Universidade de Aveiro
2006
Departamento de Economia, Gestão e Engenharia Industrial

Ana Sofia de Almeida Simaria

Balanceamento de linhas de montagem - novas perspectivas e procedimentos

Assembly line balancing - new perspectives and procedures

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Gestão Industrial, realizada sob a orientação científica do Professor Doutor Pedro Manuel Moreira da Rocha Vilarinho, Professor Auxiliar do Departamento de Economia, Gestão e Engenharia Industrial da Universidade de Aveiro.

Apoio financeiro da FCT e do FSE no âmbito do III Quadro Comunitário de Apoio (PRAXIS XXI / BD / 19554 / 99).

o júri

presidente

Doutor José Carlos da Silva Neves
professor catedrático da Universidade de Aveiro

Doutor Joaquim José Borges Gouveia
professor catedrático da Universidade de Aveiro

Doutor José Manuel Vasconcelos Valério de Carvalho
professor catedrático da Escola de Engenharia da Universidade do Minho

Doutor José Fernando da Costa Oliveira
professor associado da Faculdade de Engenharia da Universidade do Porto

Doutor Rui Manuel Moura de Carvalho Oliveira
professor associado do Instituto Superior Técnico da Universidade Técnica de Lisboa

Doutor Pedro Manuel Moreira da Rocha Vilarinho
professor auxiliar da Universidade de Aveiro (orientador)

agradecimentos

Gostaria de expressar o meu reconhecimento:

Ao meu orientador, Professor Doutor Pedro Manuel Moreira da Rocha Vilarinho, pela dedicação e empenho que colocou no acompanhamento desta dissertação bem como pela forte motivação dada ao longo da sua realização.

Ao Departamento de Economia, Gestão e Engenharia Industrial da Universidade de Aveiro, na pessoa do presidente do conselho directivo Professor Doutor Joaquim José Borges Gouveia, pela disponibilização das condições necessárias à realização deste trabalho.

À Fundação para a Ciência e a Tecnologia pelo apoio financeiro.

Às minhas colegas e aos meus colegas do DEGEI pelo apoio e amizade.

À minha Família, pelo infinito amor, apoio e incentivo.

Dedico este trabalho à Ana João.

palavras-chave

Balanceamento de linhas de montagem, optimização combinatória, meta-heurísticas.

resumo

No presente trabalho é apresentado um conjunto de procedimentos para o balanceamento de linhas de montagem de modelo-misto. Linhas de modelo-misto eficientes representam um factor chave de competitividade no actual ambiente de mercado, em que a crescente procura de produtos personalizados requer uma resposta flexível dos sistemas de produção.
Os procedimentos propostos, baseados nas meta-heurísticas '*simulated annealing*', 'algoritmos genéticos' e 'optimização por colónias de formigas', são capazes de abordar algumas características do processo de montagem presentes nas linhas reais (e.g., utilização de postos paralelos, restrições de zona, linhas de dois lados, linhas em forma de U) que a maioria das técnicas existentes na literatura não considera. Isto constitui uma contribuição relevante quer para o conhecimento científico quer para o conhecimento industrial na área do balanceamento de linhas de montagem.
Alguns dos procedimentos foram utilizados no balanceamento de linhas de montagem reais com o objectivo de testar a sua flexibilidade de adaptação às condições de operação em ambientes industriais.

keywords Assembly line balancing, combinatorial optimisation, meta-heuristics.

abstract

In this work a set of procedures to efficiently balance mixed-model assembly lines is proposed. Efficient mixed-model lines represent a key factor of competitiveness in the actual market environment, in which the growing demand for customised products increases the pressure for manufacturing flexibility.

The proposed procedures, based on the meta heuristics ‘simulated annealing’, ‘genetic algorithms’ and ‘ant colony optimisation’, are able to address some particular features of the assembly process very common in real mixed model assembly lines (e.g., use of parallel workstations, zoning constraints, task side constraints, U-shaped layouts) that most of the techniques existing in the literature do not consider. This is a major contribution to the scientific and industrial knowledge on the line balancing subject.

Some of the procedures were applied to real assembly lines in order to test their flexibility to cope with real industrial settings, as they may differ significantly from theoretical problems.

Index

Index of Figures

Index of Tables

1

Introduction

Contents

- **Relevance of the problem**
- **Objective of the thesis**
- **Structure of the thesis**

1.1 Relevance of the problem

The dynamics and intense competition in the current global marketplace together with the increased pace of technological change has led to shortening product life cycles and to a proliferation of product variety. Companies must be able to provide a higher degree of product customisation to fulfil the needs of the increasingly sophisticated customer demand (Su et al, 2005). Moreover, responsiveness in terms of short and reliable delivery lead times is demanded by a market where time is seen as a key driver. Mass customisation is a response to this phenomenon. It refers to the design, production, marketing and delivery of customised products on a mass basis. This means that customers can select, order and receive especially configured products, often selecting from a variety of product options, to meet their individual needs. On the other hand, customers are not willing to pay high premiums for these customised products compared to competing standard products in the market. They want both flexibility and productivity from their suppliers (Rudberg and Wikner, 2004).

As forecasting and planning become very complex, producing and storing all types of finished goods based on forecasts will lead to a high risk of stock out and obsolescence, while lead time often makes build-to-order impossible (Yang and Burns, 2003). Postponement arises as a strategy to contribute to the achievement of mass customisation. The concept of postponement is about delaying activities in the supply chain until real information about the market is available. The underlying logic is that the delay leads to the availability of more information and thus the risk and uncertainty of those activities can be reduced or even eliminated. In a postponement strategy uncertainty is seen as an opportunity instead of a problem (Yang et al, 2004, 2005).

Manufacturing postponement or delayed product differentiation is a type of postponement that seeks to delay the final formulation of a product until customer orders are received (Skipworth and Harrison, 2004). For example, in the automotive industry (high-volume vehicles), customers are allowed to choose their vehicle from a wide set of options. Customer involvement takes place only in the final assembly stage (Coronado et al, 2004).

Delayed product differentiation involves shipping the products in a semi-finished state from the manufacturing facility to a downstream facility where final customisation occurs, normally as an assembly process. This strategy allows companies to standardise components and create a variety of products. Here, modularity plays an important role for a good performance of the system. It is an approach for efficiently organise complex products and processes by decomposing complex tasks into simpler portions so that they can be managed independently and yet operate together as a whole (Mikkola and Skjott-Larsen, 2004). Modularity consists in the breakdown of a complex part into simple and functionally independent components which are assembled to make customised parts. Although the number of parts in the modular design is larger than in the integral design, the total time of machining operations and manufacturing costs are more likely to decrease in the modular design. Nevertheless, modular designs increase the number of assembly operations and the assembly time and, hence, may require additional assembly stations in the system (He and Babayan, 2002).

Delayed product differentiation benefits the manufacturing process in two ways: it increases flexibility by enabling to commit the work-in-process to a particular end-product at a later time, and it decreases costs of complexity by reducing the variety of components and processes within the system (Nair, 2005).

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. As manufacturing is shifting from high-volume/low-mix production to high-mix/low-volume production, mixed-model assembly lines, in which a set of similar models of a product can be assembled simultaneously, are better suited to respond to the new market demands.

Instead of an inflexible production system, like they have been before, assembly lines are now an important piece of the supply chain, essential to support manufacturing postponement strategies. On one hand, assembly lines have the ideal structure to perform final product customisation tasks under a mass customisation concept. On the other hand,

as they are labour intensive, assembly lines can be easily located geographically closer to the final customer marketplace.

