strcitly paced lines, they are independent of the sequence.

Work overload can only occur if some station times of models exceed the cycle time. In order to quantify such ***cycle time violations***, it is defined:

$$\triangle^{+}_{mk} := \max\{0, \tau_{mk} - c\} \quad \text{for } k = 1, \ldots\ldots, K \text{ and } k = 1, \ldots\ldots, K \tag{2.5}$$

According to the origin of cycle time violations, it is distunguished three types:

Operaiton-dependent Cycle Time Violations: A violation of cycle time can not be avoided by balancing if the operation time t_{jm} of a single task j of model m exceeds the cycle time.

Model-dependent Cycle Time Violations: If the average operation time $\bar{t}_m$ per unit of model m exceeds the cycle time, cycle time violations can not be prevented in one or more stations unless the cycle time c or the number K of stations is increased.

Assignment-dependent Cycle Time Violations: Cycle time violations which are neither operation-dependent nor model-dependent are caused by assignment of tasks to stations. Hence, these violations may be influenced by balancing decisions even for fixed c and K.

Station times which are smaller than the cycle time may cause idle times. These potential idle times are called ***slack times*** and are defined by:

$$\bar{\triangle}_{mk} := \max\{0,\ c - \tau_{mk}\} \text{ for } k = 1, \ldots\ldots, K \text{ and } k = 1, \ldots\ldots, K \tag{2.6}$$

Slack times may have different reasons like cycle time violations:

Model-dependent Slack Times: In case the average operation time $\bar{t}_m$ per unit of model m is smaller than the cycle time, slack times can not be avoided in one or more stations unless the cycle time c or the number K of stations is decreased if possible.

Assignment-dependent Slack Times: Slack times which are not model-dependent are caused by the assignment of the tasks to the stations and, therefore, they may be influenced by balancing.

While slack times tend to produce idle times, they give possibilities to equalize cycle time violations of other models. Therefore, it is not always the best decision to choose a task assignment which minimizes slack times.

2.2.2 Solution Approaches for Mixed-Model Assembly Line Balancing Problem

Several approaches have been presented to assembly line design and mixed–model assembly problem. Work allocation, line balancing, and job sequencing algorithms deal mainly with the mixed-model assembly line design problem. This study deals with the papers published in recent years that address the mixed-model assembly line balancing problem (MMALBP) in different layout configurations, developing exact or heuristic approaches (Batini, Faccio, Ferrari, Persona, and Sqarbossa, 2007).

Most of the authors have proposed methods of reducing the multiple models into a single one by combining their precedence relationships and adjusting the operation time. The majority of them (>50%) address the balancing problem to traditional serial assembly systems and, in some cases, they allow the use of identical parallel workstations at each stage of the serial system (Kara, Özcan, and Peker, 2007).

The single-model line-balancing problem (SALBP) is of NP-Hard complexity (Karp, 1972). Since the single-model line-balancing problem is a special case of the mixed-model line-balancing problem discussed here (where the number of models equals one), the latter is NP-hard as well (Bukchin, and Rabinowitch, 2006). The complex mathematical nature of the problem makes it difficult to solve (Erel and Gokcen, 1999). As a result, beside the exact solution procedures, heuristic approaches are developed which are gives optimal or near optimal solution at a resonable time.

In general, the combinatorial optimization problems are characterized by a finite number of feasible solutions. Especially for small sized practical problems, the optimal solution of such problems can be found by enumeration. Therefore, in

literature, it is observed that there exists a tendency to use heuristics rather than exact methods. The complexity of the assembly line balancing problem renders optimum seeking methods impractical for instances of more than a few tasks and/or workstations. If there are m tasks and r preference constraints then there are m!/2r possible task sequences (Baybars, 1986). Therefore, it can be time consuming for optimum seeking methods to obtain an optimal solution within this vast search space. Despite the vast search space, many attempts have been made in the literature to solve the ALBP using optimum seeking methods. However, none of these methods has proven to be of practical use for large problems due to their computational inefficiency. Hence, numerous research efforts have been directed towards the development of approximation methods. Figure 2.5 contains the classification of the solution methods used to solve ALBP.

2.2.2.1 Exact Methods

Several approaches for determining lower bounds on the number of stations (n) in the case of ALBP-1 (the cycle time in the case of ALBP-2) 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, which are dynamic programming and branch and bound methods.

2.2.2.1.1 Dynamic Programming. The dynamic programming (DP) method is applied to the most combinatorial optimization problems (COP) and involves the optimisation of multi-stage decision procedures. A given problem is divided into sub-problems which are sequentially solved until the initial problem is finally solved. States at a particular stage s are transformed to states at the subsequent stage s + 1 by a decision. The generation of states is described by transformation functions which depend on the current state and the decision taken. A sequence of decisions, which transforms a state at a stage s to a stage $s' > s$, is called policy. DP is a solving approach rather than a technique.

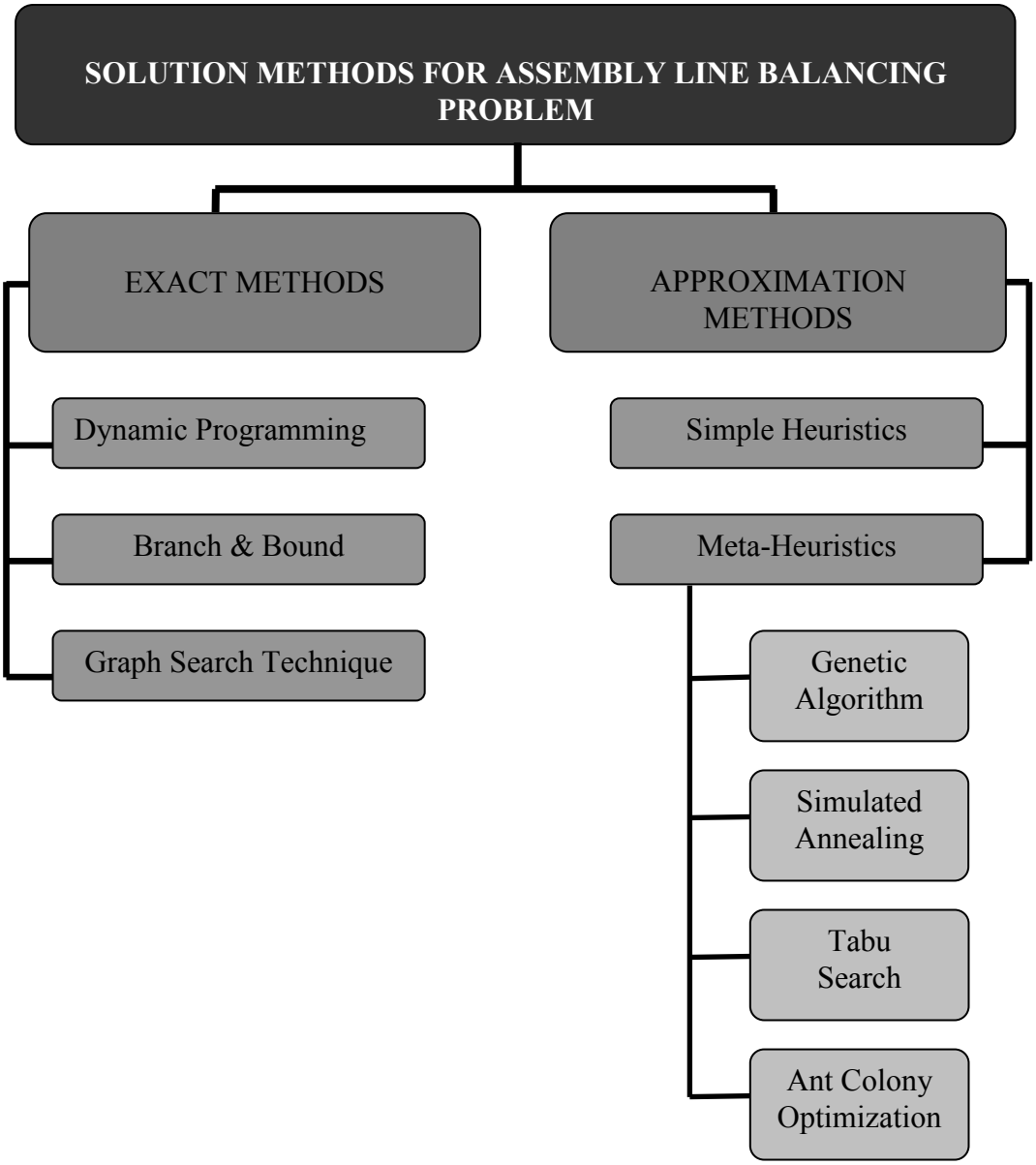


Figure 2.5 Classification of solution approaches for ALBP (Rekiek and Delchambre, 2006)

2.2.2.1.2 Branch and Bound. The branch and bound (B&B) algorithm consists of two main components: the branching and the bounding. The initial solution is developed into several sub-problems (branching). A multi-level enumeration is constructed by continuously developing such sub-problems for which the optimal solution is already known and need not be branched. These sub-problems are referred to as leaf nodes. A path from the root node to any other node of the tree is called a branch. 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. An optimal solution is found if the ‘global’ lower bound is found.

2.2.2.2 Approximation Methods

Near optimal or optimal solutions can be determined by approximation methods are more preferable and acceptable in practice because they can be obtained more efficiently. These approaches are divided into two categories, simple heuristics and meta-heuristics.

2.2.2.2.1 Simple Heuristics. Heuristics approaches are based on logic and common sense rather than on an mathematical proof. None of the methods guarantees an optimal solution, but they are likely to result in good solutions which approach the true optimum. Among simple heuristic methods, the most notable ones are: Ranked Positional Weight Technique (RPWT) (Helgeson and Birnie, 1961), Kilbridge and Wester’s (1961), and Moodie and Young's (1965) heuristics. RPWT is the first heuristic proposed to solve ALBP.

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

Batini et al. (2007) give a callasification of the published papers between the years 1989 and 2005 in relation to the adopted balancing method and the reference layout configuration taken in consideration. This classification shows us that many autors used mathematical programming models or heuristic procedures in order to solve the mixed-model assembly line balancing problem. The lack of hybrid approaches in the

literature for solving mixed-model assembly line balancing problem can also be seen from this classification.

2.3 Literature Review

According to the variety of the assembled product types, assembly lines are normally classified into single-model, multi-model and mixed-model lines (Scholl, 1999). While only one single homogeneous product is manufactured in large quantities on single-model assembly lines, different product types are simultaneously produced on mixed- and multi-model assembly lines.

Single-model approaches are frequently defined as specific versions of the restrictive Simple Assembly Line Balancing Problem (SALBP). Here, the balancing problem is reduced to the allocation of tasks to stations. Extensions of the SALBP cover for instance, the integration of parallel stations, the examination of cost-oriented objective functions, processing alternatives and their respective consequences as well as targeted job enrichment (Becker and Scholl 2006).

The Mixed-Model Assembly Line Balancing Problem (MMALBP) can be regarded as the direct counterpart of the SALBP family for mixed-model assembly lines (Bock, 2006). By introducing an aggregated model of all offered variants, three corresponding types of the MMALBP arise (Scholl 1999; Becker and Scholl 2006). Owing to the fact that the restricted MMALBP family does not provide any decision support for finding a balanced line layout, various extensions with specifically defined objective functions are proposed in the literature (Bock, 2006).

It can be find literature on MMALBP way back to the 1960s. Thomopoulos (1970) was the first to develop a heuristic on Mixed-Modeled Assembly line. He focused on general practices in mixed-model assembly line balancing like, to assign work to stations in a manner such that each station has an equal amount of work on a daily or a shift basis.

As to the scope of this study, the literature review encompasses a group of papers published subsequent to 1997 that address the mixed-model assembly line balancing problem in different layout configurations, developing exact or heuristic methods, by chronological order.

Askin and Zhou (1997) proposed a nonlinear integer program as a model for the production line balancing problem (PLBP). This problem entails the assignment of tasks to stages in a serial production line. The model allows mixed-model production and the use of identical parallel workstations at each stage of the serial production system. The objective function trades of idle workstation time with duplication of task-dependent equipment/tooling cost. A heuristic is developed to create parallel workstations and assign tasks. Station utilization is also explicitly considered by using a threshold variable for target (acceptable) levels. Testing has shown the heuristic to respond well to the economic implications of equipment/tooling cost and idle worker time. The heuristic is capable of finding good solutions quickly to large problems.

McMullen and Frazier (1997) described an approach for solving a mixed-model assembly line-balancing problem with stochastic task times when paralleling of tasks within work centers is permitted. The research modified previous work and incorporates new and existing task selection rules for assigning tasks to work centers. The heuristic is applied to six different line-balancing problems for each presented rule. The resulting layouts are simulated and performance results are analyzed.

Gökçen and Erel (1997) proposed a binary goal programming model for the mixed-model ALB problem. The proposed model provides a considerable amount of flexibility to the decision maker since several goals of which some may be conflicting with each other, can simultaneously be considered.

Gökçen and Erel (1998) developed a binary integer programming model for the mixed-model version of the problem in which they utilize some properties that prevent the fast increase in the number of variables. However, their model suggests a

significant improvement relative to the models in the literature. Their model's objective function is to minimize the number of stations utilized. The experimentation revealed that their model is capable of solving problems with up to 40 tasks in the combined precedence diagram. Due to the NP-hardness of the problem, their model size would be too large to obtain the optimal solutions of larger problems.

A shortest-route formulation of the mixed-model assembly line balancing problem is presented by Erel and Gökçen (1999). Common tasks across models are assumed to exist and these tasks are performed in the same stations. Their formulation is based on an algorithm which solves the single-model version of the problem. The mixed-model system is transformed into a single-model system with a combined precedence diagram. Their model is capable of considering any constraint that can be expressed as a function of task assignments. The performance mesaure of their model is the sum of the idle times associated with each model.

Merengo, Nava and Pozzettı (1999) present new balancing and production sequencing methodologies which pursue the following common goals: (1) minimizing the rate of incomplete jobs (in paced lines and in moving lines) or the probability of blocking/ starvation events (in unpaced lines); (2) reducing WIP. The balancing methodology also aims at minimizing the number of stations on the line; the sequencing technique also provides a uniform part's usage, which is a typical goal in just in time production systems. Moreover, they developed a heuristic for balancing problem and tested in four different versions.

A new method using a coevolutionary algorithm, search algorithm that imitate the biological coevolution that is a series of reciprocal changes in two or more interacting species, proposed by Kim and Kim (2000). This algorithm can solve the balancing and sequencing problems at the same time. In the algorithm, it is important to promote population diversity and search efficiency. They adopted a localized interaction within and between populations, and developed methods of selecting symbiotic partners and evaluating fitness. Efficient genetic representations and

operator schemes are also provided. When designing the schemes, they take into account the features specific to the problems. Also presented are the experimental results that demonstrate the proposed algorithm is superior to existing approaches.

Matanachai and Yano (2001) proposed a new line balancing approach for mixed-model assembly lines. Their focus is on assigning tasks to stations so that: (i) workloads are reasonably well balanced; and (ii) it is relatively easy to construct daily sequences of jobs that provide stable workloads (in a minute-to-minute sense) on the assembly line. They proposed a heuristic filtered beam search algorithm in which feasible subsets are constructed at each station. This heuristic is shown to perform well, both in an absolute sense (on small problems), and relative to heuristic solutions for the traditional objective (on larger problems). Because the performance of the heuristic depends on the number of different feasible subsets considered, it can be improved, if desired, by increasing the number of subsets retained for each station.

The goal chasing method is simple and easy to implement, but it is a very greedy algorithm and uses up ‘good’ parts in the early sequence so that the whole performance of the solution is influenced (Jin and Wu, 2002). They provided the definition of good parts and good remaining sequence and analyze their relationship with the optimal solution’s objective function value. Jin and Wu (2002) developed a new heuristic algorithm called ‘variance algorithm’ the numerical experiments show that the new algorithm can yield better solution with little more computation overhead. The objective of the problem is to minimize the variation in rate of consuming the parts of the sequence. They discussed several improvement methods for correcting the myopic problem of the goal chaisng method in JIT. They developed their variance improvement for goal chasing method by integrating the variance as the opportunity cost in the cost function and prove its effectiveness and efficiency by theory and numerical experiments. Besides mixed-model assembly line problem, this improvement can be used everywhere goal chasing method can be used and correct the myopic problem very well.