The efficient design and operation of mixed-model assembly lines is, therefore, a crucial factor for the success of the supply chain in delivering customised products at low costs.

1.2 Objective of the thesis

The main objective of this thesis is to present a set of procedures to efficiently tackle different types of mixed-model assembly line balancing problems.

The proposed procedures based on the meta-heuristics, such as *simulated annealing*, *genetic algorithms* and *ant colony optimisation algorithms*, are able to address some particular features of the assembly process very common in real mixed-model assembly lines (e.g., use of parallel workstations, zoning constraints, task side constraints, U-shaped layouts) that most of the techniques covered in the current literature do not consider. This is a major contribution to scientific and industrial knowledge on the assembly line balancing subject.

Some of the procedures were applied to real assembly lines in order to test their efficiency to cope with real industrial settings, as they may differ significantly from theoretical problems. So, another goal of this thesis is to share the experience (successful applications and difficulties) of dealing with the conditions of real production systems.

1.3 Structure of the thesis

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

The second chapter gives an overview of the assembly line balancing problem. It presents the main characteristics of assembly line systems and defines the assembly line balancing problem, emphasising the mixed-model perspective. Different types of assembly line configurations and particular features of the assembly process that may restrict the configuration of the lines are also presented.

The third chapter is dedicated to review the available literature reporting meta-heuristic based approaches to tackle the assembly line balancing problem. It firstly describes the main characteristics of the selected meta-heuristics (simulated annealing, genetic algorithms and ant colony optimisation) and then presents a literature review of their applications to line balancing problems.

The fourth chapter presents the models and algorithms developed in this work for balancing mixed-model assembly lines with a linear configuration. A mathematical programming model was built to formally describe the problem and three heuristic procedures were developed to solve the problems. The procedures are based on well-known meta-heuristics, such as simulated annealing, genetic algorithms and ant colony optimization. A comparison between the performances of the three procedures, based on a set of computational experiments, is also provided.

In the fifth and sixth chapters mathematical programming models and heuristic procedures for balancing U-shaped assembly lines and 2-sided assembly lines, respectively, are presented. Conclusions about the heuristics' performance are withdrawn, based on a set of computational experiments.

In the seventh chapter four industrial case studies are presented. They resulted from the analysis of real assembly lines and consequent application of the proposed heuristic procedures to improve the lines' efficiency.

Finally, conclusions and directions for future research are pointed out in the eighth chapter.

2

Overview of the mixed-model assembly line balancing problem

Contents

- **Chapter introduction**
- **Main characteristics of assembly line systems**
- **The mixed-model assembly line balancing problem**
- **Particular features of the assembly process**
- **Assembly line performance measures**

2.1 Chapter introduction

This chapter aims to provide an overview of the main features of production systems organised as assembly lines and to introduce the main concepts required to understand the mixed-model assembly line balancing problem – the object of the research presented in this work.

The chapter begins by introducing the main characteristics of assembly line systems, in order to point out the importance of mixed-model production, and the general mixed-model assembly line balancing problem is briefly described. Then, some particular features of the assembly line process that may be present in real assembly lines are described and the most common line performance measures are presented.

2.2 Main characteristics of assembly line systems

An **assembly line** is a set of sequential **workstations** connected by a material handling system, usually a conveyor belt. Manufacturing a product in an assembly line requires partitioning the total amount of work into a set of elementary operations called **tasks**. In each workstation a set of tasks is performed using a predefined assembly process, in which the following issues are defined:

- the task **processing time**: the time required to perform each task;
- a set of **precedence constraints** that, due to technological or organisational conditions, determine the sequence in which the tasks can be performed.

Figure 2.1 shows an example of a precedence diagram, in which the nodes represent tasks and the arcs express the precedence relationships between the tasks. For example, task 12 can only be performed after tasks 8 and 9 are completed (tasks 8 and 9 are direct **predecessors** of task 12).

In a typical workstation the work is performed manually by human operators using simple tools or by semi-automated machines controlled by those operators. The time required to perform all tasks assigned to a workstation is termed **workload**.

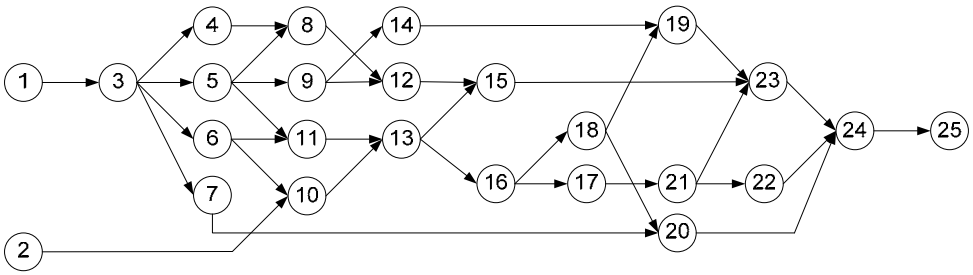

Figure 2.1 – Example of a precedence diagram

In a **paced** assembly line each workstation has a predefined amount of time to complete all the tasks assigned to it: the **cycle time**. When this time is elapsed the sub-assembly must be moved to the next workstation and the workstation receives a new sub-assembly from the previous workstation. Thus, the cycle time determines the production rate of the assembly line. Since tasks are indivisible work elements, the cycle time cannot be less than the maximum task processing time (for assembly lines with no parallel workstations, as it will be explained in section 2.4.4). The difference between the cycle time and the workload is called **workstation idle time**. The sum of the idle times of all the workstations in the assembly line is the line idle time or **total idle time**.

In **unpaced** assembly lines there is no fixed time for a workstation to complete its tasks. All workstations operate at an individual speed so that sub-assemblies may have to wait before they can enter the next workstation and/or workstations may get idle waiting to receive a sub-assembly from the previous workstation. To avoid these difficulties, buffers between workstations are normally introduced in order to keep in-process inventories. The work developed in the present study only addresses paced assembly lines.

Considering the number of products to be assembled and the way they are processed, there are, basically, three types of assembly lines:

- **single-model** assembly lines, in which a single homogenous product is continuously assembled in large quantities;
- **mixed-model** assembly lines, in which a set of similar models of a product can be assembled simultaneously, in an arbitrarily intermixed sequence;
- **multi-model** assembly lines, in which batches of similar models are assembled with intermediate setup operations.

Figure 2.2 illustrates the different line types, where geometrical shapes symbolize the different models assembled on the line.

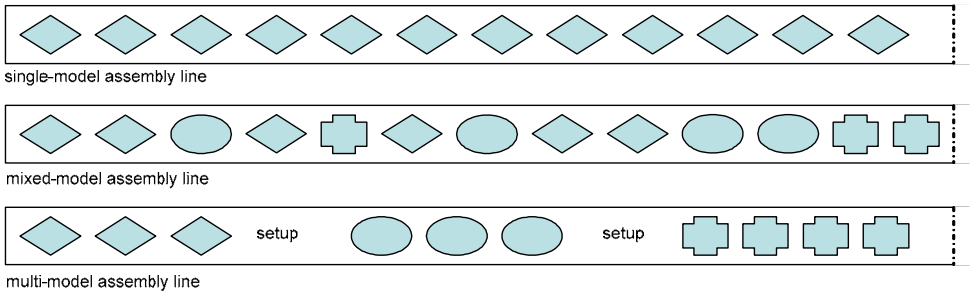

Figure 2.2 – Types of assembly lines

Single-model assembly lines are suitable for large-scale production, since they ensure very low production costs. High productivity is achieved by manufacturing a single product in very large quantities, using the principles of specialisation and division of work among operators. But long gone are the days when everyone could purchase a low priced car of 'any colour as long as it was a black Model T Ford'.