Vilarinho and Simaria (2002) presented a new mathematical programming model for the mixed-model assembly line balancing problem with parallel workstations and zoning constraints. The model minimizes the number of workstations and allows the user to control the replication process. As a secondary goal, the model looks to obtain a good workload balance between and within the workstations. Due to the model complexity a two-stage heuristic procedure was developed to tackle the problem, which uses the simulated annealing algorithm. Computational experiments showed that the proposed heuristic performs very well, producing good quality solutions in reasonable running times.

The design problem of mixed-model assembly lines in a make-to-order environment is adressed by Buckhin, Dar-El and Rubinovitz (2002). A mathematical formulation of the problem is presented and a heuristic, which takes into consideration the relaxation of the assigment constraint that, provides performing some specific tasks at different stations for different models. The heuristic minimizes the number of stations for a predetermined cycle time. It consists of three stages: the balancing of the combined precedence diagram, balancing each model separately subject to the constrained tasks (resulting from the preciding stage), and an imrpvemenet procedure based on neighborhood search which uses the appropriate performance measure in order to compare solutions. The cycle time of each final solution is then recieved from simulation, and compared to the required cycle time.

Liu and Chen (2002) proposed a two-stage approach. In the first stage, a multiple objective mixed-integer zero-one programming model is developed. Combined with the developed mathematical programming model, an interactive procedure is devised to simultaneously minimize workstation cycle time and number of workstations while satisfying the required total operation cost. In the second stage, a visual interactive modelling system and the associated human-machine interface are built. Results suggest that the potential benefit of the proposed approach is significant.

Karabatı and Sayın (2003) considered the assembly line balancing problem in a mixed-model line which is operated under a cyclic sequencing approach. They

specifically study the problem in an assembly line environment with synchronous transfer of parts between the stations. They formulate the assembly line balancing problem with the objective of minimizing total cycle time by incorporating the cyclic sequencing information. They showed that the solution of a mathematical model that combines multiple models into a single one by adding up operation times constitutes a lower bound for this formulation. As an approximate solution to the original problem, they proposed an alternative formulation that suggests minimizing the maximum subcycle time. They also developed a simple heuristic approach for this alternative problem. Their computational results indicate that this approach may be beter in finding good solutions, however at a higher computational cost.

An approach is presented by McMullen and Tarasewich (2003), based on ant techniques, to effectively address the assembly line balancing problem with the complicating factors of parallel workstations, stochastic task durations, and mixed-models. A methodology was inspired by the behavior of social insects in an attempt to distribute tasks among workers so that strategic performance measures are optimized. This methodology is used to address several assembly line balancing problems from the literature. The assembly line layouts obtained from these solutions are used for simulated production runs so that output performance measures (such as cycle time performance) are obtained. Output performance measures resulting from this approach are compared to output performance measures obtained from several other heuristics, such as simulated annealing. A comparison shows that the ant approach is competitive with the other heuristic methods in terms of these performance measures.

Zhao, Ohno and Lau (2004) stated a balancing problem for mixed model assembly lines with a paced moving conveyor as: Given the daily assembling sequence of the models, the tasks of each model, the precedence relations among the tasks, and the operations parameters of the assembly line, assign the tasks of the models to the workstations so as to minimize the total overload time. They presented a heuristic procedure for balancing a mixed model assembly line. The heuristic attempts to optimize directly the operational performance criterion UT (T) (total

overload time); its computational requirement increases linearly with the number of stations, and can therefore handle large-scale problems. They also presented a procedure for estimating how much the total overload time of any given line balance (T) deviates from the optimally attainable total overload time without actually knowing the optimal line balancing. This provides a powerful and practical approach for assessing the quality of balances for realistic-size lines whose theoretical optimal solutions are typically unobtainable.

Simaria and Vilarinho (2004) presented a mathematical programming model for MMALBP-II which accounts for the use of parallel workstations, in a controlled way, and zoning constraints. Besides the goal of minimising the cycle time, the model also balances the workloads within the workstations for the different models to be assembled. Due to the combinatorial nature of the model, an efficient iterative genetic algorithm-based procedure was developed to tackle the problem.

Vilarinho and Simaria (2006) presented ANTBAL, an ant colony optimization algorithm for balancing mixed-model assembly lines. The proposed algorithm accounts for zoning constraints and parallel workstations and aims to minimize the number of operators in the assembly line for a given cycle time. In addition to this goal, ANTBAL looks for solutions that smooth the workload among workstations, which is an important aspect to account for in balancing mixed-model assembly lines. Computational experience showed the superior performance of the ANTBAL algorithm.

Hop (2006) solved the fuzzy mixed-model assembly line balancing problem with an improvement heuristic. Due to the difficulty of fuzzy comparison and fuzzy arithmetic operations, a simple signed distance ranking method is used to rank fuzzy numbers and new approximated fuzzy arithmetic operations are developed to calculate fuzzy numbers. The problem is then formulated as a mix-integer programming model. This one could be use as a benchmark for a reasonable size of problem. Finally, a heuristic method is developed using a flexible exchange sequence procedure to allocate jobs into workstations. Experiment results show that the

developed algorithm could give very good results in terms of balancing efficient coefficient and number of workstations.

Bock (2006) proposed the use of specifically designed distributed search methods in order to solve complex balancing problems more efficiently. Specifically, Bock (2006) introduces a new mixed-model assembly line balancing approach that makes use of specifically designed distributed solution procedures. The underlying model formulation of this approach distinguishes itself from former concepts by a modular variant definition, a detailed personel planning, and an integrated task process planning. Since the obtained results are very promising, future research should be extended to two major directions. First of all, detailed analyses of the implemented CTS-procedure suggest that specific extensions of the proposed solution approaches are reasonable. For instance, it becomes obvious that keeping a fixed clustering throughout the searching process seems to be not reasonable. In this connection, one can think of an automated team size adaptation within the Clustered Tabu Search approach in order to intensify the search in specific regions of the solution space. For this purpose, teams may be dynamically combined or separated throughout the search.

A common assumption in the literature on mixed-model assembly line balancing is that a task that is common to multiple models must be assigned to a single station (Buckhin and Rabinowitch, 2006). Buckhin and Rabinowtich (2006) relaxed this restriction, and allow a common task to be assigned to different stations for different models. They searched to minimize the sum of costs of the stations and the task duplication. They developed an optimal solution procedure based on a backtracking branch-and-bound algorithm and evaluate its performance via a large set of experiments. A branch-and-bound based heuristic is then developed for solving large-scale problems. The heuristic solutions are compared with a lower bound and experiments show that the heuristic provides much better solutions than those obtained by traditional approaches.

A new genetic approach, called endosymbiotic evolutionary algorithm, is proposed to solve the two problems of line balancing and model sequencing at the same time by Kim et al. (2006). The algorithm imitates the natural evolution process of endosymbionts that is an extension of existing cooperative or symbiotic evolutionary algorithm. The distinguishing feature of the proposed algorithm is that it maintains endosymbionts that are a combination of an individual and its symbiotic partner. The existence of endosymbionts can accelerate the speed that individuals converge to good solutions. This enhanced capability of exploitation together with the parallel search capability of traditional symbiotic algorithms results in finding better quality solutions than existing hierarchical approaches and symbiotic algorithms.

Haq, Jayaprakash, and Rengarajan (2006) presented a hybrid genetic algorithm approach that used the solution from the modified ranked positional method (MRPW) for the initial solution to reduce the search space within the global space, thereby reducing search time to solve the mixed-model assembly line balancing problem. They compared the pure genetic algorithm, MRPW and hybrid genetic algorithm. The genetic algorithm approach is shown to produce beter results than the MRPW in the minimization of workstations. Experimantal results show that the hybrid genetic approach is superior to the classical genetic algorithm.

Batini et al. (2007) deals with the application of a mixed-model assembly balancing problem to an assembly-to-order environment in the case of low production rates and large number of tasks. The aim of their work is to propose an alternative design procedure for the balancing of semi-automated and mixed-model assembly systems under low product demand effects by the application of multi-turn circular transfers, such as a multi-stations rotating table. This layout configuration permits a job enlargement for human operators and, at the same time, provides an increment in task repeatability through the work-pieces assembling by increasing the number of the turns of the transfer.

Kara, Ozcan and Peker (2007) proposed an approach for simultaneously solving the balancing and sequencing problems of mixed-model U-lines. The primary goal of the proposed approach is to minimize the number of workstations required on the line (Type I). To meet this aim, the proposed approach uses such a methodology that enables the minimization of the absolute deviation of workloads among workstations as well. In terms of minimizing the number of workstations required on the mixed-model U-line, as well as minimizing the absolute deviation of workloads among workstations, the proposed approach is the first method in the literature dealing with the balancing and sequencing problems of mixed-model U-lines. The newly developed neighborhood generation method employed in the simulated annealing (SA) method is another significant feature of the proposed approach.

Venkatesh and Dabade (2008) reported an experimental study conducted for a relatively less researched area of assembly line balancing namely the mixed model assembly line balancing problem of type II (MMALBP-2). Three operational objectives namely realized cycle time, model variability and station variability were identified as important for a better evaluation of the solutions. Two performance measures namely squared base model deviation (SBMD) and squared base model deviation plus smoothness index (BMI) have been proposed in their paper. The two performance measures along with eight others (reported in literature earlier) have been used as fitness function for a GA (developed for this work) to obtain solutions of 3000 MMALBP-2 instances generated randomly for the experimental setup.

Bock (2008) presented a new tabu search based approach that provides detailed offshoring decision support for mass customization manufacturing processes is generated. It is focused on specific manufacturing processes where theoretically a large number of variants have to be simultaneously produced on the same mixed-model assembly line. In order to provide an appropriate estimation of the resulting manufacturing costs, the layout of the mixed-model assembly line was respectively generated for competing locations. This approach is the first one that provides a detailed analysis of the tradeoff between lower worker wages and additional manufacturing costs caused by reduced worker skills.

A design methodology for team-oriented mixed model assembly lines is proposed by Cevikcan, Durmusoglu and Unal (2009). The discussed method includes some relevant issues that reflect the operating conditions of real world mixed model assembly lines, such as zoning constraints and workload smoothing and also allows the decision maker to control the size of multi-manned workstations. There exist intensive input–output interrelations among methodology steps. The first two steps of the methodology, horizontal and vertical balancing, are necessary for performing the third step, which is to create physical stations. Teams consist of a minimum number of workers whose workloads are smoothed via the first two steps of the methodology. The third step uses a scheduling algorithm under limited resources during forming of the teams.

Simaria and Vilarinho (2009) presented an approach to deal with the two-sided mixed-model assembly line balancing problem. First, a mathematical programming model is presented to formally describe the problem. Then, an ant colony optimisation algorithm is proposed to solve the problem. In the proposed procedure two ants 'work' simultaneously, one at each side of the line, to build a balancing solution which verifies the precedence, zoning, capacity, side and synchronism constraints of the assembly process. The main goal is to minimise the number of workstations of the line, but additional goals are also envisaged

Choi (2009) presented a new mathematical model of line balancing for processing time and physical workload at the same time. He proposed a zero-one integer program model that combines the overload of processing time and physical workload with various risk elements. For the solution techniques, he adopted the goal programming approach and designed an appropriate algorithm process. Various computational test runs are performed on the processing time only model, the physical workload only model, and the integrated model. Comparing the pay-offs between the two overloads, test results show that well balanced job allocation is able to be obtained through the proposed model.

A new mathematical model and a simulated annealing algorithm for the mixed model two-sided assembly line balancing problem are presented by Ozcan and Toklu (2009). The proposed mathematical model minimizes the number of mated-stations as the primary objective and minimizes the number of stations as a secondary objective for a given cycle time. In the proposed simulated annealing algorithm, two performance criteria are considered simultaneously: maximizing the weighted line efficiency and minimizing the weighted smoothness index. The proposed approach is illustrated with an example problem, and its performance is tested on a set of test problems. The experimental results show that the proposed approach performs well.

The literature review we provided is summarized in Table 2.2. This table contains a group of published papers, that adress the MMALBP between the years 1997 and 2009 in chronological order.

Our conclusions about this review are listed below.

- ✓ Many papers address the balancing problem to traditional serial assembly systems and, in some cases, they allow the use of identical parallel workstations at each stage of the serial system.

- ✓ Most of methods are based on reducing the multiple models in to a single one by combining their precedence relationships and adjusting the operation time.

- ✓ Mathematical models proposed for MMALBP are not suitable for large scale problems.

- ✓ To deal with large scale problems, approximation methods have been proposed.

- ✓ There is a lack of hybrid algorithms for MMALBP.

Table 2.2 Evolution of proposed solution approaches for MMALBP

PUBLICATIONS	LINE CONFIGURATION	METHODOLOGY
Askin and Zhou (1997)	straight line, parallel stations	nonlinear integer programming+heuristic
McMullen and Frazier (1997)	straight line, parallel stations	heuristic, simulation
Gökçen and Erel (1997)	straight line	binary goal programming
Gökçen and Erel (1998)	straight line	binary integer programming
Erel and Gökçen (1999)	straight line	network programming
Merengo et al. (1999)	paced and unpaced lines	heuristic
Kim and Kim (2000)	straight line	coevolutionary algorithm
Matanachai and Yano (2001)	closed station, paced line	mathematical model, filtered beam search
Jin and Wu (2002)	straight line	mathematical model, heuristic
Vilarinho and Simaria (2002)	straight line, parallel stations	mathematical model, simulated annealing
Buckhin et al. (2002)	straight line	mathematical model, heuristic
Liu and Chen (2002)	straight line	mixed-integer zero-one programming, heuristic, simulation
Karabatı and Sayın (2003)	straight line	integer programming, heuristic
McMullen and Tarasewich (2003)	straight line, parallel stations	ant colony optimization, simulation
Zhao, Ohno and Lau (2004)	paced line	heuristic
Simaria and Vilarinho (2004)	straight line, parallel stations	mathematical model, genetic algorithm
Vilarinho and Simaria (2006)	straight line	ant colony optimization
Hop (2006)	straight line	fuzzy binary linear programming, heuristic
Bock (2006)	straight line	distributed search procedures
Buckhin and Rabinowitch (2006)	straight line	branch and bound algorithm based heuristic
Kim et al. (2006)	U-line	endosymbiotic evolutionary algorithm (genetic algorithm)
Haq et al. (2006)	straight line	hybrid genetic algorithm
Batini et al. (2007)	multi stations	heuristic
Kara, Ozcan and Peker (2007)	U-line	simulated annealing
Venkatesh and Dabade (2008)	straight line	genetic algorithm
Bock (2008)	straight line	tabu search
Cevikcan et al. (2009)	straight line	heuristic
Simaria and Vilarinho (2009)	two-sided line	ant colony optimization
Choi (2009)	straight line	goal programming
Ozcan and Toklu (2009)	two-sided line	mathematical model, simulated annealing

CHAPTER THREE
AN OVERWIEW ON META-HEURISTICS, HYBRIDIZATION AND GENETIC ALGORITHMS

3.1 Meta-Heuristics

In the last 20 years, a new kind of approximate algorithm has emerged which basically tries to combine basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space. These methods are commonly called ***metaheuristics*** (Blum and Roli, 2003). The term *metaheuristic*, first introduced in Glover [1986], derives from the composition of two Greek words. *Heuristic* derives from the verb *heuriskein* which means "to find", while the suffix *meta* means "beyond, in an upper level".

This type of algorithms includes Ant Colony Optimization (ACO), Evolutionary Computation (EC) including Genetic Algorithms (GA), Iterated Local Search (ILS), Simulated Annealing (SA), and Tabu Search. For a basic chronology of well-known meta-heuristics see Figure 3.1. There is no commonly accepted definition for the term of metaheuristic. It is just in the last few years that some researchers in the field tried to propose a definition. For the definitions proposed to explain the term meta-heuristic the reader can refer to Blum and Roli (2003).

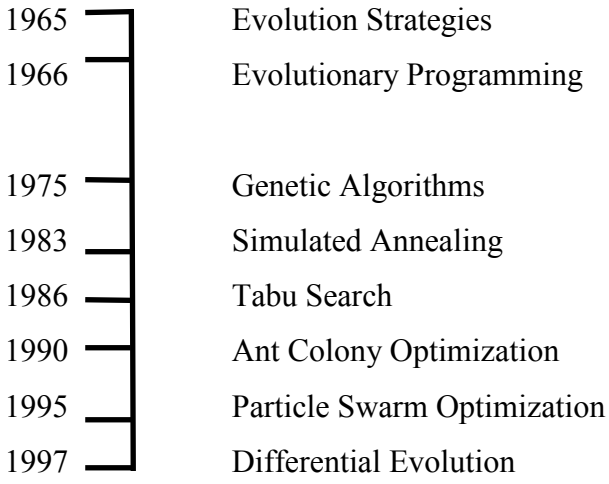


Figure 3.1 Chronology of meta-heuristics

3.1.1 Properties of Meta-Heuristics

It can be said that meta-heuristics are high level approaches for discovering the search space of the problem on hand by using different methods. For a powerful algorithm there must be a dynamic balance between ***diversification*** and ***intensification***. The term diversification generally refers to the exploration of the search space, whereas the term intensification refers to the exploitation of the accumulated search experience (Blum and Roli, 2003), it is important to clarify that the terms ***exploration*** and ***exploitation*** are sometimes used instead. In fact, the notions of exploitation and exploration often refer to rather short-term strategies tied to randomness, whereas intensification and diversification also refer to medium- and long-term strategies based on the usage of memory (Blum and Roli, 2003).

The main properties given by Blum and Roli (2003) in order to characterize meta-heuristics:

- ✓ Metaheuristics are strategies that guide the search process.
- ✓ The goal is to efficiently explore the search space in order to find (near-) optimal solutions.
- ✓ Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
- ✓ Metaheuristic algorithms are approximate and usually non-deterministic.
- ✓ They may incorporate mechanisms to avoid getting trapped in confined areas of the search space.
- ✓ The basic concepts of metaheuristics permit an abstract level description.
- ✓ Metaheuristics are not problem-specific.
- ✓ Metaheuristics may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper level strategy.
- ✓ Todays more advanced metaheuristics use search experience (embodied in some form of memory) to guide the search.

3.1.2 Classification of Meta-Heuristics

There are different ways to classify and describe metaheuristic algorithms. Depending on the characteristics selected to differentiate among them, several classifications are possible, each of them being the result of a specific viewpoint. According to Blum and Roli (2003) the most important ways of classifying metaheuristics are summarized as follows.

Nature-inspired or non-nature-inspired: This classification of meta-heuristics is based on the origins of the algorithm. Genetic Algorithms and Ant Algorithms are in the class of nature-inspired while Tabu Search and Iterated Local Serach are in the class of non-nature-inspired. It may be sometimes difficult to clearly attribute an algorithm to nature-inspired or non-nature-inspired.

Population- based or single-point search: Depending on the number of solutions used at the same time meta-heuristics can be classified as population-based or single-point searchs. Single-point searchs are also called as trajectory methods. Tabu Search and Simulated Annealing are best known single-point search techniques. On the contrary Genetic Algorithms and Ant Colony Optimization are population-based approaches.

Dynamic or static objective function: This type of classification depend on the way meta-heuristics make use of the objective function. While some algorithms keep the objective function given in the problem representation “as it is”, some others, like Guided Local Search (GLS), modify it during the search in order to escape from local minima by modifying the search landscape.

One or various neighborhood structures: Most meta-heuristic algorithms work on one single neighborhood structure. It means that, the fitness landscape topology does not change in the course of the algorithm. Other meta-heuristics, such as Variable Neighborhood Search (VNS), use a set of neighborhood structures which gives the possibility to diversify the search by swapping between different fitness landscapes.

Memory usage or memory-less methods: Another important chracteristic in order to classify meta-heuristics is the use they make of the search history, that is, whether they use memory or not. Memory-less algorithms perform a Markov process, as the information they exclusively use to determine the next action is the current state of the search process. There are several different ways of making use of memory. Usually it is differentiated between the use of short term and long term memory. The first usually keeps track of recently performed moves, visited solutions or, in general, decisions taken. The second is usually an accumulation of synthetic parameters about the search. The use of memory is nowadays recognized as one of the fundamental elements of a powerful metaheuristic.

3.2 Hybrid Algorithms

Search techniques have been widely used to solve many combinatorial optimization problems (COPs) of both theoretical and practical importance. The available approaches for COPs can be classified into two main categories: exact (complete) and approximate (heuristic) algorithms. Exact methods such as branch and bound or dynamic programming guarantee to find an optimal solution for a finite sized problem in bounded time. However, as the size of the problem gets larger, the time needed by the complete algorithms may increase exponentially. On the other hand, approximation methods are able to find a good (optimal or near optimal) solution in less amount of time.

Another approach that was taken to solve optimization problems was the introduction of hybrid algorithms. A hybrid algorithm is a combination of complete or approximate algorithms (or both) used to solve the problem in hand (El-Abd and Kamel, 2005). The interest among researchers in this field has risen in the past years since many of the best results obtained for many combinatorial optimization problems where found by hybrid algorithms.

Especially over the last years a large number of algorithms were reported that do not purely follow the concepts of one single traditional metaheuristic, but they

combine various algorithmic ideas, sometimes also from outside of the traditional metaheuristics field. These approaches are commonly referred to as *hybrid metaheuristics* (Raidl, 2006).

Hybridization of heuristics involves a few major issues which may be classified as design and implementation. The former category concerns the hybrid algorithm itself, involving issues such as functionality and architecture of the algorithm. The implementation consideration includes the hardware platform, programming model and environment on which the algorithm is to run (Talbi, 2002).

The motivation behind hybridization of meta-heuristics concepts is usually to obtain better performing systems that exploit and unite advantages of the individual pure strategies, i.e. such hybrids are believed to benefit from synergy. The vastly increasing number of reported applications of hybrid metaheuristics specifies the popularity, success, and importance of this specific line of research. In fact, today it seems that choosing an adequate hybrid approach is determinant for achieving top performance in solving most difficult problems (Raidl, 2006).

3.2.1 Classification of Hybrid Meta-Heuristics

There is a lack of studies in the literature providing a global view of the art of hybridization. Also there is a shortage of guidelines to help in the choice of how to hybridize and which algorithms should be combined to obtain the best result on a certain instance of the problem on hand. However, Preux and Talbi (1999), Talbi (2002) and Raidl (2006) aimed at classifying the concept of hybridization of metaheuristics from different points of view. Among these three publications the more comprehensive one for classification of hybrid meta-heuristics is given by Raidl (2006). Figure 3.2 illustrates the various classes and properties by which Raidl (2006) want to categorize hybrids of meta-heuristics. This classification depends on the methods used (what is hybridized?), level of hybridization, order of execution and control strategy. For the details of this classification the reader may refer to Raidl (2006).

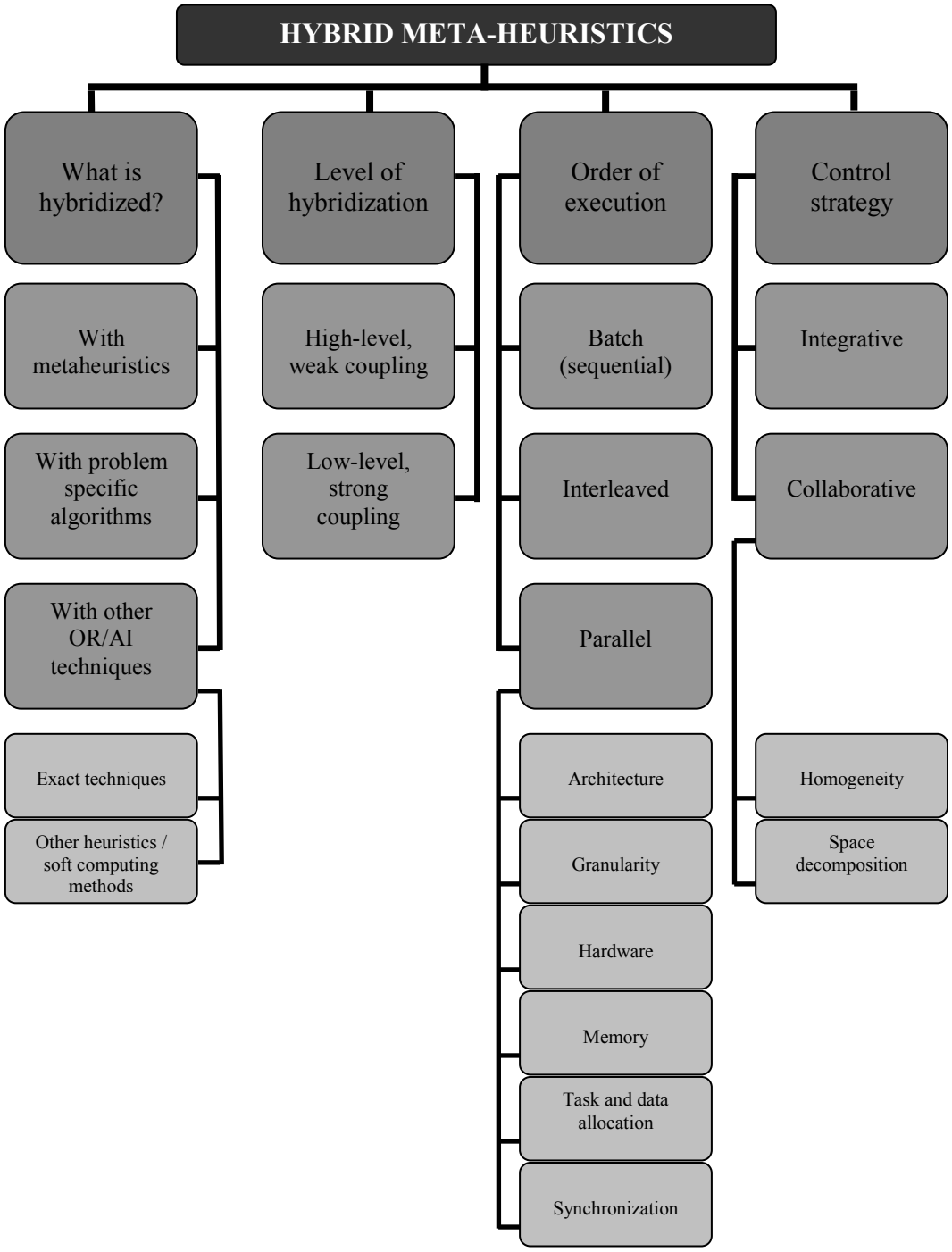


Figure 3.2 Classification of hybrid meta-heuristics

The first class of the hybrid meta-heuristics is based on the methods used to combine. Meta-heuristics can be combined with (i) different meta-heuristic strategies, (ii) certain algorithms specifically developed for the considered problem,

(iii) other more general techniques coming from the fields of operations research (OR) and artificial intelligence (AI) (Raidl, 2006).

Level of hybridization is the second classification of hybrid meta-heuristics. This class may be distinguished between low-level and high-level hybridizations. The low-level hybridization addresses the functional composition of a single optimization method. In this hybrid class, a given function of a metaheuristic is replaced by another metaheuristic (Talbi, 2002). On the contrary, high-level combinations in principle retain the individual identities of the original algorithms and cooperate over a relatively well defined interface; there is no direct, strong relationship of the internal workings of the algorithms (Raidl, 2006).

.

The order of execution is another feature by which the third class of hyrid meta-heuristics may be distinguished as sequential, interleaved and parallel. In the sequential model, one algorithm is strictly performed after the other, and information is passed only in one direction. An intelligent preprocessing of input data or a postprocessing of the results from another algorithm would fall into this category. On the contrary, the interleaved and parallel models, in which the algorithms might interact in more sophisticated ways (Raidl, 2006). For more detailed information of hybrid parallel meta-heuristics the reader can refer to El-Abd and Kamel (2005).

The fourth and final class of hybrid meta-heuristics is based on control strategy, integrative and collaborative combinations. In integrative approaches, one algorithm is considered a subordinate, embedded component of another algorithm. This approach is extremely popular. In collaborative combinations, algorithms exchange information, but are not part of each other (Raidl, 2006).

In this study we deal with the class of sequential hybrid meta-heuristics. We aim at solving mixed-model assembly line balancing problem with this type of hybridization. Some authors have used the technique of sequential hybridization, i.e., Haq, Jayaprakash, and Rengarajan (2006). Figure 3.3 contains some types of sequential hybridization of meta-heuristics.

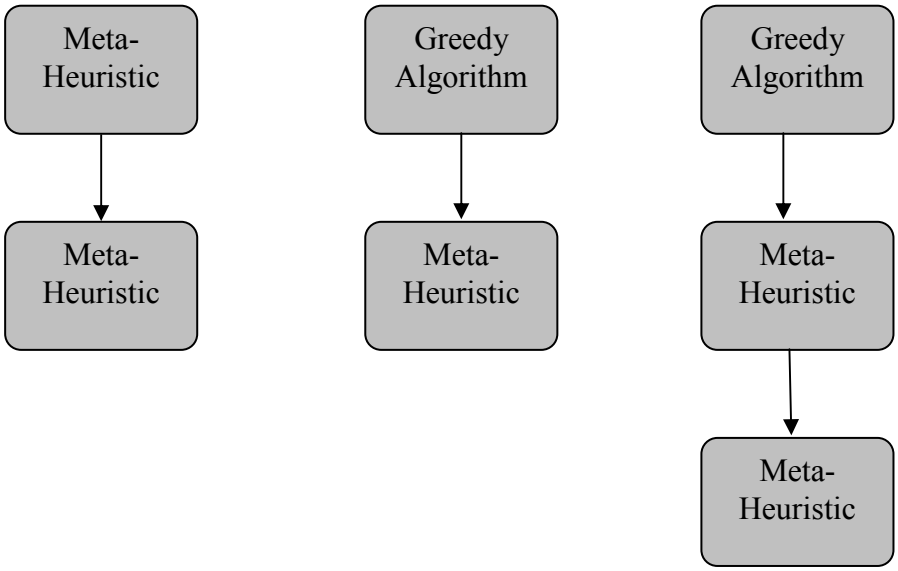


Figure 3.3 Some types of sequential hybrid approaches

According to Preux and Talbi (1999) a fundamental and practical remark is that after a certain amount of time, the population is quite uniform and the fitness of the population is no longer decreasing, the odds to produce fitter individuals being very low. That is, the process has fallen into a basin of attraction from which it has a (very) low probability to escape.

This point leads to raise two issues:

- ✓ once fallen in a basin, the algorithm is not able to know if it has found the optimal point, or if it has fallen into a local optimum. Hence, we need to find ways to escape the optimum in order to try to find another optimum,

- ✓ the exploitation of the already found basin of attraction has to be realized in order to find, as efficiently as possible, the optimal point in the basin.

3.3 Genetic Algorithms

Genetic algorithms (GAs) are powerful and broadly applicable stochastic search and optimization approaches, which simulate the natural behaviour of biological systems. Holland (1975) introduced the developed fundemantel ideas of genetic algorithms and genetic algorithms were popularized as a solution method by one of Holland's students, David Goldberg, who was able to solve a difficult problem involving the control of gas-pipeline transmission (Goldberg, 1989). GAs have been succesfully adapted to solve several combinatorial optimization problems (COPs) in the literature and have become increasingly popular among approximation techniques for finding optimal or near optimal solutions in a resonable time to COPs. This popularity of GAs depend on the below listed advantages, which are intriguing and produce stunning results when traditional optimization approaches fail miserably (Haupt and Haupt, 2004).

- ✓ Optimize with continuous or discrete variables,
- ✓ Do not require derivative information,
- ✓ Simultaneously searchs from a wide sampling of the cost surface,
- ✓ Deal with a large number of variables,
- ✓ Are well suited for parallel computers,
- ✓ Optimize variables with extremely complex cost surfaces (they can jump out of a local minimum),
- ✓ Provide a list of optimum variables, not just a single solution,
- ✓ May encode the variables so that the optimization is done with the encoded variables, and
- ✓ Work with numerically generated data, experimental data, or analytical functions.

Goldberg (1989) introduced the differences of genetic algorithms from traditional optimization techniques in for ways:

- ✓ GAs work with a coding of the parameter set, not the parameters themselves.
- ✓ GAs search from a population of points, not a single point.
- ✓ GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge.
- ✓ GAs use probabilistic transition rules, not deterministic rules.

Besides these differences of GAs, a genetic algorithm must have five basic components, as summarized by Michalewicz (Gen and Cheng, 2000):

- ✓ A genetic representation of solutions to the problem.
- ✓ A way to create an initial population of solutions.
- ✓ An evaluation function rating solutions in terms of their fitness.
- ✓ Genetic oprerators that alter the genetic composition of children during reproduction
- ✓ Values for the parameters of genetic algorithms.