The recent market trends show that there is a growing market demand for customised products, increasing the pressure for industries to diversify their production mix with more models and optional features being offered. Here it is evident the need for flexible systems, able to produce different versions of the same product without, however, increasing the costs excessively. This is the reason for companies to implement assembly line configurations, with specific measures being taken to make the system suitable for the production of different models. Assembly systems must still achieve high productivity, uniform quality and low assembly costs. Flexibility is also essential to cope with shorter product life cycles, low production volumes, changing demand patterns and a higher variety of product models and options.

In some cases multi-model lines are used: they can produce batches of different models with relatively low setup times. The line configuration is unique for each model so that tasks must be reassigned whenever the production changes from one model to another. When more flexibility is required the most suitable system is a mixed-model assembly line, in which setup is almost non-existent, allowing for the production of very small batches (even one-unit batches) in any sequence.

According to Zhao et al (2004), there are two basic issues to address in mixed-model assembly lines: (i) at the 'design' level, the assignment of tasks to workstations in order to optimise a given 'design measure' and (ii) at the 'operational' level, the determination of the sequence in which the difference models are launched into the line, in order to optimise a given 'operational performance measure'. The first is the **balancing** problem that must be addressed before building the line and the second is the **sequencing** problem that must be addressed everyday when implementing a production plan.

The present work addresses the balancing problem, which is defined in the following section.

2.3 The mixed-model assembly line balancing problem

The simple assembly line balancing problem (SALBP) was first mathematically formulated by Salveson (1955) and it consists in assigning a set of tasks, required to assemble a single homogenous product, to a set of workstations in order to minimise the number of workstations in the line or minimising the cycle time of the line (both these objectives are equivalent to minimise the idle time of the line). The assignment of tasks to workstations must ensure that the product demand is met and verify the following set of conditions (Shtub and Dar-El, 1990):

- a task is indivisible and therefore must be totally performed in a single workstation;
- the sequence of the assigned tasks must respect the technological precedence constraints;
- all workstations have conditions to perform any task;
- the task processing times are known and are independent of the workstation to which they are assigned;
- the sum of the processing times of the tasks assigned to each workstation cannot exceed the cycle time, determined by the product's demand.

The following characteristics are specific for the mixed-model assembly line balancing problem (MALBP):

- a set of similar models is simultaneously assembled on the line;
- each model has a predefined demand over a planning horizon;

- the cycle time of the line is given by the ratio between the planning horizon and the total demand of the different models;
- each model has its own set of precedence relationships, but it is possible to combine all the relationships into only one precedence diagram – the combined precedence diagram, as exemplified in Figure 2.3;
- the time required to perform a task may vary between the models;
- workstations are flexible enough to perform their tasks on the different models.

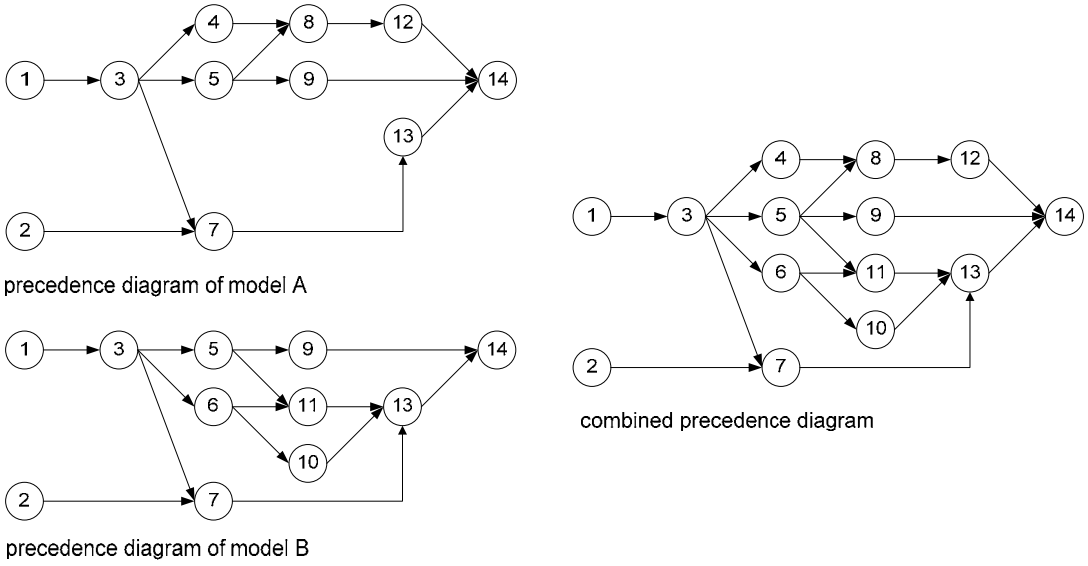

Figure 2.3 – An example of a combined precedence diagram

According to the pursued goal, the MALBP can be classified into two different types, which are referred as dual problems (Scholl, 1999):

- MALBP-I: minimises the number of workstations, for a given cycle time;
- MALBP-II: minimises the cycle time, for a given number of workstations.

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. Type II problems deal with the maximisation of the production rate of an existing assembly line and are applied when, for example, changes in the assembly process or in the product range require the line to be redesigned. Both types of problems have the same mathematical formulation. The only difference is in what is given as input and what is the decision variable. While for type I the cycle time is given

and the number of workstations is to be determined, for type II the opposite occurs, i.e., the number of workstations is given and the cycle time is to be determined.

The MALBP can be formulated as a binary integer programming model, as presented in Figure 2.4, in which:

- N is the number of tasks of the combined precedence diagram;
- M is the number of models assembled on the line;
- D_m is the demand of 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 line cycle time computed by $P / \sum_{m=1}^{M} D_m$;
- t_{im} is the processing time of task i for model m;
- Suc_i is the set of tasks that cannot be performed before task i is completed (successors of task i), derived from the combined precedence diagram;
- $x_{ik} = \begin{cases} 1, \text{if task } i \text{ is assigned to workstation } k \\ 0, \text{otherwise} \end{cases}$

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

subject to:

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

$$\sum_{k=1}^{S} k x_{ik} - \sum_{k=1}^{S} k x_{jk} \leq 0 \qquad i \in N,\ j \in Suc_i \qquad (3)$$

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

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

Figure 2.4 – Binary integer programming model for the MALBP

The objective function (1) minimises the weighted idle time of the assembly line, considering each model's production share. This goal is equivalent to minimise the number of workstations for a given cycle time in MALBP-I and to minimise the cycle time for a given number of workstations in MALBP-II. 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.

The binary integer programming model becomes very complex even for small size problems, which makes it impossible to be solved to optimality in acceptable time. The problem is $\mathcal{NP}$-hard (Scholl, 1999), which explains the interest of researchers in the development of heuristic procedures to address the problem.

Although the minimisation of the idle time is the main goal of the MALBP, additional goals, like the workload balance between and within workstations, are also important to obtain good balancing solutions. Later in this work these goals will be described in detail and included in the proposed approaches.

2.4 Particular features of the assembly process

In order to better reflect the operating conditions of real assembly lines, some relevant issues of the assembly process need to be included when addressing an assembly line balancing problem. Scholl (1999) and Becker and Scholl (2006) present a comprehensive explanation on some particular features of the assembly process. Here, only a briefly description of these aspects is provided.