Genetic algorithms maintain a population of individuals, each of them represents a feasible solution to the problem at hand. Each individual is evaluated to give measure of it fitness. In order to create new individuals some individuals from the population undergo stochastis tranformations by means of genetic operations. There are two ways creating new individuals, mutation and crossover called genetic operators. Mutation creates new individuals by making changes (mutating) in an single individual while crossover creates new individuals by combining parts (mating) from two individuals. New individuals, called offspring, are then evaluated. A new population is formed by selecting individuals from the parent population and the

offspring population according to a selection procedure. After several generations (a predefined iteration number), the algorithm converges the most fit individual, which represents an optimal or suboptimal solution to the problem at hand. Figure 3.4 illustrates main steps of a generalized genetic algorithm.

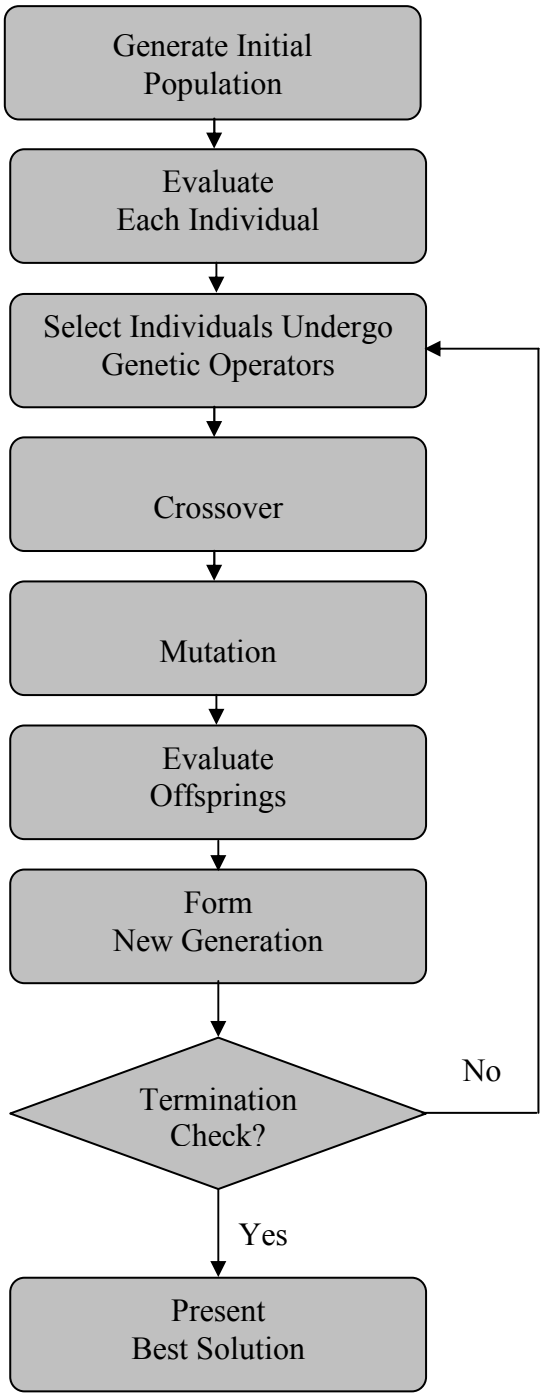


Figure 3.4 Main steps of a generalized genetic algorithm

3.3.1 Terminology of Genetic Algorithms

In order to understand the philosophy of genetic algorithms seven basic terms relating to GAs must be defined. These basic components include coding of solutions (chromosomes or individuals), population, fitness function, selection, genetic operators (mutation and crossover), forming new generation (survival scheme) and termination criteria.

In GA terminology, coding of a feasible solution called an *individual* or a *chromosome*. Chromosomes are made of discrete units called *genes,* each of them controls one or more features of the chromosome. Genes are assumed to be binary digits in the original implementation of GA by Holland (see Figure 3.5-a), however more varied chromosome types have been introduced in later implementations (see Figure 3.5-b). A chromosome corresponds to a feasible solution in the solution space. This requires a mapping mechanism, called *encoding*, between the solution space and the chromosomes. In fact, GAs work on the encoding of a problem, not on the problem itself.

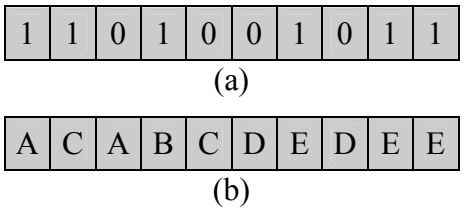


Figure 3.5 Chromosome represantations

GAs operate with a group of solutions (chromosomes), called a *population*. *Initial population* is generated often randomly. For problems with small feasible regions, initialization can incorporate problem-specific knowledge to increase the likelihood of having feasible individuals and to generate some good solutions in the initial population. Another alternative in population initialization is to locate approximate solutions by using other methods and to start the algorithm from such points (Coley, 2003).

In order to evaluate and rate the performance of a chromosome GAs use problem specific *fitness functions,* which are the monotonic functions of the problem's objective function at hand, usually.

Selection involves choosing individuals that will be exposed to genetic operations. The parents are selected among existing chromosomes in the population with preference towards fitness so that offspring is expected to inherit good genes which make the parents fitter. The selection procedure has a significant influence on driving the search towards a promissing area and finding good solutions in a short time (Pham and Karaboga, 2000). Common types of selection procedure are roulette whell selection, $(\mu+\lambda)$-selection, tournament selection, steady-state reproduction, ranking and scaling and sharing. Roulette whell selection, proposed by Holland (1975), is the best known selection type (Gen and Cheng, 2000). Coley (2003) summarised the fitness-proportional selection (roulette whell) in below:

1. Sum the fitness of all the population members. Call this f_{sum}.

2. Choose a random number, R_s, between 0 and f_{sum}.

3. Add together the fitness of the population members (one at a time) stopping immediately when the sum is greater R_s. The last individual added is the selected individual and copy is passed to the next generation.

The selection mechanism is applied twice (from step 2) in order to select a pair of individuals to undergo, or not to undergo, crossover. Selection continued until N (the population size, assumed to be even) individuals have been selected (Coley, 2003).

Crossover and *mutation* are the operators, used by GAs, in order to generate new individuals from selected chromosomes in the population. *Crossover* is the main operator of GAs. The crossover procedure: generally two chromosomes, called parents, are combined together to form new chromosomes, called offspring. A two-point crossover exchanges all genes between the cut-points, which are randomly

determined in general (see Figure 3.6). The aim of crossover is to transmit good characteristics from parents to offspring.

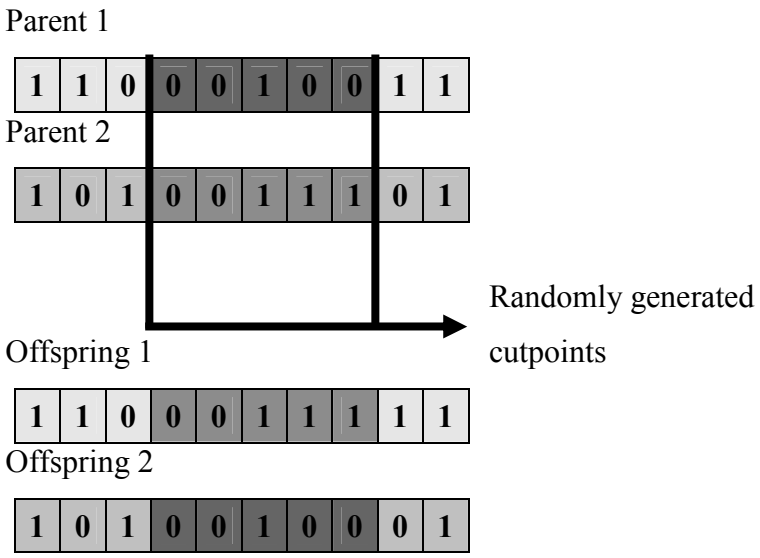


Figure 3.6 Two point crossover

Mutation is a background operator, which is used to maintain genetic diversity from one generation of a population of chromosomes to the next. Figure 3.7 shows the simplest mutation, which is performed by changing the value of a randomly selected gene from 0 to 1 (or from 1 to 0) in a binary string. In GAs, mutation serves the crucial role of either (a) replacing the genes lost from the population during the selection process so that they can be tried in a new context or (b) providing the genes that were not present in the initial population (Gen and Cheng, 1997).

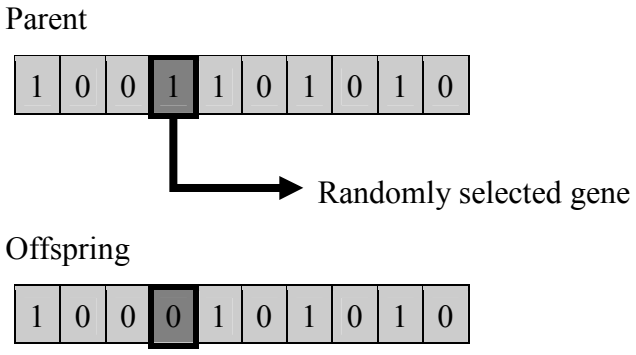


Figure 3.7 Mutation

A *replacement strategy* is required in order to form new generation. This strategy determines which chromosomes stay in population and which are replaced by offsprings, generated by crossover or mutation. The individuals of the new generation may be (i) individuals from the current generation, (ii) offspring product of crossover or (iii) individuals who underwent mutation. The most commonly used relacement strategy is elitism, which makes survive some number of the best individuals at each generation, hence guaranteeing that the final population contains the best solution ever found. There are several approaches for the way the offspring replace their parents. Some favour the maintenance of the parents in the population while others always replace the parents by the offspring, even if they are worse than the parents. In either case, a random component is always present to avoid premature convergence to local optima.

3.3.2 Determining GA Parameters

Another important decision to make in implementing a genetic algorithm is how to set the values for the various parameters, such as the population size, crossover rate, mutation rate and termination criteria. The values of these parameters greatly determine whether the algorithm will find a near-optimum solution and whether it will find such a solution efficiently. Choosing the right parameter values, however, is a time-consuming task and considerable effort has gone into developing good heuristics for it. Eiben et al. (1999) classifies parameter setting efforts in two major groups: *parameter tuning* and *parameter control* (see Figure 3.8).

Parameter tuning means that the commonly practiced approach that amounts to finding good values for the parameters before the run of the algorithm and then running the algorithm using these values, which remain fixed during the run. Parameter tuning is a common practice in evolutionary computation. Typically one parameter is tuned at a time, which may cause some suboptimal choices, since parameters often interact in a complex way. Simultaneous tuning of more parameters, however, leads to anenormous amount of experiments. The technical

drawbacks to parameter tuning based on experimentation can be summarized as follows.

- ✓ Parameters are not independent, but trying all different combinations systematically is practically impossible.

- ✓ The process of parameter tuning is time consuming, even if parameters are optimized one by one, regardless to their interactions.

- ✓ For a given problem the selected parameter values are not necessarily optimal, even if the effort made for setting them was significant.

Parameter control forms an alternative, as it amounts to starting a run with initial parameter values which are changed during the run. Controlling the parameters during a run is an alternative to tuning parameters before running the algorithm. It is intuitively obvious that different values of parameters might be optimal at different stages of the evolutionary process. Methods for changing the value of a parameter can be classified into three categories.

- ✓ *Deterministic Parameter Control:* This takes place when the value of a strategy parameter is altered by some deterministic rule. This rule modifies the strategy parameter deterministically without using any feedback from the search. Usually, a time-varying schedule is used, i.e., the rule will be used when a set number of generations have elapsed since the last time the rule was activated.

- ✓ *Adaptive Parameter Control:* This takes place when there is some form of feedback from the search that is used to determine the direction and/or magnitude of the change to the strategy parameter. The assignment of the value of the strategy parameter may involve credit assignment, and the action of the Evolutionary Algorithm (EA) may determine whether or not the new value persists or propagates throughout the population.

- ✓ *Self-Adaptive Parameter Control:* The idea of the evolution of evolution can be used to implement the self-adaptation of parameters. Here the parameters to be adapted are encoded into the chromosomes and undergo mutation and recombination. The better values of these encoded parameters lead to better individuals, which in turn are more likely to survive and produce offspring and hence propagate these better parameter values.

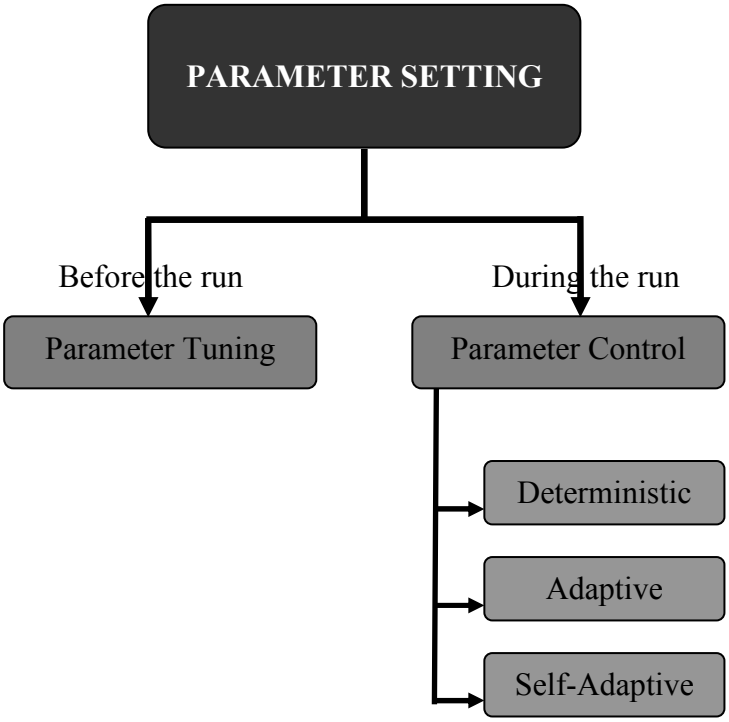


Figure 3.8 Taxonomy for parameter setting in GAs

3.3.3 GAs for Assembly Line Balancing

The main and the most important feature of the application of a GA to an optimization problem, i.e. line balancing problem, is the development of good encoding schemes and genetic operators in order to attain feasible solutions. Another difficulty found in the application of GA to the ALBP is related with the fitness function (Scholl and Becker, 2006). In this section, firstly, it is mentioned the classification of chromosomes schemes given by Gen, Cheng and Lin (2008) then the genetic operators used by implementing GAs to the ALBP. Second, the use of some fitness functions for ALBP in the literature is given.

3.3.3.1 Chromosomes Representation Schemes

The major characteristic in applying GA to problem is to convert a feasible solution of ALB model into a genetic representation form, called chromosome. Up to now, several genetic representations, *i.e.*, task-based, embryonic, workstation-based, grouping-based, and heuristic-based have been proposed (see Figure 3.9).

- ✓ *Task-based Encoding:* The chromosomes are defined as feasible precedence sequences of tasks. The length of the chromosome is defined by the number of tasks. In order to calculate the fitness of a task based chromosome, additional operations, which assign the tasks to workstations according to the task sequence in the chromosome, are needed. Task-based representation is the most appropriate representation for ALB Type-1 models, since Type-1 models consider the minimization of stations as an objective function.

- ✓ *Embryonic Encoding:* Embryonic chromosome representation is actually a special version of the task based chromosome. The only difference between the two is that the embryonic representation of a solution considers the subsets of solutions rather than the individual solutions. During the generations, the embryonic chromosome evolves through a full length solution. Therefore, the chromosome length varies throughout the generations. The length is initially defined by a random number and then increases until it reaches the number of tasks.

- ✓ *Workstation-based Encoding:* The chromosome is defined as a vector containing the labels of the stations to which the tasks are assigned. The chromosome length is defined by the number of tasks. This kind of chromosome representation scheme is generally used for ALB Type-2 models.

- ✓ *Grouping-based Encoding:* In grouping-based representation, proposed especially for grouping problems, i.e. ALB Type-1 models, the stations are represented by augmenting the workstation-based chromosome with a group part. The group part of the chromosome is written after a semicolon to list all of the workstations in the current solution. The length of the chromosome varies from solution to solution. The first part is the same as in workstation-based chromosome. The difference comes from the grouping part, which list all the stations.