2.4.1 Variability of task processing times

The variability of task processing times depends on the nature of the tasks and operators. While for simple tasks the expected variance is very small, the processing time

of complex and failure sensitive tasks may have significant variability, especially if performed by human operators, influenced by physical, psychological and social factors.

The use of **deterministic** values for the task processing times is justified when the expected variance is low. In most assembly lines using human workforce the number of tasks assigned to each operator is small and each task is usually very simple. Also, operators are especially trained to perform efficiently that small set of tasks. This way, the variability inherent to the human nature of the work is reduced by the simplicity of tasks and qualification of operators. The increased automation is also able to reduce the variability of task processing times, by using computer-controlled machines and robots able to work at constant speed.

If the tasks performed by human operators are long or complex, the variability of the task processing times should be considered when modelling the problem because the variance may significantly affect the system's performance. In the case of automated lines, in which processing times are almost constant, there is a need to deal with the occurrence of machine breakdowns, by incorporating in the model a **stochastic** component of the task times reflecting the probability of machine breakdowns and the duration of repair processes.

When installing a new assembly line or introducing a new product in the line, the operators may have an adjustment period in which they take longer time to perform the tasks than after they are fully adapted. **Dynamic** task processing times may be used when learning effects allow systematic reductions or successive improvements of the production process.

2.4.2 Assignment constraints

Assignment constraints reduce the set of workstations to which tasks can be assigned. Several types of assignment constraints can be included in an assembly line balancing problem.

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. For example, if two tasks need the same equipment or have similar processing conditions

(temperature, pressure, operator qualification level, etc.) it is desirable that they are assigned to the same workstation. Negative zoning constraints are usually imposed by technological issues like, for example, when it is necessary to have a minimum time between the execution of the tasks or when it is not possible to perform them in the same workstation, for safety reasons.

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 (like cars, washing machines, etc.) 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 p**osition 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.

2.4.3 Layout

In traditional or **straight** assembly lines, workstations are physically arranged along a linear conveyor belt and operators perform tasks on a continuous portion of the line. The implementation of *just-in-time* principles in industrial facilities made companies to switch from straight to **U-shaped** assembly lines. In a U-shaped line both ends of the line are closely together forming a ‘U’ and operators can move between the two legs of the line to perform combinations of tasks that would not be allowed in a straight line. It is an attractive alternative for assembly systems since operators become multi-skilled by executing tasks located at different parts of the assembly line. It improves visibility and communication between operators, which may facilitate problem solving. Also, a U-shaped line configuration allows for more possibilities on the assignment of tasks to

workstations, so the number of workstations may be reduced, when compared with the number of workstations needed for a straight line. Figure 2.5 illustrates the differences between the assignment of tasks in straight and U-shaped assembly lines. A more detailed description of U-shaped assembly lines is provided in chapter 5.

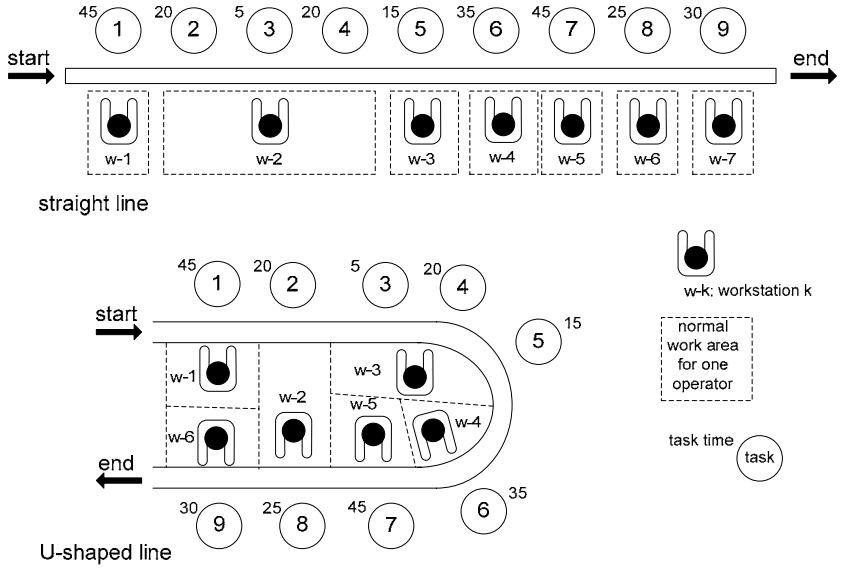

Figure 2.5 – Assignment of tasks in straight and U-shaped assembly lines

Other assembly line layouts may be found in industrial facilities, like the C-shaped layout, illustrated in Figure 2.6 (Aase et al, 2004).

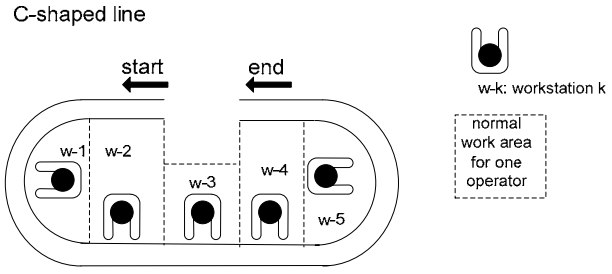

Figure 2.6 – Configuration of a C-shaped assembly line

2.4.4 Parallelism

The implementation of **parallel lines** to assemble one or several products allows increasing flexibility and decreasing failure sensitivity of the production system. Parallel lines facilitate quick responses to product demand variations as the number of working lines can be easily changed. Also, the risk of production stoppage due to machine breakdowns is significantly reduced. Moreover, the cycle time can be increased, which brings additional advantages such as (i) better line balances, due to the higher number of possible task combinations and (ii) job enrichment, as the operators perform a larger number of different tasks.

The strategic problem of determining the optimal number of parallel line is of major importance as the duplication of lines involves increasing capital investment. However, when parallel lines are introduced, the number of tasks performed by each worker increases, the limit being one worker at each line performing all the tasks of the assembly process. This contradicts one of the main advantages of using assembly lines: the use of low skilled labour that can be easily trained (due to the strict division of labour). So, this aspect must be considered when installing parallel lines.

Even in single lines, parallelism can be implemented. When the production rate required to meet the demand is so high that the processing times of some of the tasks exceed cycle time, the implementation of **parallel workstations** is necessary to achieve the desired production rate. In parallel workstations, different workpieces are distributed among several operators who perform the same tasks. The local cycle time in these workstations is a multiple of the global cycle time, depending on the number of replicas installed. An example of the use of parallel workstations is shown in Figure 2.7. The longest task processing time of this example is 45, which limits the cycle time in the first configuration where no parallel workstations are used. With the use of parallel workstations it is possible to decrease the cycle time, for the same number of operators (seven), as it is shown in the second line configuration.

The use of parallel workstations is a common practice that allows a more flexible assignment of tasks and a reduction of the line cycle time. However, as for parallel lines, if the replication of workstations is not controlled, the advantage of the strict division of labour inherent to assembly lines can be lost.

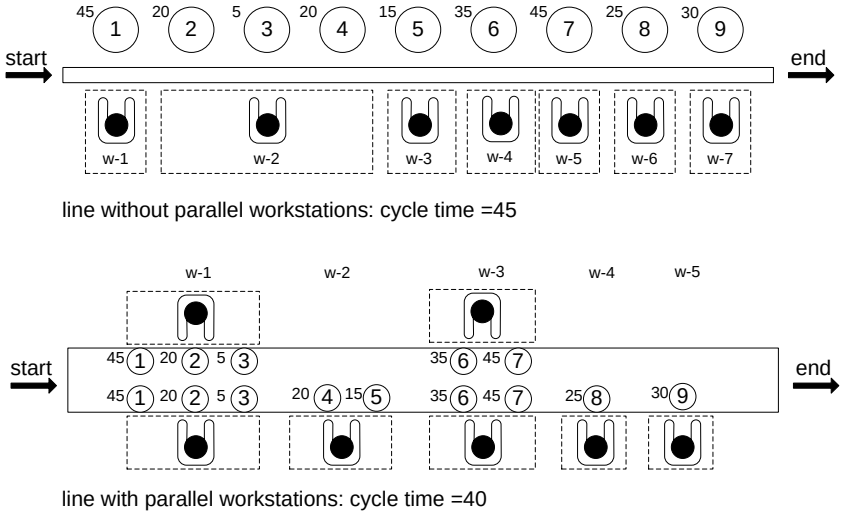

Figure 2.7 – Illustration of the use of parallel workstations

2.5 Assembly line performance measures

The implementation of an assembly line requires high capital investments, so it is very important that the line is designed and balanced to work as efficiently as possible. Also, re-balancing an existing assembly line is necessary when changes in the production process or demand structure occur. To assess the performance of the line, several criteria of technical and economical nature can be included in assembly line balancing problems.

According to Gosh and Gagnon (1989) the most widely used criteria of technical nature are related with the maximisation of the capacity utilisation which is measured by the line efficiency (the percentage of productive time in the line). Among them are (i) the minimisation of the number of workstations, for a given cycle time, (ii) the minimisation of cycle time, for a given number of workstations and (iii) the minimisation of the idle time of the line. Other capacity related criterion is the smoothing of workloads between the workstations, important to ensure similar workloads for all operators (see, for example, Merengo et al, 1999, Matanachai and Yano, 2001, Vilarinho and Simaria, 2002).

The economical nature criteria seek to minimise 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. As stated by Scholl (1999), the most important cost categories are (i) costs of machinery and tools, (ii) labour costs, (iii) materials costs, (iv) idle time costs, (v) penalties for not meeting the demand, (vi) incompletion costs, (vii) setup costs and (viii) inventory costs.

Several authors present multi-objective approaches. Shtub and Dar-El (1990) consider simultaneously (i) the minimisation of the line idle time and (ii) the minimisation of the number of parts at each workstation.

Malakooti (1991) includes (i) the number of workstations, (ii) the cycle time and (iii) the line operation costs, in his multi-criteria approach. In Malakooti (1994) the previous work is extended to include the size of buffers in the assembly line as another goal.

McMullen and Frazier (1998) use multi-objective criteria that comprise (i) the cost of labour and equipment, (ii) the workload balance between workstations and (iii) the probability of lateness.

Ponnambalam et al (2000) consider (i) the number of workstations, (ii) the workload balance between workstations and (iii) the assembly line efficiency as criteria to evaluate line balancing solutions.

Zhao et al (2004) aim to minimise the operational performance measure 'total overload time', i.e., the amount of time that exceeds the cycle time of the line, when considering mixed-model production. These authors state that the total overload appropriately reflects the relevant additional operating cost of the line, as when overload occurs the unfinished work has to be completed offline or the conveyor must be temporarily stopped to finish the tasks.

Besides capacity and cost related objectives, social goals may be important to fulfil, such as (i) job enrichment, avoiding the assignment of many monotonous tasks to an operator and (ii) job enlargement, increasing the number of tasks performed by an operator.

Although a wide variety of objectives may be included in line balancing approaches, the fact is that most of the objectives described in this section are basically influenced by the number of workstations and the cycle time of the line. Thus, this two goals can be considered the most important when balancing an assembly line.

3

Meta-heuristics for assembly line balancing: characterisation and review of existing procedures

Contents

3.1 Chapter introduction

The assembly line balancing problem was firstly formulated by Salveson (1955) and, since then, numerous procedures have been developed to solve the problem. The literature on the subject is extensive and it focuses mainly on the simple version of the assembly line balancing problem. Comprehensive literature reviews on both exact and heuristic solution techniques for the different types of assembly line balancing problems are presented by Gosh and Gagnon (1989), Erel and Sarin (1998), Scholl (1999) and more recently by Becker and Scholl (2006) and Scholl and Becker (2006).

Although many optimising methods have been proposed, mainly branch-and-bound and dynamic programming procedures, their application is only possible for very restricted versions of the assembly line balancing problem, as the problem is $\mathcal{NP}$-hard. To better reflect the characteristics of real world assembly lines, additional constraints must be included when solving the problem and this only increases its complexity. So, instead of exact procedures that find optimal solutions for simplified problems, heuristic procedures are used to find good solutions for much more complex problems. A large variety of heuristic approaches have been proposed in the literature. According to Scholl and Becker (2006), the development of constructive procedures, based on priority rules, to build one or more feasible solutions was presented in the literature until the mid nineties. In the last decade, the focus of researchers has been on improvement procedures using meta-heuristics like simulated annealing (Kirkpatrick et al, 1983), genetic algorithms (Holland, 1975, Goldberg, 1989), taboo search (Glover, 1989, 1990), and more recently, ant colony optimisation algorithms (Dorigo et al, 1996).

Meta-heuristics are general search principles organised in a general search strategy used to solve combinatorial optimisation problems (Pirlot, 1996). They are able to search large regions of the solution's space without being trapped in local optima, a major disadvantage of pure local search algorithms. As the research carried out for this work involves the application of meta-heuristics to mixed-model assembly line balancing problems, this chapter will focus on (i) the description of the main characteristics of the selected meta-heuristics (simulated annealing, genetic algorithms and ant colony optimisation

algorithms) and on (ii) the literature review of their application to assembly line balancing problems.

3.2 Simulated annealing algorithms

3.2.1 Overview

Simulated annealing (SA) is a randomised search technique that draws its inspiration from the physical annealing of solids. In this process, a solid is brought to its lowest energy state by first heating it to a very high temperature (usually the melting point temperature) and then cooling it at a very slow rate, to a very low temperature. When this heating and subsequent slow cooling occur, the particles within the solid rearrange themselves in such a way that the solid acquires some desired attribute, such as high strength or surface hardness.

The SA algorithm was introduced by Kirkpatrick et al (1983) to solve $\mathcal{NP}$-hard combinatorial optimisation problems, by using the analogy with the simulation of the physical annealing of solids, in order to optimise the value of an objective function. Figure 3.1 presents the structure of a general SA algorithm.