- ✓ *Heuristic-based (Indirect) Encoding:* This type of representation scheme represents the solutions in an indirect manner. In this type, it is first coded the priority values of the tasks (or a sequence of priority rules), then applied these rules to the problem to generate the solutions. The chromosome length is defined by the number of heuristics.

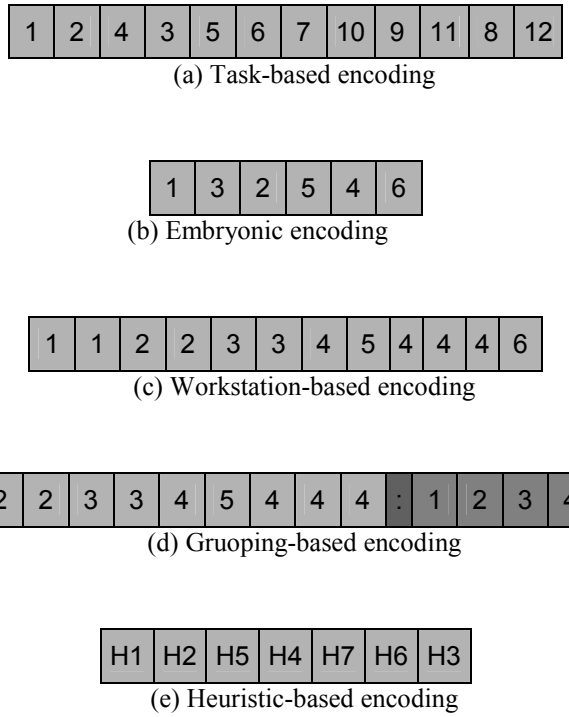


Figure 3.9 Chromosome representation schemes

3.3.3.2 Genetic Operators

Applying standard genetic operators (mutation and crossover) may lead to highly unfeasible solutions due to the precedence constraints of the tasks involved. To tackle this problem, Anderson and Ferris (1994) included in the objective function a penalty cost related with the number of precedence violations of each particular solution.

Another way to address this issue is to force feasibility by using specific genetic operators and applying adaptation procedures to properly build the solutions. The crossover operator proposed by Kim et al (1998) starts by selecting a crossover point p, which corresponds to a workstation. Then, the genes representing workstations 1 to p, in the first parent, are copied to the same position in the first offspring. The remaining positions are copied from those of p+1 to the last workstation in the second parent. Usually, in the resulting offspring there are tasks with no workstation assigned, hence, a reassignment procedure is performed in order to ensure feasibility. The reassignment procedure aims to reassign the remaining tasks to workstations with available capacity, in such a way that the feasibility of the resulting solution is ensured.

The mutation operator proposed by the same authors consists in selecting at random a number of genes and applying the reassignment procedure. Anderson and Ferris (1994) implement mutation by changing a task's workstation (with a small probability) to either the workstation immediately before or immediately after, even so incurring the risk of unfeasibility.

The *two-point order crossover* is typically used for the recombination of chromosomes with order encoding (Leu et al, 1994, Sabuoncuoglu et al, 2000, Khoo and Alisantoso, 2003). Two crossover points are randomly selected, dividing the chromosomes in three parts. The first offspring is a direct copy of the first and last parts of the first parent. The middle part is obtained by rearranging the missing tasks

in the order by which they appear in the second parent. This ensures the feasibility of the resulting task sequence.

Levitin et al (2006) used a crossover operator called *fragment reordering crossover*. This type of crossover works as follows: first, all elements of the first parent are copied to the same positions of the offspring, then, the elements of a random fragment of the offspring are rearranged according to their order in the second parent. This operator seems equivalent to the two-point order crossover.

Mutation operators perform mainly by (i) changing the position of two tasks in the chromosome (Levitin et al, 2006) or (ii) scrambling the genes of the chromosome after a randomly selected point (Leu et al, 1994 and Sabuncuoglu et al, 2000).

3.3.3.3 Fitness Function

Fitness function is used for the performance evaluation of each feasible solution (each chromosome) of the problem at hand. A chromosome is evaluated based on the objective function value of the solution it represents. This value is assigned as the chromosome's fitness. Several fitness functions have been proposed in the literature for the ALBP. For problems of Type-1, there often exist a large number of alternative feasible solutions with the same number of workstations, so it is necessary to use objective functions beyond the minimisation of the number of workstations, for a beter guidance of the search process.

Falkenauer and Delchambre (1992), and Brown and Sumichrast (2005) used the following fitness function:

$$Fitness\text{-}function = \sum_{i=1}^{N} \frac{(F_i/C)^2}{N} \tag{3.1}$$

N is the number of stations, F_i is the i^{th} station time and C is the cycle time.

The *workload balance* is a common goal that ensures equity in the distribution of work among operators. Several expressions to compute workload balance are found in the literature. Leu et al. (1994) minimise the sum of mean squared workstation idle times, computed by the following fitness function:

$$Fitness\text{-}function = \sum_{i=1}^{N} \frac{(t_i - C)^2}{N} \tag{3.2}$$

t_i is the time at the i^{th} station, C is the cycle time, and N is the number of stations.

Sabuncuoglu et al.(2000) used a fitness function that consists of two objectives. The first term aims to balance the workloads between workstations while the second minimises the number of workstations. This fitness function is computed as follows:

$$Fitness\text{-}function = 2\sqrt{\frac{\sum_{k=1}^{n}(S_{max} - S_k)^2}{n}} + \frac{\sum_{k=1}^{n}(S_{max} - S_k)}{n} \tag{3.3}$$

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

Bautista et al. (2000) used the following fitness function with three terms. First term considers the number of workstations, the lower bound of number workstations are dealt in the second term and the degree of imbalance in the third term.

$$Fitness\text{-}function = -NE_j + \left\lceil \frac{\sum_{i=1}^{N} t_i}{C} \right\rceil + \frac{\sqrt{\sum_{k=1}^{NE_j}(c_k - C)^2}}{C\sqrt{NE_j}} \tag{3.4}$$

t_i is the duration of task i, C is the cycle time, NE_j is the number of stations in the j^{th} solution and c_k occupied time in the kth station ($1 \leq k \leq NE_j$).

Goncalves and De Almeida (2002) used the following fitness function which consist of maximizing the line effeciency. It is defined as:

$$Fitness\text{-}function = \frac{t_{sum}}{m.C} 100\% \tag{3.5}$$

C is the cycle time, m is the number of stations and t_{sum} is the total operation time of operations.

Su and Lu (2007) used a fitnees function which aims at smoothing the workload balance within each workstation for mixed-model assembly line balancing problem. It is computed as follows:

$Fitness - function = 1 - \lambda$, where $\lambda = J_1 / U$, constant $U > J_1$ and

$$J_1 = \sqrt{\frac{\sum_{k=1}^{S}\left(\sum_{m=1}^{M} d_m T_{mk} - \frac{\sum_{j=1}^{S}\sum_{m=1}^{M} d_m T_{mj}}{S}\right)^2}{S}} \tag{3.6}$$

M is the number of models, S is the number of workstations, d_m ist he demand for model m in a minimum part set and T_{mj} (T_{mk}) is total processing time for model *m* on workstation *j(k)*,

CHAPTER FOUR
PROPOSED HYBRID GENETIC ALGORITHM FOR SOLVING MMALBP WITH PARALLEL STATIONS

4.1 Problem Definition

This study deals with MMALBP with parallel workstations under the zoning constraints. This type of MMALBP is formulated by Vilarinho and Simaria (2002). Their mathematical programming model minimizes the number of workstations and balances the workloads between workstations and within each workstation for a required cycle time. In this subsection it will be explained this mathematical model by giving its assumptions, constraints and objective function in details, then the complete model will be given. We propose a solution approach to tackle this type of MMALBP according to the scope of this study. The proposed solution approach will be explained and the performance of the approach will be tested on a benchmark problem set.

The proposed method is a hybrid approach based on genetic algorithm. Genetic algorithm have been proven effective in many combinatorial optimization problems, and it seems natural to apply the approach to a mixed-model assembly line balancing problem (Haq, Jayaprakash, and Rengarajan, 2006). To improve the capability of searching for a good solution, we introduce a sequential hybrid genetic algorithm with the modified versions of three well known heuristics, Ranked Positional Weight Technique (RPWT), Kilbridge and Wester Heuristic and Phase-I of Moodie and Young Method. The modifications of these heuristics will be explained later. In the proposed hybrid genetic algorithm, the initial solutions are obtained from the modified versions of these three heuristics and included in the initial population of the genetic algorithm.

We select these three heuristics because they perform well according to performance criterions (Ponnambalam et al., 1999) of average line efficiency and average smoothness index. We use the average line efficiency to test the performance

of the proposed hybrid genetic algorithm and the fitness function that we used has components, workload balance between workstations and workload balance within each workstation, based on the average smoothness index. Second term of the fitness function minimizes the workload balance between workstations and third term minimizes workload balance within each workstation. The fitness function will be explained later in details.

RPWT uses the priority of positonal weight, Kilbridge and Wester Heuristic uses the priority of minimum number of successor and Moodie and Young Method uses the priority of maximum task time rule when assigning tasks to workstations. Thus the obtained solutions with these heuristics are from different regions of the search space. When we included these solutions in the initial population of the genetic algorithm we make the hybrid algorithm drive to different regions of the search space. This type of hybrid reduces the search space from the global space and increases the probability finding the global optimum.

4.1.1 Assigning Parallel Workstations

Parallel lines provides increasing flexibility and decreasing failure sensitivity of the production system. Also, the risk of production stoppage due to machine breakdowns is significantly reduced by implementation of parallel lines. Additional advantages can be obtained such as better line balances, due to the higher number of possible task combinations and job enrichment, as the operators perform a larger number of different tasks.

The implementation of parallel workstations is necessary to achieve the required production rate in case of the production rate required to meet the demand is so high that the processing times of some of the tasks exceed cycle time. Different workpieces are distributed among several operators who perform the same tasks, when parallelling workstations. The local cycle time (the capacity in time units of parallelled workstation) in these workstations is a multiple of the global cycle time, depending on the number of replicas installed. It is possible to decrease cycle time for the same number of operators with the use of parallel workstations.

4.1.2 Assumptions and Constraints of the Problem

A set of similar models of a product (m=1,…,M) assembled on the line, in any order or mix, over a pre-specified planning horizon, P. The forecasted demand, over the planning horizon, for model m is Dm, requiring the line to be operated with a cycle time given by equation (4.1).

$$C = P/\sum_{m=1}^{M} D_m \tag{4.1}$$

The overall proportion of the number of units of model m being assembled, i.e., the production share of each model, is computed by equation (4.2).

$$q_m = D_m/\sum_{p=1}^{M} D_p \tag{4.2}$$

There is a subset of tasks common to all models, however each model has its own set of precedence diagrams. As a result, the precedence diagrams for all the models can be combined in order to form joint precendence diagram and the joint precedence diagram has N tasks. The N tasks are performed in a set of workstations, S. The time required to perform task i on model m, t_{im}, may vary among models (t_{im}=0 means that model m does not require task i).

A task can be assigned to only one workstation and, consequently, the tasks that are common to several models need to be performed on the same workstation. Equation (4.3) ensures assigning a task to only one workstation.

$$\sum_{k=1}^{S} x_{ik}=1 \qquad i = 1,.....,N \tag{4.3}$$

Successors of task i, F_i, derived from joint precedence diagram, can not be performed before task i completed. There is no possibility of assigning any successors of task i to an earlier workstation than the workstation to which task i is assigned. Equation (4.4) ensures this set of precedence constraints.

$$\sum_{k=1}^{S} x_{ak} - \sum_{k=1}^{S} x_{bk} \leq 0 \qquad a \in N, b \in F_i \tag{4.4}$$

Minimum replication time (MRT) is defined in order to decide when a workstation needs to duplicate. Workstations, which perform tasks with processing time higher than MRT for one of the models at least, allowed to be replicated. A workstation can be duplicated up to a maximum number of replicas (MAXP). The number of replicas of a workstation k, R_k, is determined by its longest task processing time (for all models) and it is given by equation (4.5).

$$R_k = \left\lceil \frac{\max\limits_{m=1,..,M;i=1,..,N} \left\{ t_{im} x_{ik} \right\}}{MRT} \right\rceil \qquad k = 1,.....,S \tag{4.5}$$

At this point, it is necessary to distinguish between the total number of operators (S') working on the line and the number of different workstations in the line (S). The total number of operators working on the line computed by the sum of workstations including all replicas, equation (4.6).

$$S' = \sum_{k=1}^{S} R_k \tag{4.6}$$

A workstation's capacity depend on the processing times of tasks assigned to it. The workload of workstation k for model m, W_{km}, after the assignment of candidate task h, defined as the sum of the task processing times for each model assigned to workstation k plus the processing time of task h, and given by equation (4.7).

$$W_{km} = \sum_{i=1}^{N} t_{im} x_{ik} + t_{hm} \qquad k = 1,.....,S; m = 1,.....,M \tag{4.7}$$

It is also necessary defining the capacity constraints if workstation k performs a task with a processing time higher than MRT, for at least one of the models. The

capacity such a workstation is required to perform the task wtih processing time higher than MRT. Equation (4.8) ensures this set of constraints.

$$W_{km} \le R_k C \qquad k = 1,.....,S; m = 1,.....,M \tag{4.8}$$

Zoning constraints, negative or positive, are also taken into consideration. Positive zoning constraints (equation (4.9) where ZP represents the set of pairs of tasks that must be assigned to the same workstation) deal with the pairs of task must be assigned to the same station and negative zoning constraints (equation (4.10) where ZN is the set of pairs of incompatible tasks) forbid the assignment of pairs of tasks to the same workstation.

$$\sum_{k=1}^{S} kx_{ik} - \sum_{k=1}^{S} kx_{jk} = 0 \qquad ((i, j) \in ZP) \tag{4.9}$$

$$\sum_{k=1}^{S} kx_{ik} - \sum_{k=1}^{S} kx_{jk} \neq 0 \qquad ((i, j) \in ZN) \tag{4.10}$$

Equations (4.11), (4.12) and (4.13) describe the decision variables.

$$x_{ik} = \begin{cases} 1, & \text{if task } i \text{ assigned to workstation } k \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

$$r_k = \begin{cases} 1, & \text{if workstation } k \text{ can be replicated} \\ 0, & \text{otherwise} \end{cases} \tag{4.12}$$

$$s_{km} = \text{idle time of workstation } k \text{ due to model } m \tag{4.13}$$

The idle time of a workstation is the difference between the capacity of the workstation and its workload and calculated by equation (4.14).

$$\sum_{i=1}^{N} t_{im} x_{ik} + s_{km} = R_k C \qquad k = 1,.....,S; m = 1,.....,M \tag{4.14}$$

4.1.3 Objective Function

ALB Type-1 problem aims at minimising total number of workstations for a predefined cycle time as the main purpose. For that reason the first term of the used objective function minimizes the index of the workstation to which the last task is assigned, hence minimising the number of workstations. Equation (4.15) shows this calculation.

$$minimise = \sum_{k=1}^{S} kx_{Nk} \tag{4.15}$$

Additional goals, concerning workload smoothing, can also be taken into consideration besides minising total number of worksations. The second term of the objective function aims at balancing the workload between workstations, while distributing idle time across workstations as equally as possible for each model and given by equation (4.16).

$$minimise = \frac{S'}{S'-1} \sum_{m=1}^{M} q_m \sum_{k=1}^{s'} \left(\frac{s_{km}}{\sum_{l=1}^{s'} s_{lm}} - \frac{1}{S'} \right)^2 \tag{4.16}$$

Each task processing time may vary among the different models and, so, the workload assigned to a workstation may also vary, due to the mixed-model nature of the problem. The third term of the objective function aims at balancing the workload within each station in order to ensure that each operator performs approximately the same amount of work for each model being assembled. Equation (4.17) aims at smoothing the workload balance wtihin each workstation.

$$minimise = \frac{M}{S'(M-1)} \sum_{k=1}^{S} \sum_{m=1}^{M} \left(\frac{q_m s_{km}}{\sum_{m=1}^{M} q_m s_{km}} - \frac{1}{M} \right)^2 \tag{4.17}$$

$$\sum_{m=1}^{M} q_m s_{km} = S_k \tag{4.18}$$

4.1.4 Complete Mathematical Model

The mathematical model for mixed model assembly line balancing with parallel workstations (Vilarinho and Simaria, 2002) presented in Figure 4.1.