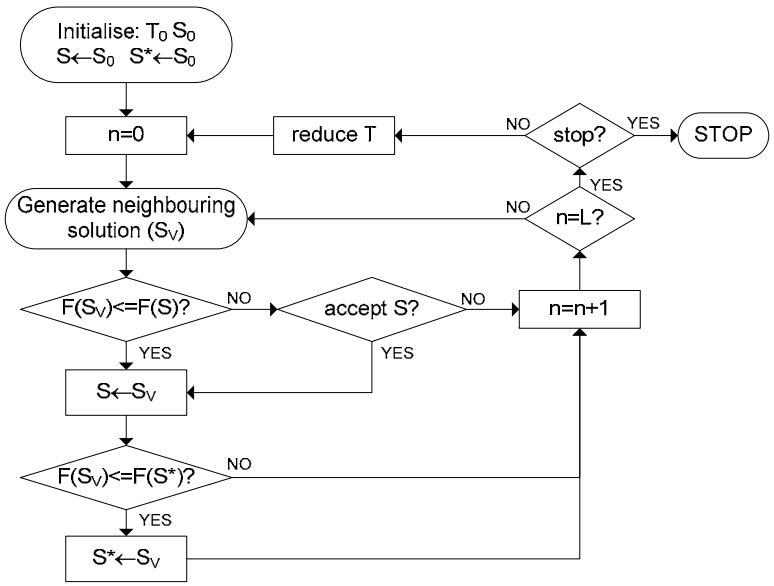

Figure 3.1 – Structure of a simulated annealing algorithm

It starts from an initial solution to the problem, S_0 and a control parameter, T, which is set to an initial temperature value, T_0. During the algorithm, the value of T is systematically decreased according to an annealing schedule as shown in Figure 3.2. In this schedule the following issues are defined: (i) a temperature reduction function and (ii) the length of each temperature level, L, that determines the number of solutions generated at a certain temperature.

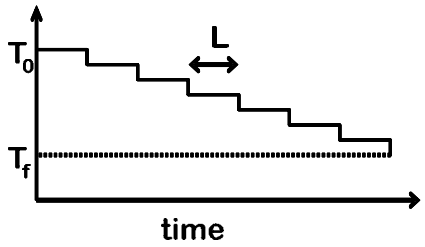

Figure 3.2 – Annealing schedule

At each temperature level, and as the temperature decreases, neighbouring solutions of the current solution are generated. A neighbouring solution, S_V, is accepted, i.e., replaces the current solution, if it is not worse than the current solution, S, ($F(S_V) \leq F(S)$, where F is the general objective function to minimise). If the neighbouring solution is worse than the current solution ($F(S_V) > F(S)$), it still may be accepted with a certain probability, $p=e^{-\Delta/T}$ where

$$\Delta = \frac{F(S_V) - F(S)}{F(S_V)} \times 100 \qquad \textbf{(3.1)}$$

This probability of accepting inferior solutions allows the simulated annealing algorithm to escape from local minima.

S^* is the best solution found by the algorithm.

The performance of the algorithm depends on the definition of the following annealing schedule parameters:

(i) The initial temperature, T_0, should be high enough so that in the first iteration of the algorithm the probability of accepting worst solutions is, at least, 80% (Kirkpatrick et al, 1983).

(ii) The most commonly used temperature reduction function is geometric: $T_i=a_iT_{i-1}$ ($a_i<1$ and constant). Typically, $0.8\leq a_i \leq 0.99$ (Eglese, 1990).

(iii) The length of each temperature level, L, determines the number of solutions generated at each temperature, T, and its value usually depends of the dimension of the problem.

(iv) The stopping criterion defines when the system has attained a desired energy level. Some of the most common stopping criteria are based on:

- the total number of solutions generated;
- the temperature at which the desired energy level is attained (freezing temperature);
- the acceptance ratio (the ratio between the number of solutions accepted and the number of solutions generated).

Naturally, each of these control parameters must be refined according to the specific problem on hand. Two other important issues that need to be defined when adapting this general algorithm to a specific problem are the procedures to generate both the initial solution and the neighbouring solutions. These aspects will be addressed in the following section, in which a review of the application of SA procedures to the assembly line balancing problem is provided.

3.2.2 SA approaches for assembly line balancing

Heinrici (1994) proposes a SA procedure to solve the single-model assembly line balancing problem of type II, in which the objective is to minimise the cycle time for a given number of workstations. Suresh and Sahu (1994) solve the problem of type I and address variability by using stochastic task processing times. The SA approach presented by Erel et al (2001) aims at balancing U-shaped assembly lines. McMullen and Frazier (1998) present a multi-objective procedure to balance mixed-model assembly lines with stochastic task processing times and parallel workstations.

In the following sections a brief description of the application of simulated annealing to the assembly line balancing problem is provided, namely (i) the way the initial solution is obtained, (ii) the procedures to generate neighbouring solutions and (iii) the objective function used to evaluate the solutions and guide the search.

3.2.2.1 Initial solution

The precedence constraints of an assembly process determine the set of tasks available for assignment at a particular moment. The initial solution of a SA based procedure is typically obtained by a constructive heuristic, in which, from the set of available tasks, one task is selected according to a certain rule and assigned to the current workstation, as long as it does not exceed the workstation's capacity. In the approach of Suresh and Sahu (1994) tasks are assigned according to their numerical order to build an initial feasible solution. The Ranked Positional Weight technique, originally developed by Helgeson and Birnie (1961), is the basis of the assignment of tasks to workstations in the initial solution of the procedure of Heinrici (1994). The assignment of tasks to workstations in the initial solution is done arbitrarily in the approach presented by McMullen and Frazier (1998).

Erel et al (2001) propose a different way of building the initial solution. First, each task is assigned to a different workstation and then the number of workstations is reduced by combining two adjacent workstations. When the workload of the combined workstation exceeds cycle time (leading to unfeasibility), the initial solution is complete and the subsequent steps of the SA procedure are initialised.

3.2.2.2 Neighbouring solutions

All the SA procedures mentioned in the previous section generate neighbouring solutions using two different movements:

(i) swapping two tasks in different workstations;

(ii) transferring a task to another workstation.

The tasks and workstations are usually randomly selected and the resulting balancing solution must be feasible, regarding precedence and cycle time constraints.

3.2.2.3 Objective function

In the problem of type I the goal is to minimise the number of workstations for a given cycle time. But an objective function which only considers the number of workstations may not be effective, as there may exist several different balancing solutions with the same number of workstations. So, an important challenge is to determine an appropriate objective function that can efficiently guide the search through the solution space.

Depending on the nature of the problem or study, different objective functions are proposed to evaluate the balancing solutions and guide the SA procedure.

Dealing with stochastic task processing times, Suresh and Sahu (1994) use the probability of a workstation exceeding the cycle time and the balance of workloads between workstations (smoothness index) to compare their procedure with others available in the literature.

McMullen and Frazier (1998) in their multi-objective approach use the line design cost, the smoothness index and the probability of lateness to evaluate the solutions. They also build composite functions with combinations of these three objectives.

The SA procedure of Erel et al (2001) aims at achieving feasibility regarding cycle time constraints. The objective function used is the minimisation of the maximum station time, thus eliminating the unfeasibility caused by the workstation exceeding the cycle time.

Heinrici (1994) uses the minimisation of cycle time, as the addressed problem is of type II.

3.3 Genetic algorithms

3.3.1 Overview

Genetic algorithms (GA) are iterative search procedures, based on the biological process of natural selection and genetic inheritance, which maintain a population of a number of candidate members over many simulated generations. Hopefully the good characteristics of the members will be retained over the generations, maximising a determined fitness function.

GA do not operate directly on the solution space: solutions are coded in strings, over a finite alphabet, called chromosomes. An encoding is selected in a way that each solution in the search space is represented by one chromosome. Each chromosome is then decoded according to a user defined mapping function, enabling the computation of the corresponding fitness value, which reflects the quality of the solution represented by the chromosome. Figure 3.3 shows an example of representing a solution of the well known knapsack problem as a chromosome with binary codification. Each position in the

chromosome corresponds to an item, which takes the value 1 if it is selected and zero, otherwise.