The first term in the objective function (1) minimizes the index of the workstation to which the last task is assigned, thus minimizing the number of workstations. The second term balances the workload between the workstations. The third term balances the workload within each workstation.

(2) constraints ensuring that each task is assigned to only one workstation of the station interval.

(3) constraints ensuring that no successor of a task is assigned to an earlier station than that task.

(4) compatibility zoning constraints.

(5) incompatibility zoning constraints.

(6) the set of constraints ensuring that: each workstation time capacity is not exceeded. The maximum number of replicas of a workstation is not exceeded and only workstations where the processing time of the tasks assigned to it, for at least one model, exceeds a certain proportion (α%) of the cycle time can be duplicated (μ is a very large positive integer).

(7) the set of constraints defining the decision variables domains.

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

subject to:

$$\sum_{k=1}^{s} x_{ik} = 1 \qquad i = 1, ..., N \qquad (2)$$

$$\sum_{k=1}^{s} x_{ak} - \sum_{k=1}^{s} x_{bk} \le 0 \qquad a \in N, b \in F_a \qquad (3)$$

$$\sum_{k=1}^{s} x_{ak} - \sum_{k=1}^{s} x_{bk} = 0 \qquad (a,b) \in ZP \qquad (4)$$

$$x_{ak} + x_{bk} \le 1 \qquad (a,b) \in ZN, k = 1, ..., S \qquad (5)$$

$$\sum_{i=1}^{N} t_{im}.x_{ik} + s_{km} = C\left[1 + r_k\left(MAXP - 1\right)\right] \qquad k = 1, ..., S, m = 1, ..., M \qquad (6a)$$

$$r_k \le \sum_{i:\exists t_{im} > \alpha C; m=1,...,M}^{n} x_{ik} \qquad k = 1, ..., S, 0 \le \alpha \le 100\% \qquad (6b)$$

$$\mu r_k \le \sum_{i:\exists t_{im} > \alpha C; m=1,...,M}^{n} x_{ik} \qquad k = 1, ..., S, 0 \le \alpha \le 100\% \qquad (6c)$$

$$s_{km} \ge 0 \qquad k = 1, ..., S, m = 1, ..., M \qquad (7a)$$

$$x_{ik} \in [0,1] \qquad k = 1, ..., S, i = 1, ..., N \qquad (7b)$$

$$r_k \in [0,1] \qquad k = 1, ..., S \qquad (7c)$$

Figure 4.1 Mathematical programming model for the mixed-model ALBP with parallel workstations under the zoning constraints

4.2 Hybrid Genetic Algorithm

The complexity of the proposed mathematical model is high and it can not be solved to optimality, at least for real world problems (Vilarinho and Simaria, 2002). For that reason a two stage procedure (Vilarinho and Simaria, 2002) based on simulated annealing technique and ANTBAL (Vilarinho and Simario, and 2006), an ant colony optimization algorithm for balancing mixed-model assembly lines, were proposed by the authors to tackle the problem. Both of the approaches were tested on a set of 20 problems, whose main charecteristics will be given later. Their computational experience shows that ANTBAL performs better than the simulated annealing based algorithm.

In the current study we propose a sequential hybrid genetic algorithm to solve the MMALBP of type-1. Then we test the performance of the proposed hybrid genetic algorithm on a benchmark problem set. In Figure 4.2, flow diagram of the proposed algorithm is given.

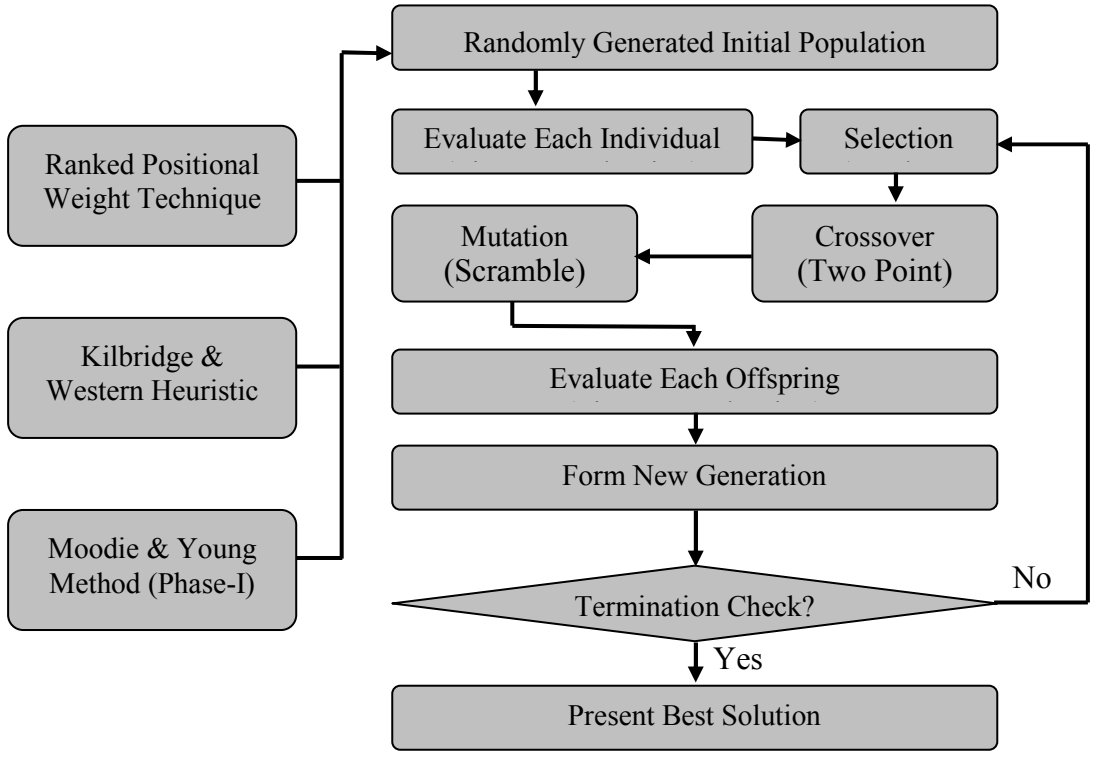


Figure 4.2 Proposed hybrid genetic algorithm

4.2.1 Representation of Solutions

We deal with MMALBP of type-1 problem, minimizing the number of workstations for a predefined cycle time, in this study. Therefore we used task based representation (Leu et al., 1994; Ajenblit and Wainwright, 1998; Sabuncuoglu et al., 2000), which is the most appropriate repsesentation scheme (chromosome type) for type-1 problems. An example of this type of chromosome was already presented in Figure 3.9 of chapter 3.

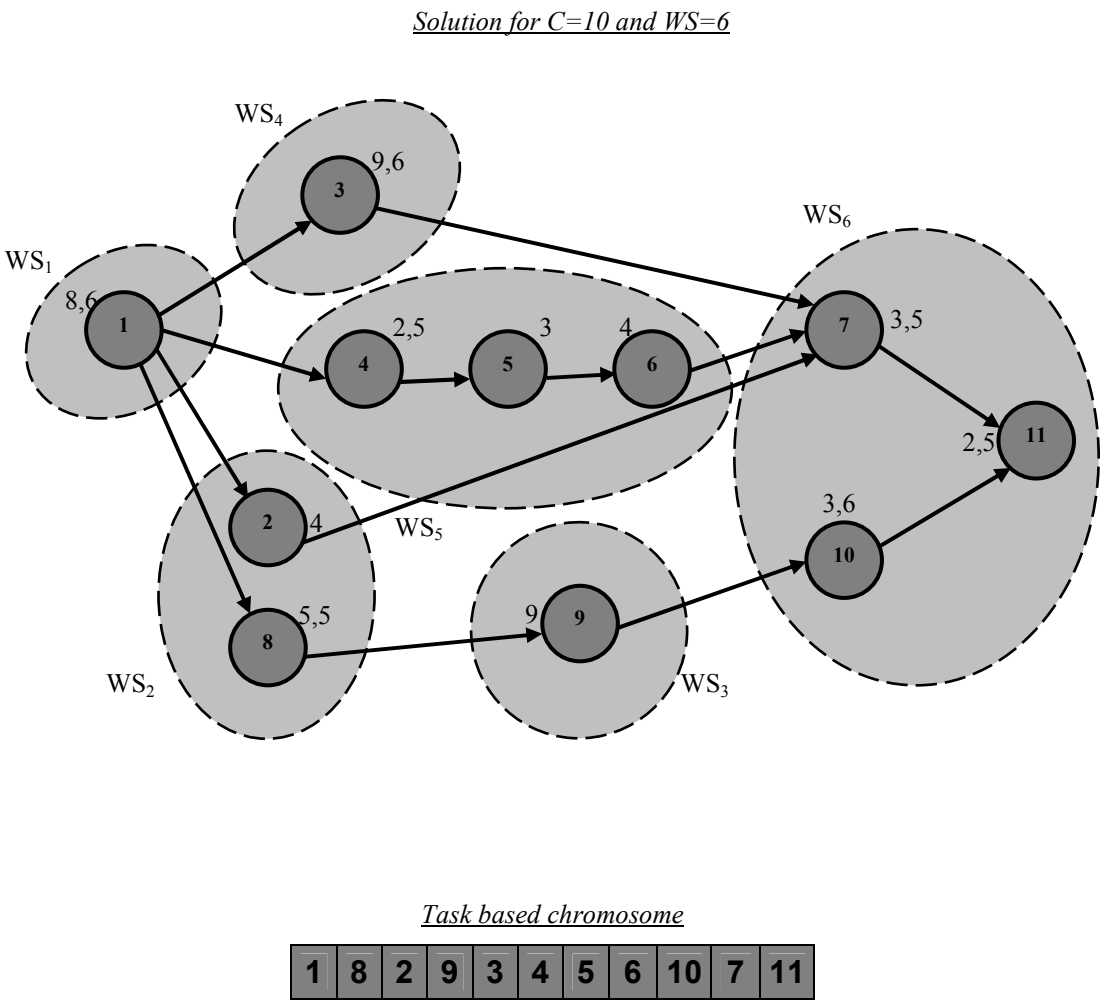


Figure 4.3 Assigment procedure according to chromosome

The length of the chromosome is defined by the number of tasks and each element of the chromosome represents a task. Tasks are assigned to workstations acording to the task sequence in the chromosome, as long as the predetermined cycle time is not

exceeded. Once the cycle time is exceeded, a new workstation is opened for assignment, and the process is repeated. Figure 4.3 illustrates an example of assignment of tasks to workstations according to a chromosome.

4.2.2 Initial Population

In the proposed hybrid genetic algorithm, the initial population is generated randomly. The well known problem specific algorithms, Ranked Positional Weight Technique (Helgeson and Birnie, 1961), Kilbridge and Wester Heuristic (Kilbridge and Wester, 1961) and Phase-I of Moodie and Young Method (Moodie and Young, 1965), are used to generate initial solutions. Then, these generated solutions are included in the randomly generated initial population. This approach has an advantage of reducing the search space from the size of the global search space. Using the different algorithms for initialize the genetic algorithm provides another advantage, starting the search from different points of global search space, hence driving the search space towards a promissing area of the global search space and increasing the probability of finding good solutions.

The original versions of these algorithms only address the simple assembly line balancing problem, where one single model is assembled and no parallel workstations are allowed. For applying ranked positional weight technique positional weights of each task must be calculated. The positional weight of a task in a mixed-model assembly line is the cumulative average task time associated with itself and its successors. The average task time is the sum of the processing times of that task for each model weighted by the respective production share. Tasks are assigned to the lowest numbered feasible workstation by decreasing order of their positional weight and considering the individual task processing times for each model. In Kilbridge and Wester Heuristic, numbers are assigned to each operation describing how many predecessors it has. Operations with the lowest predecessor number are assigned first to the workstations. The predecessors numbers for each task are derived from joint precedence diagram and assigmnets are made by taking into consideration the task processing times for each model in a mixed-model assembly line. Moodie and Young

Method uses two types of matrixes, **P** (immediate predecessors of each task) and **F** (immediate followers of each task). Tasks are assigned to consecutive workstations on the assembly line by the largest candidate rule (maximum task time rule). In mixed-model lines the matrixes **P** and **F** are derived from joint precedence diagram and tasks are assigned to workstations by considering processing times for each model in a mixed-model assembly line. In implementation of these three algorithms for mixed-model assembly lines with parallel workstations, if a workstation performs a task with processing time larger than minimum replication time (here cycle time), for at least one model, its time capacity is set as C. MAXP (maximum number of replication) in order to procure the parallel workstation assigment.

4.2.3 Fitness Evaluation

Genetic algorithms aim at finding the most fit chromosome over a set of generations. The fitness function provides a measure of an individual's performance (fitness) in the search space. We used the objective function of the mathematical programming model for the mixed-model ALBP with parallel workstations under the zoning constraints as the fitness function (Equation 4.19) of hybrid genetic algorithm. The main goal is to minimize the number of workstations while smoothing the workload balance between workstations and within each workstation. The fitness function is then, typically, a minimisation function.

$$minZ = \sum_{k=1}^{s} k.x_{Nk} + \frac{S'}{S'-1} \sum_{m=1}^{M} q_m \sum_{k=1}^{s'} \left(\frac{s_{km}}{\sum_{l=1}^{s'} s_{lm}} - \frac{1}{S'} \right)^2 + \frac{M}{S'(M-1)} \sum_{k=1}^{s'} \sum_{m=1}^{M} \left(\frac{q_m s_{km}}{S_k} - \frac{1}{M} \right)^2 \quad (4.19)$$

4.2.4 Selection

The selection of the individuals for mating is done using roulette whell selection (Holland, 1975), which is the best known selection strategy. Roulette wheel selection scheme, which scales the fitness values of the members within the population so that

the sum of the rescaled fitness values equals to 1, is used. To select a parent, first, a uniform random number within the interval (0, 1) is generated (wheel is spun), and then the member whose cumulative rescaled fitness value is greater than the generated number is selected as parent. The steps of the used roulette whell selection strategy can be summarised in below:

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

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

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

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

4.2.5 Genetic Operators

Genetic operators, crossover and mutation, has considerable effect upon the balancing performance. Crossover is the operation by which two individuals in the current population are used in order to create offsprings for the next population. Mutation operator changes the value of single genes within chromosomes randomly.

We used *two point crossover* and *scramble mutation* (Leu et al., 1994). Both types of these genetic operators guarantee the feasibility of individuals in the current population by forcing. During genetic operations, only chromosomes, i.e., feasible sequence of tasks are employed. Besides, the workstation assignments remain untouched until the new offspring is completely formed. After offspring is formed, the tasks are assigned to workstations and individual's fitness is evaluated.

4.2.5.1 Two Point Crossover

In the two point crossover, two points, which cut each of the parent into three parts (head (H), middle (M) and tail (T)), are generated randomly. In a general genetic algorithm recombination, new offspring are created by swapping the middle sections of the parents' chromosomes, i.e., parent-1, represented by $H_1M_1T_1$, recombines with parent-2, represented by $H_2M_2T_2$, in order to form the new children $H_1M_2T_1$ and $H_2M_1T_2$. An example of two point crossover was already presented in Figure 3.6 of chapter 3.

In the assembly line balancing problem the situation is not so simple because of the feasibility problem. Recombination must guarantee feasibility. Two point crossover applied as in Figure 4.4 provides this guarantee. The resulting offspring are always feasible.