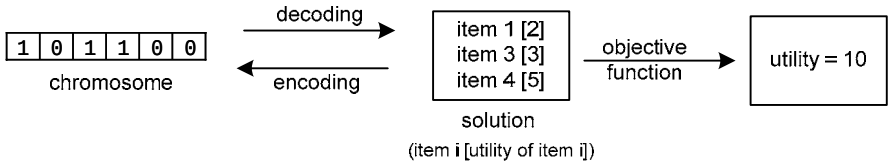

Figure 3.3 – Codification of a solution of a binary knapsack problem

The most fit individuals (chromosomes) are selected to form a basis for subsequent generations, i.e., for reproduction. However, the selection is not deterministic. Each individual has a probability of being selected for reproduction that increases with its fitness. The *selection* scheme should provide a balance between population diversity and selective pressure in order to avoid premature convergence, allowing for an effective search. A very popular selection technique is called *tournament* and it aims to imitate mutual competition of individuals during casual meetings. It works the following way: two individuals are randomly selected from the population and the worst one is placed at the top of an empty list. The best individual returns to the population and the process is repeated until all individuals have been placed on the list. Then, starting from the top of the list, chromosomes are selected to undergo genetic operators. Figure 3.4 illustrates the tournament selection strategy (adapted from Falkenauer, 1998).

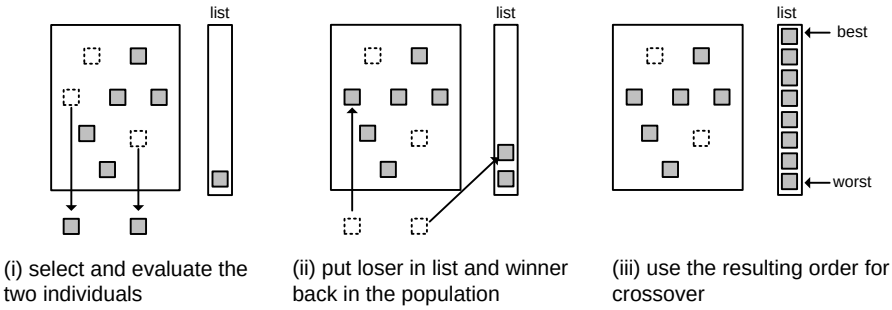

Figure 3.4 – Tournament selection

The main genetic operator is the *crossover*, which has the role of combining pieces of information from different individuals in the population. The selected individuals (parents) are joined in pairs and combine their genetic material to produce two new individuals (offspring) as it is shown in Figure 3.5. The main objective of crossover is to transmit good characteristics from parents to offspring.

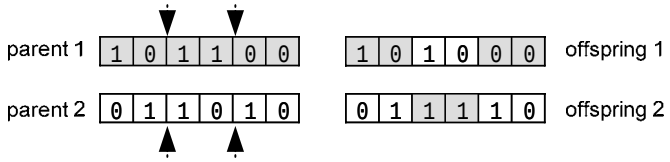

Figure 3.5 – A crossover example

Some individuals from the offspring population are randomly selected to undergo *mutation*, i.e., small random changes are made in their genetic information. For example, the mutation in a binary string is performed by changing the value of a randomly selected gene from 0 to 1 (or from 1 to 0). The use of mutation aims to ensure diversity among individuals, preventing premature convergence.

A *replacement* strategy is necessary to determine which individuals stay in the population and which are replaced by offspring. The members of the new generation can be (i) individuals from the current generation, (ii) offspring product of crossover or (iii) individuals who underwent mutation. The most common replacement approach is *elitism*, which allows the best chromosome in each generation to survive in the next generation, thus guaranteeing that the final population contains the best solution ever found. 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.

In general, the main steps of a GA procedure are:

1. Generation of a random initial population of solutions in the form of chromosomes.
2. Evaluation of each individual in the population according to a pre-defined fitness function.

3. Selection of a set of individuals to undergo genetic operators.
4. Evaluation of the individuals created by the genetic operators.
5. Application of a replacement strategy to form the new generation.
6. If a satisfactory solution is achieved (or the stopping criteria are met, usually, a pre-defined number of generations), stop, otherwise go to step 3.

Several studies point out the effectiveness of GA in solving combinatorial optimisation problems, since they work with sets of solutions instead of only one solution at the time. Also they are flexible enough to include problem specific characteristics in the encoding scheme. The following section provides details of the application of GA to assembly line balancing problems and gives a review of the more relevant published approaches.

3.3.2 GA approaches for assembly line balancing

Evolutionary approaches have been widely applied to solve problems related with the design and organisation of manufacturing systems. In this section, solely the application of GA to the assembly line balancing problem (ALBP) is described. For other manufacturing problems the interested reader is referred to the reviews provided by Dimopoulos and Zalzala (2000) and Pierreval et al (2003).

The main challenge of the application of GA to the assembly line balancing problem is the development of good encoding schemes and genetic operators in order to attain feasible solutions. In the first part of this section, a review of the existing codification procedures and genetic operators is provided. A difficulty found in the application of GA to the assembly line balancing problem is related with the fitness function (Scholl and Becker, 2006). When addressing the assembly line balancing problem of type I, the objective function to minimise is the number of workstations. However, in a population, there might be several different solutions with the same number of workstations, so, the sole use of this performance measure as the fitness function may not be effective to guide the search. A review of the fitness functions proposed in the literature for the ALBP is presented, in the second part of this section. Finally, a glance of other features of the application of GA to ALBP is given in the last part of the section.

3.3.2.1 Codification and genetic operators

The *standard encoding* scheme assigns directly the tasks to the workstations in a balancing solution. Each chromosome is a string of length *N* (number of tasks) where each element represents a task and the value of each element represents the workstation to which the corresponding task is assigned. Figure 3.6 presents an example of this standard encoding and the corresponding balancing solution.

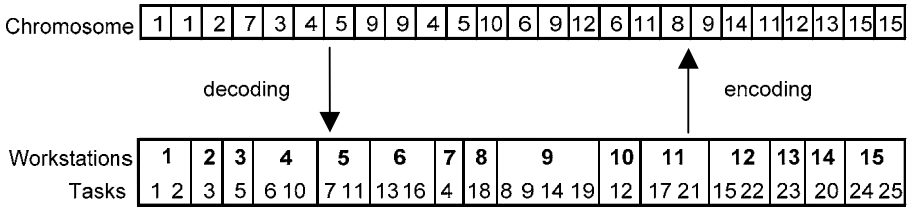

Figure 3.6 – Standard encoding and the corresponding balancing solution

Applying standard genetic operators, like crossover and mutation as described in the previous section, 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, 2000) 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, as it is shown in Figure 3.7, 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.

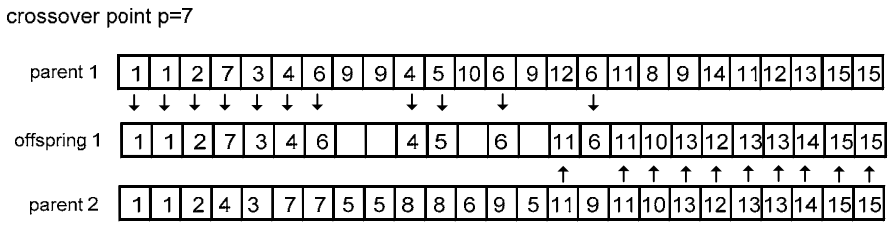

Figure 3.7 – Example of crossover specific for standard encoding

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.