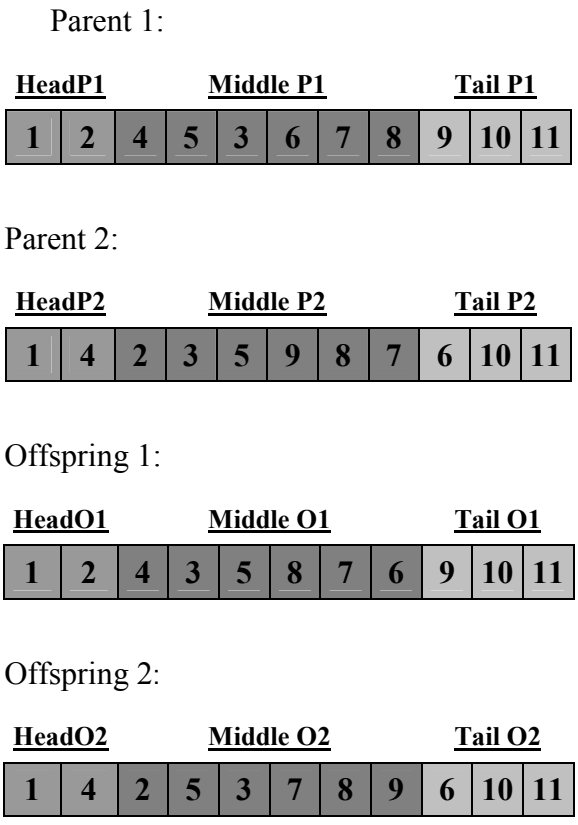


Figure 4.4 Recombination: Two point crossover

The first offspring keeps the head and the tail parts of the first parent. The middle part of the first offspring is filled in by adding the all missing tasks in the order in which they are contained in the second parent. The other offspring is built analogously based on the head and the tail parts of the second parent and its middle part is filled in by adding the missing tasks in the order in which they are contained in the first parent. Both of the generated offsprings become feasible as their middle part is also filled according to the precedence feasible order. The purpose of the two-point crossover is to conduct a neighborhood search; this is done by keeping the head and tail of each child the same as its parent. The child should be "close" in fitness to its parents because only its middle genes have changed (Leu et al., 1994).

4.2.5.2 Scramble Mutation

In the scramble mutation, first a random point, where the mutation occurs, is selected. After that the head from the chosen parent is placed on the new, mutated offspring. Then the tail of the new child is reconstructed using the procedure explained in below. This procedure uses a table, called prohibit table (see Table 4.1), and also guarantees the feasibility.

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

For example, consider parent 1 (1-2-4-5-3-6-7-8-9-10-11) in Figure 4.4 and assume the mutation point is chosen to be after task 3. Then the head of the child will be 1-2-4-5-3. The rest of the child will consist of tasks 6, 7, 8, 9, 10, and 11, placed in random order, but in such a manner as not to violate precedence constraints. If the original prohibit table is as shown on the Table 4.1, then the prohibit table with all references to tasks in the head of the child will be as shown on Table 4.2.

Table 4.1 Original prohibit table

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

Table 4.2 Modified prohibit table

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

Only tasks 6, 7, 8, and 9 can be selected to follow task 3 since they alone have no "can not precede" tasks in the modified prohibit table; assume they are chosen randomly in the order 9, 7, 8, 6. Then the final prohibit table (with these tasks deleted) becomes as Table 4.3.

Table 4.3 Final prohibit table

Task	Can Not Precede
11	**10**
10	**-**

Therefore, task 10 must be selected next, and 11 must be chosen last. The new child chosen with scramble mutation is 1-2-4-5-3-9-7-8-6-10-11 (see Figure 4.5).

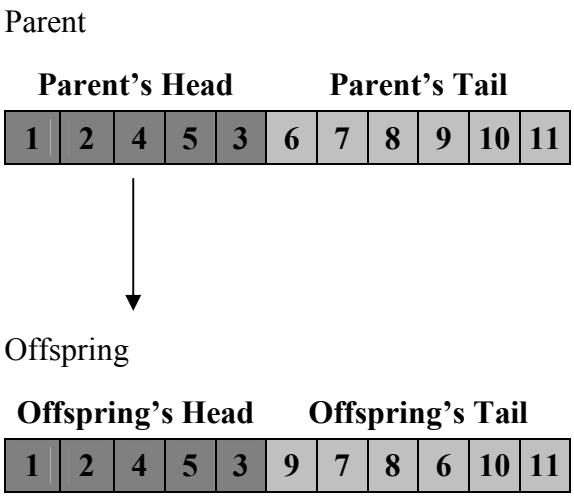


Figure 4.5 Scramble mutation

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

4.2.6 New Generation

The replacement strategy determines which individuals stay in the population and which are replaced and it takes into account the fitness value of the individuals. As mentioned in chapter 3, the individuals of the new generation may be (i) individuals from the current generation, (ii) offspring product of crossover or (iii) individuals who underwent mutation. The most commonly used relacement strategy is elitism, which makes survive some number of the best individuals at each generation, hence guaranteeing that the final population contains the best solution ever found. We used at this study an elitist replacement strategy, which builds 25% of the new generation by surviving the related number of best individulas. The other individuals of the new

generation are seleceted randomly among the current population, the offspring product of crossover and the individuals who underwent mutation.

4.2.7 Termination Criteria of the Algorithm

Searching the solution space can be ended when one of the following conditions is achieved.

- ✓ The fitness function of the best solution does not improve more than x% (x, user defined) after a predetermined number of consecutive iterations.
- ✓ The total number of iterations exceeds a maximum number.

In this study we stopped the proposed algorithm when the total number of iterations exceeds a predefined number (250 iterations).

4.2.8 Parameter Setting

The selection of genetic algorithm parameters such as the size of the population, the rate of mutation and crossover have great importance on ensuring high performance. If the population size is too small, the genetic algorithm may not explore the solution space extensively and consistently to find good solutions. If the rate of genetic change is too high, beneficial schema may be disrupted and the population may enter error catastrophe. A genetic algorithm can be defined by the control parameter set $\pi = \{P_S, R_C, R_M\}$ where;

- ✓ P_S: The size of the population
- ✓ R_C: The rate of crossover
- ✓ R_M: The rate of mutation

In this study we set the rate of crossover as $R_C = 0.5$ and the rate of the mutation as $R_M=0.15$. The size of population is a parameter related to search space, which

differs from problem to problem. For that reason we did not set a value for the size of population. We set different population sizes for each problem. Table 4.4 contains the parameter set used for the proposed hybrid genetic algorithm.

Table 4.4 Parameter set

Problem	P_S	R_C	R_M	Problem	P_S	R_C	R_M
1	10	0.5	0.15	2	10	0.5	0.15
3	20	0.5	0.15	4	20	0.5	0.15
5	30	0.5	0.15	6	30	0.5	0.15
7	40	0.5	0.15	8	40	0.5	0.15
9	50	0.5	0.15	10	50	0.5	0.15
11	60	0.5	0.15	12	60	0.5	0.15
13	70	0.5	0.15	14	70	0.5	0.15
15	80	0.5	0.15	16	80	0.5	0.15
17	90	0.5	0.15	18	90	0.5	0.15
19	100	0.5	0.15	20	100	0.5	0.15

4.3 Computational Experiments

To evaluate the performance of the proposed hybrid GA approache on MMALBP with parallel workstation assignment, we solved a benchmark set of 20 problems given by Vilarinho and Simaria, 2002. The proposed hybrid genetic algorithm was coded in Matlab 7.2. A direct comparison can be performed with the results obtained using a simulated annealing procedure (Vilarinho and Simaria, 2002) and ANTBAL (Vilarinho and Simaria, 2006) in a set of 20 mixed-model assembly line balancing problem with parallel workstations and zoning constraints. For this set of problems the number of operators (S) and the weighted line efficiency (WE) of the solutions provided by hybrid genetic algorithm were compared with the lower bound on the total number of operators (LB_{pmix}) derived by Vilarinho and Simaria (2002). The obtained solutions were also compared with the solutions provided by the simulated annealing procedure and with the solutions provided by ANTBAL.

The goal of the MMALBP Type-1 problem is minimizing the total number of workstations (S) for a given cycle time (C), equivalent with maximizing WE (equation 4.20).

$$WE = \sum_{m=1}^{M} \left[q_m \frac{\sum_{i=1}^{N} t_{im}}{S'C} \right] \tag{4.20}$$

The lower bound for MMALBP with parallel workstations, LB_{pmix}, was derived by Vilarinho and Simaria (2006) using the following three assumptions:

- ✓ The maximum number of replicas per workstation is two (MAXP=2).
- ✓ A workstation can be duplicated only if the task time of one of the tasks assigned to it exceeds the cycle time (α=100%, MRT=C).
- ✓ The task time of the longest task does not exceed twice the cycle time (tmax≤2C).

The steps required to compute the LB_{pmix} are as follows:

Step 1: For each model, classify the tasks according to the corresponding task time, as shown in Table 4.5.

Step 2: For each model, compute $LB'(m)$ (Equation 4.21).

$$LB'(m) = \left[(2\left(n_A + n_B + n_C\right) + y\left(n_D - n_C\right) + \frac{1}{n} w\left(n_E - n_B\right) + \frac{5}{3} n_F + \frac{4}{3} n_G + \frac{2}{3} n_H + \frac{1}{3} n_I) \right] \tag{4.21}$$

Step 3: For each model, compute $Z(m)$ (Equation 2.22).

$$Z(m) = \left[\left[\sum_{i=J} t_i - \left(LB'(m)C - \sum_{i \neq J} t_i \right) \right] / C \right] \tag{4.22}$$

Step 4: For each model, compute $LB_{pmix}(m) = LB'(m) + Z(m)$.

Step 5: Select LB_{pmix} for the problem. LBp_{mix}=maxm[LB_{pmix}(m)].

For more information about the calculation of this lower bound the reader can refer to Vilarinho and Simaria (2002)

Table 4.5 Classification of the task to compute LB_{pmix}

Task Type	Task Time
A	$\frac{5}{3}C < t_A \leq 2C$
B	$\frac{4}{3}C < t_B \leq \frac{5}{3}C$
C	$C < t_C \leq \frac{4}{3}C$
D	$\frac{2}{3}C < t_D \leq C$
E	$\frac{1}{3}C < t_E \leq \frac{2}{3}C$
F	$t_F = \frac{5}{3}C$
G	$t_G = \frac{4}{3}C$
H	$t_H = \frac{2}{3}C$
I	$t_I = \frac{1}{3}C$
J	$t_J < \frac{1}{3}C$

4.3.1 Benchmark Problem Set

Table 4.6 contains the benchmark set of 20 problems (Vilarinho and Simaria, 2002), whose main characteristics are shown in the second, third and fourth columns respectively, the number of tasks of the combined precedence diagram (N), the number of models (M) and the assembly line cycle time (C). The precedence diagrams used for the problem set are shown in the last column. Additional charateristics, model mix, model demands, incopatible tasks and production periods, can also be seen in the table.

Table 4.6 Characteristics of the benchmark problem set

Problem ID	N	M	C	Model Mix			Incompatible Tasks	Model Demand			Production Period	Precedence Relations
				q_A	q_B	q_C		m_A	m_B	m_C		
1	8	2	10	0.42	0.58	-	3;4	20	28	-	480	Bowman
2	8	3	10	0.33	0.50	0.17	3;4	16	24	8	480	Bowman
3	11	2	10	0.42	0.58	-	7;8	20	28	-	480	Gokcen & Erel (1998)
4	11	3	10	0.33	0.50	0.17	7;8	16	24	8	480	Gokcen & Erel (1998)
5	21	2	10	0.42	0.58	-	7;8	20	28	-	480	Mitchel
6	21	3	10	0.33	0.50	0.17	7;8	16	24	8	480	Mitchel
7	25	2	10	0.42	0.58	-	9;10	20	28	-	480	Vilarinho & Simaria (2002)
8	25	3	10	0.33	0.50	0.17	9;10	16	24	8	480	Vilarinho & Simaria (2002)
9	28	2	10	0.42	0.58	-	4;5	20	28	-	480	Heskiaoff
10	28	3	10	0.33	0.50	0.17	4;5	16	24	8	480	Heskiaoff
11	30	2	10	0.42	0.58	-	21;22	20	28	-	480	Sawyer
12	30	3	10	0.33	0.50	0.17	21;22	16	24	8	480	Sawyer
13	32	2	10	0.42	0.58	-	18;19/23;32	20	28	-	480	Lutz 1
14	32	3	10	0.33	0.50	0.17	18;19/23;32	16	24	8	480	Lutz 1
15	35	2	10	0.42	0.58	-	14;18/31;32	20	28	-	480	Gunther
16	35	3	10	0.33	0.50	0.17	14;18/31;32	16	24	8	480	Gunther
17	45	2	10	0.42	0.58	-	14;30/34;35	20	28	-	480	Kilbridge & Wester
18	45	3	10	0.33	0.50	0.17	14;30/34;35	16	24	8	480	Kilbridge & Wester
19	70	2	10	0.42	0.58	-	14;30/34;35	20	28	-	480	Tonge
20	70	3	10	0.33	0.50	0.17	14;30/34;35	16	24	8	480	Tonge

4.3.2 Results and Discussions

The main goal of this study is to test the performance of the proposed hybrid genetic algorithm on MMALBP with parallel workstations. For hybridization of the genetic algorithm three well known heuristics, Kilbridge and Wester Heuristic, Phase-I of Moodie and Young Method and Ranked Positional Weight Tehnique (RPWT) are used. Both of these approaches only address the simple assembly line balancing problem, where one single model is assembled and no parallel workstations are allowed. In order to apply these methods to mixed-model assembly line balancing problem modified versions, as explained before, of them are used.

First, the problems were solved by the modified versions of these three approaches and by pure genetic algorithm. The results are shown in Table 4.8. According to obtained results pure GA superior to the modified versions of these three problem specific heuristics. On the other hand pure GA has satisfactory results when it compared with ANTBAL and simulated annealing based heuristic.

Finally, the problems were solved by the proposed hybrid genetic algorithm. Table 4.7 contains the obtained results. As the the scope of this study a direct comparison of the proposed algorithm was performed with the results obtained using a simulated annealing procedure and ANTBAL. The values shown in Table 4.7 and Table 4.8 (for pure GA) are the best results of ten runs. For this set of problems the number of operators (S) and the weighted efficiency (WE) of the solutions were compared with the lower bound on the total number of operators (LB_{pmix}). The optimal solutions for the problems 1-2-3-4 were only found by the mathematical model (see Figure 4.1), which was coded in Cplex MIP solver by Vilarinho and Simaria, 2002. The differences between the solutions obtained by the approaches and the lower bound (or the optimal solutions for the problems 1-2-3-4) are depicted by the value of ***D%*** in the correspondent columns of the Tables 4.7 and 4.8. The situation, the calculation of the lower bound does not take into account the precedence and zoning constraints, must take into consideration, when the conclusions made about the value of ***D%***.

Table 4.7 Computational results for benchmark problem set

Problem	N	M	C	LBpmix	Optimal	Simulated Annealing			ANTBAL			***Hybrid Genetic Algorithm***		
						S	WE (%)	D (%)	S	WE (%)	D (%)	***S***	**WE (%)**	**D (%)**
1	8	2	10	4	4	4	85.6	0	4	85.6	0	***4***	***85.6***	***0***
2	8	3	10	6	8	8	54.9	0	8	54.9	0	***8***	***54.9***	***0***
3	11	2	10	7	7	7	71.0	0	7	71.0	0	***7***	***71.0***	***0***
4	11	3	10	6	7	7	76.5	0	7	76.5	0	***7***	***76.5***	***0***
5	21	2	10	14	-	16	72.6	14.3	16	72.6	14.3	***16***	***72.6***	***14.3***
6	21	3	10	13	-	15	79.6	15.4	15	79.6	15.4	***15***	***79.6***	***15.4***
7	25	2	10	14	-	16	76.8	14.3	16	76.8	14.3	***16***	***76.8***	***14.3***
8	25	3	10	14	-	15	82.0	7.1	14	87.9	0	***14***	***87.9***	***0***
9	28	2	10	19	-	21	86.5	10.5	20	90.8	5.3	***20***	***90.8***	***5.3***
10	28	3	10	18	-	20	83.2	11.1	20	83.2	11.1	***20***	***83.2***	***11.1***
11	30	2	10	15	-	16	86.6	6.7	16	86.6	6.7	***16***	***86.6***	***6.7***
12	30	3	10	17	-	19	83.4	11.8	19	83.4	11.8	***19***	***83.4***	***11.8***
13	32	2	10	16	-	19	77.3	18.8	19	77.3	18.8	***19***	***77.3***	***18.8***
14	32	3	10	17	-	19	81.0	11.8	19	81.0	11.8	***19***	***81.0***	***11.8***
15	35	2	10	20	-	24	80.0	20.0	23	83.5	15.0	***23***	***83.5***	***15.0***
16	35	3	10	21	-	24	85.2	14.3	24	85.2	14.3	***24***	***85.2***	***14.3***
17	45	2	10	23	-	25	85.4	8.7	25	85.4	8.7	***25***	***85.4***	***8.7***
18	45	3	10	24	-	28	81.4	16.7	27	84.4	12.5	***27***	***84.4***	***12.5***
19	70	2	10	41	-	44	87.0	7.3	44	87.0	7.3	***44***	***87.0***	***7.3***
20	70	3	10	39	-	44	86.0	12.8	44	86.0	12.8	***44***	***86.0***	***12.8***

For small-sized problems (for which an optimal solution was found) the proposed hybrid genetic algorithm finds the optimal solution. For large-sized problems, the worst performance is for problem 13, where the difference between the solutions obtained and the lower bound is 18.8%. Nevertheless, as the calculation of the lower bound does not take into account the precedence and zoning constraints, it must be take into condideration that the results obtained from the proposed hybrid GA are very well. This conclusion is reinforced by the values for the line efficiency shown in column 14 of Table 4.7 (Weighted Efficiency (%)), where a high line usage rate can be perceived, particularly for the largest sized problems.

Table 4.8 shows that Phase-I of Moodie and Young Method finds the same results in problems 1-3-4-5-6-13, RPWT finds the same results in problems 1-4-5-13 and Kilbridge and Wester Heuristic finds the same results in problems 1-5-13 with the hybrid genetic algorithm. This means that our proposed hybrid genetic algorithm finds the best solution at the first iteration in these six problems. When we compare these three heuristics with each other:

- ✓ Phase-I of Moodie & Young Method outperforms Kilbridge & Wester Heuristic in seven problems (3-5-7-14-18-19-20) and RPWT in six problems (3-6-7-11-17-20)
- ✓ RPWT outperforms Phase-I of Moodie & Young Method in three problems (12-15-16) and Kilbridge & Wester Heuristic in seven problems (5-12-14-15-16-18-19)
- ✓ Kilbridge & Wester Heuristic outperforms RPWT in three problems (7-11-17)

If we compare the pure GA with these heuristics it is clear that, pure GA performs well, however the pure GA gives worst solution in six problems (8-9-15-18-19-20) when it compared with the proposed hybrid genetic algorithm. Namely pure GA performs worst for large-sized problems compared with small-sized problems. The proposed hybrid genetic algorithm overcome this disadvantage of the pure GA by improving these six problems' solution.

Table 4.8 Comparison of the approaches

Problem	N	M	C	LBpmix	Optimal	Kilbridge & Wester			Moodie & Young (Phase I)			RPWT			Pure GA		
						S	WE (%)	D (%)	S	WE (%)	D (%)	S	WE (%)	D (%)	S	WE (%)	D (%)
1	8	2	10	4	4	4	85.6	0	4	85.6	0	4	85.6	0	4	85.6	0
2	8	3	10	6	8	9	48.7	12.5	9	48.7	12.5	9	48.7	12.5	8	54.8	0
3	11	2	10	7	7	8	62.0	14.3	7	70.9	0	8	62.0	14.3	7	70.9	0
4	11	3	10	6	7	7	76.4	0	7	76.4	0	7	76.4	0	7	76.4	0
5	21	2	10	14	-	17	68.2	21.4	16	72.5	14.3	16	72.5	14.3	16	72.5	14.3
6	21	3	10	13	-	16	74.6	23.1	15	79.5	15.4	17	70.2	30.8	15	79.5	15.4
7	25	2	10	14	-	17	72.4	21.4	17	72.4	21.4	18	68.3	28.6	16	76.9	14.3
8	25	3	10	14	-	15	81.9	7.1	15	81.9	7.1	15	81.9	7.1	15	81.9	7.1
9	28	2	10	19	-	21	86.5	10.5	21	86.5	10.5	21	86.5	10.5	21	86.5	10.5
10	28	3	10	18	-	21	79.2	16.7	21	79.2	16.7	21	79.2	16.7	20	83.2	11.1
11	30	2	10	15	-	18	77.9	20.0	18	77.9	20.0	19	73.8	26.7	16	87.7	6.7
12	30	3	10	17	-	21	77.0	23.5	21	77.0	23.5	20	80.9	17.6	19	85.1	11.8
13	32	2	10	16	-	19	77.3	18.8	19	77.3	18.8	19	77.3	18.8	19	77.3	18.8
14	32	3	10	17	-	21	73.3	23.5	20	77.0	17.6	20	77.0	17.6	19	81.1	11.8
15	35	2	10	20	-	25	76.8	25.0	25	76.8	25.0	24	80.0	20.0	24	80.0	20.0
16	35	3	10	21	-	26	78.7	23.8	26	78.7	23.8	25	81.8	19.0	24	85.2	14.3
17	45	2	10	23	-	27	79.7	17.4	27	79.7	17.4	28	76.9	21.7	25	86.1	8.7
18	45	3	10	24	-	31	74.1	29.2	29	79.2	20.8	29	79.2	20.8	28	82.0	16.7
19	70	2	10	41	-	50	76.5	22.0	48	79.7	17.1	48	79.7	17.1	45	85.0	9.8
20	70	3	10	39	-	48	79.2	23.1	47	80.9	20.5	48	79.2	23.1	45	84.5	15.4

In considering the total number of workstations obtained by algorithms, hybrid genetic algorithm outperformed pure GA, Kilbridge & Wester Heuristic, Phase-I of Moodie & Young Method and Ranked Positional Weight Technique. The other performance criterion, weighted efficiency and %D are also depend on number of total number of workstations. Within the framework of main goal of this study when the hybrid genetic algorithm compared with ANTBAL and simulated annealing procedure, the obtained results show that, the hybrid genetic algorithm outperformed the simulated annealing procedure. The results also show that hybrid genetic algorithm's performance is as good as ANTBAL's performance. Our hybrid genetic algorithm produced the same results with ANTBAL. When the solutions obtained with the simulated annealing procedure compared with the solutions obtained with the proposed hybrid genetic algorithm, hybrid genetic algorithm improved the solution in four problem instances (problems 8-9-15-18), reaching in one of them (problem 8) the lower bound, thus guaranteeing the optimum.

CHAPTER FIVE
CONCLUSION

The aim of this thesis was to address the MMALBP with parallel workstations by providing a hybrid genetic algorithm to efficiently tackle the problem. First, for balancing mixed-model assembly lines with straight line configuration, the modified versions of three well known heuristics, Kilbridge & Wester Heuristic, Phase-I of Moodie & Young Method and Ranken Positional Weight Technique, and pure genetic algorithm were used. We modified Kilbridge & Wester and Phase-I of Moodie & Young methods. Modified version of RPWT is taken from Vilarinho and Simaria (2002). Their performance was compared through a set of computational experiments. The major contribution of this approach was to address problems with characteristics that reflect some operating conditions of real world assembly lines (e.g., use of parallel workstations, zoning constraints).

The original versions of these algorithms only address the simple assembly line balancing problem, where one single model is assembled and no parallel workstations are allowed. For applying ranked positional weight technique positional weights of each task were calculated. The positional weight of a task in a mixed-model assembly line is the cumulative average task time associated with itself and its successors. The average task time is the sum of the processing times of that task for each model weighted by the respective production share. Tasks were assigned to the lowest numbered feasible workstation by decreasing order of their positional weight and considering the individual task processing times for each model. In Kilbridge and Wester Heuristic, numbers were assigned to each operation describing how many predecessors it has. Operations with the lowest predecessor number were assigned first to the workstations. The predecessors numbers for each task were derived from joint precedence diagram and assigmnets were made by taking into consideration the task processing times for each model in a mixed-model assembly line. Moodie and Young Method uses two types of matrixes, **P** (immediate predecessors of each task) and **F** (immediate followers of each task). Tasks were assigned to consecutive workstations on the assembly line by the largest candidate

rule (maximum task time rule). In mixed-model lines the matrixes **P** and **F** were derived from joint precedence diagram and tasks were assigned to workstations by considering processing times for each model in a mixed-model assembly line.

Second, a hybrid genetic algorithm, which aims at deriving benefits from the advantages of the aforementioned three heuristics and overcoming the disadvantages of them, was proposed. The hybrid type was selected as sequential. In the proposed hybrid genetic algorithm, the initial population was generated randomly. The modified versions of Ranked Positional Weight Technique, Kilbridge & Wester Heuristic, and Phase-I of Moodie & Young Method were also used to generate initial solutions. Then, the generated solutions were included in the randomly generated initial population. Namely, the solutions generated by the heuristic methods are introduced randomly into initial population and proceed by the GA method. . This approach has an advantage of reducing the search space from the size of the global search space. Using different algorithms for initializing the genetic algorithm provided another advantage, starting the search from different points of the global search space, hence driving the search space towards different zones of the global search space and increasing the probability of finding good solutions.

The performance of the proposed hybrid genetic algorithm was also tested on a benchmark data set including 20 MMALBPs of type 1. Table 5.1 shows a summary of these computational experiments. This table gives the comparisons of the proposed hybrid genetic algorithm with other methods and contains two criteria for comparison, odds value and improvement in percentage. The odds value is determined as the difference between the number of workstation obtained by the proposed hybrid GA and that by another algorithm.

As a result of this comparisons, the proposed hybrid genetic algorithm superior Kilbridge & Wester Heuristic in 16 (%80 of the total number of problems) problems, Phase-I of Moodie & and Young Method in 14 (%70 of the total number of problems) problems, RPWT in 16 (%80 of the total number of problems) problems, pure GA in 6 (%3 of the total number of problems) problems and simulated annealing based method in 4 (%2 of the total number of problems) problems.

Table 5.1 Comparision of the proposed hybrid GA with other algorithms

PROBLEM ID	Kilbridge & Wester - Hybrid GA		Moodie & Young - Hybrid GA		RPWT - Hybrid GA		Pure GA - Hybrid GA		Simulated Annealing - Hybrid GA		ANTBAL - Hybrid GA	
	odds	improvement (%)	odds	improvement (%)	odds	improvement (%)	odds	improvement (%)	odds	improvement (%)	odds	improvement (%)
Problem 1	0	0	0	0	0	0	0	0	0	0	0	0
Problem 2	1	11.11	1	11.11	1	11.11	0	0	0	0	0	0
Problem 3	1	12.5	0	0	1	12.5	0	0	0	0	0	0
Problem 4	0	0	0	0	0	0	0	0	0	0	0	0
Problem 5	1	5.88	0	0	0	0	0	0	0	0	0	0
Problem 6	1	6.25	0	0	2	11.76	0	0	0	0	0	0
Problem 7	1	5.88	1	5.88	2	11.11	0	0	0	0	0	0
Problem 8	1	6.67	1	6.67	1	6.67	1	6.67	1	6.67	0	0
Problem 9	1	4.76	1	4.76	1	4.76	1	4.76	1	4.76	0	0
Problem 10	1	4.76	1	4.76	1	4.76	0	0	0	0	0	0
Problem 11	2	11.11	2	11.11	3	15.79	0	0	0	0	0	0
Problem 12	2	9.52	2	9.52	1	5	0	0	0	0	0	0
Problem 13	0	0	0	0	0	0	0	0	0	0	0	0
Problem 14	2	9.52	1	5	1	5	0	0	0	0	0	0
Problem 15	2	8	2	8	1	4.17	1	4.17	1	4.17	0	0
Problem 16	2	7.69	2	7.69	1	4	0	0	0	0	0	0
Problem 17	2	7.41	2	7.41	3	10.71	0	0	0	0	0	0
Problem 18	4	12.9	2	6.9	2	6.9	1	3.57	1	3.57	0	0
Problem 19	6	12	4	8.33	4	8.33	1	2.22	0	0	0	0
Problem 20	4	8.33	3	6.38	4	8.33	1	2.22	0	0	0	0

On the other hand the proposed hybrid GA has the same results with ANTBAL. Thus hybrid GA outperforms other algorithms except ANTBAL, however performs as well as ANTBAL. ANTBAL is an Ant Colony Algorithm (ACO) which simulates group behaviors between communities and environment made up with simple individual, as well as individuals, which may cause unpredictable group behavior. But GA simulate genetic evolutional process. On the other hand, ACO converges to the optimal path through the accumulation and update of information, but the lack of pheromone in initial stages leads to slower speed of convergence. Genetic Algorithm have rapid global searching capability, but the feedback of information in the system has not been utilized, sometimes leading to do-nothing redundant iteration and inefficient solution. Therefore, if the size of the problem gets larger pure GA's searching capability will gradually decline to a certain extent. Our proposed hybrid genetic algorithm overcomes this disadvantage of pure GA by reducing the search space from the size of the global space. Both the proposed hybrid genetic algorithm and ANTBAL have not special requirements to searching space, such as derivative, continuity, concavo-convex and other auxiliary information. However, ANTBAL has priority rules, which require some further computational effort, for selecting a task for assigment. In this sense, our proposed hybrid genetic algorithm performs less computational effort than ANTBAL while assigning tasks to workstations.

These results showed that the proposed algorithm is very suitable to deal with the mixed-model assembly line balancing problem with parallel workstations under the zoning constraints

REFERENCES

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

Anderson, E.J. & Ferris, M.C. (1994). Genetic Algorithms for Combinatorial Optimization: The Assembly Line Balancing Problem. *ORSA Journal on Computing*, 6, 161-173.

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

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

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.

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

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

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

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

Blum, C. & Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35 (3), 268-308

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, Y. & Rabinowitch, I. (2005) 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.

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

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

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

Coley, D. (2003). *An Introduction to Genetic Algorithms for Scientists and Engineers*. World Scientific Press, Singapore.

Eiben, A.E., Hinterding, R. & Michalewicz, Z. (1999). Parameter Control in Evolutionary Algorithms. *IEEE Transactions On Evolutionary Computation*, 3, 124-141.

El-Abd, M. & Kamel, M. (2005). A Taxonomy of Cooperative Search Algorithms. *Lecture Notes in Computer Science*, 3636, 32-41.

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

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.

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

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

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

Glover, F. (1986). Future paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, 5:533-549.

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

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). *GAs in search, optimization and machine learning*. Reading, Massachusetts: Addison-Wesley.

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

Haupt, R.L. & Haupt, S.E. (2004). *Practical Genetic Algorithms*. Second edition, Wiley, New York, NY.

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

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

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

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

Kara, Y., Ozcan, U. & Peker, A. (2007) An approach for balancing and sequencing mixed- model JIT U-lines. *The International Journal of Advanced Manufacturing Technology*, 32(11-12), 1218-1231.

Karabatı, S. & Sayın, S. (2003). Assembly line balancing in a mixed-model sequencing environment with synchronous transfers. *Europen Journal of Operational Resarch*, 149, 417-429.

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

Khoo, L.P. & Alisantoso, D. (2003) Line balancing of PCB assembly line using immune algorithms. *Engineering with Computers*, 19, 92-100.

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

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

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

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

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, C.M. & Chen C.H. (2002). Multi-section electronic assembly line balancing problems: a case study. *Production Planning and Control*, 13(5), 451-461.

Matanachai, S. & Yano, C.A. (2001) Balancing mixed-model assembly lines to reduce work overload. *IIE Transactions*, 33, 29-42.

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

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

McMullen, P.R., & Tarasewich, P. (2003). Using Ant Techniques to Solve the Assembly Line Balancing Problem. *IIE Transactions: Design and Manufacturing,* 35(7), 605-617.

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.

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

Ozcan, U. & Toklu, B. (2009). Balancing of mixed-model two-sided assembly lines. *Computers & Industrial Engineering*, in press.

Pham, D.T. & Karaboga, D. (2000). *Intelligent Optimisation Techniques*. Great Britain: Springer-Verlag London Limited

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

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

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

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

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

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

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

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

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

Su, P. & Lu, Y. (2007). Combining Genetic Algorithm and Simulation for the Mixed-Model Assembly Line Balancing Problem. *Proceedings of the Third International Conference on Natural Computation*, Haikou, China, August, 314-318.

Suwannarongsri, S., Limnararat, S. & Puangdownreong D. (2007). A New Hybrid Intelligent Method for Assembly Line Balancing. *Proceedings of the 2007 IEEE IEEM,* Singapore, 115-119.

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

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

Venkatesh, J.V.L. & Dabade, B.M. (2008). Evaluation of performance measures for representing operational objectives of a mixed model assembly line balancing problem. *International Journal of Production Research*, 46(22), 6367-6388.

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

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

Zhao, X., Ohno, K. & Lau, H.S. (2004). A balancing problem for mixed-model assembly lines with a paced moving conveyor. *Naval Research Logistics*, 51, 446-64.