A frequently used codification scheme is the *order encoding* where the chromosome is a sequence of tasks which verifies the precedence constraints. In order to obtain a balancing solution it is necessary to apply a construction procedure: the tasks are assigned to workstations in the sequence dictated by the chromosome. However, different chromosomes may lead to the same balancing solution, as the sequence of tasks within the workstations is not relevant for most balancing problems.

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. An illustration of this encoding is shown in Figure 3.8.

Rubinovitz and Levitin (1995) present a crossover operator called *fragment reordering crossover*, later used by Levitin et al (2006), which 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.

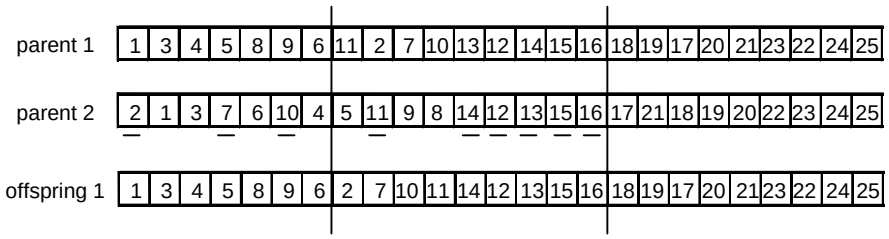

Figure 3.8 – Two-point order crossover

The use of the *partially mapped crossover* (Goldberg, 1989) is also reported in the applications of GA to ALB (Rubinovitz and Levitin, 1995 and Tsujimura et al, 1995) but the resulting task sequences are often unfeasible. This operator compares the two parents and performs task position exchanges such that each offspring is partially determined by each of its parents. Figure 3.9 gives an example of this operator.

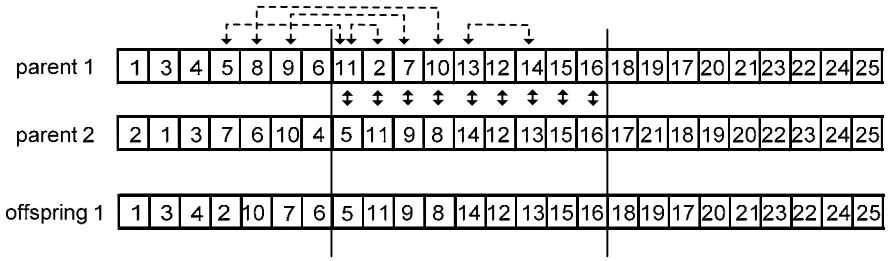

Figure 3.9 – Partially mapped crossover

Mutation operators perform mainly by (i) changing the position of two tasks in the chromosome (Rubinovitz and Levitin, 1995, Tsujimura et al, 1995 and 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).

Falkenauer (1998) presents a *grouping genetic algorithm*, especially suited for grouping problems, with a codification scheme called *group encoding*. In its application to ALBP, the groups are the workstations and the elements belonging to the groups are the tasks. The chromosome has two parts. The first part shows the assignment of tasks to workstations and it is similar to the standard encoding scheme. The second contains the groups, i.e., one gene for each workstation. Figure 3.10 shows an example of group encoding.

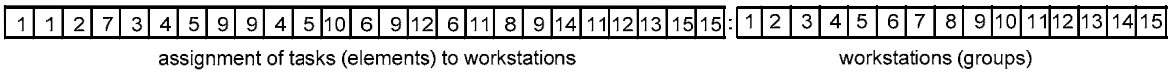

Figure 3.10 – Group encoding

The genetic operators are only applied to the group part of the chromosome. Rekiek et al (2000, 2001) use a crossover operator that performs in the following way: (i) selection of two crossover points, (ii) injection of the contents of the crossing section of the first parent at the first crossover point of the second parent, (iii) elimination of groups from the second parent containing duplicated elements and (iv) reinsertion of missing elements using problem specific heuristic rules. Figure 3.11 shows an illustration of this operator. For ease of demonstration, workstations are represented together with their tasks.

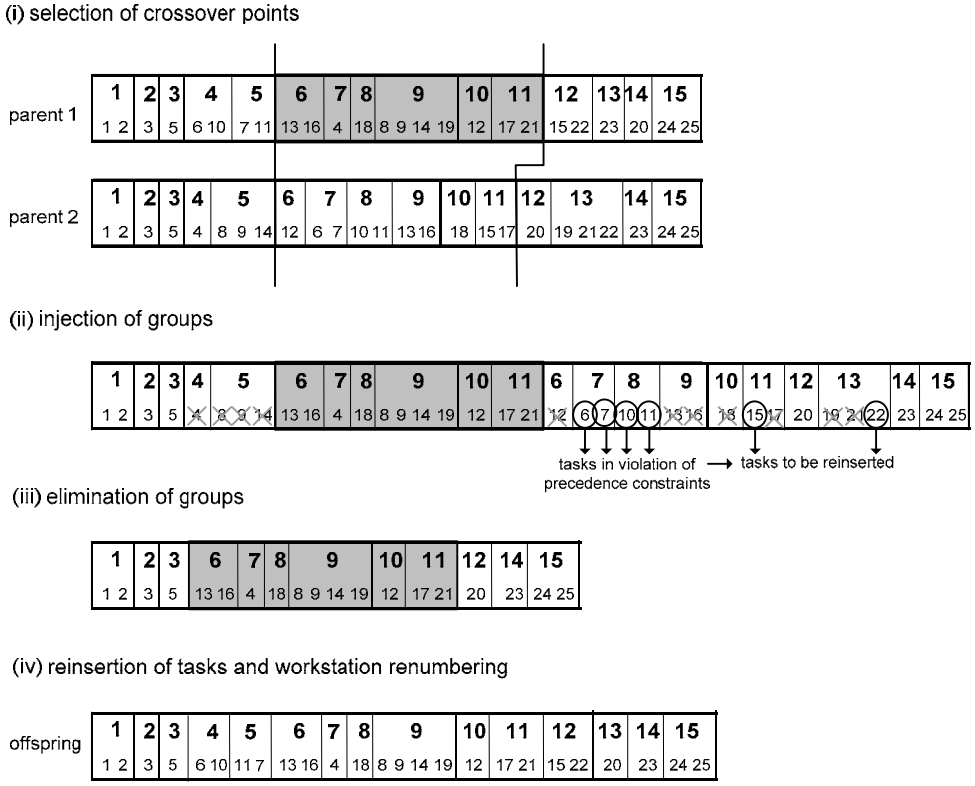

Figure 3.11 – Crossover in grouping genetic algorithms

Scholl and Becker (2006) use the term *indirect encoding* to designate other ways of encoding found in the ALBP literature. Gonçalves and Almeida (2002) and Ponnambalam

et al (2000) use chromosomes to represent a priority rule for each task. Each chromosome generates a balancing solution by applying a constructive priority-based heuristic. The first authors also apply a local search procedure in order to improve the solution and use a crossover operator called *uniform crossover* in which, gene by gene, there is a random selection of which of the two parents will provide the information.

The codification of solutions proposed by Zhao and de Souza (2000), for balancing an automated production line, is a matrix with the values of several adjustable variables of the problem (machine settings, facilities downtime, manpower assignment, batch size, etc.). For balancing a printed circuit board assembly line, Ji et al (2001) also use a matrix format chromosome in which each element x_{ij} represents the number of components of type j to be assembled on machine i. Lee et al (2000) use genetic algorithms as an input for a simulation model to balance a semi-automated assembly line. The chromosomes represent the processing times of the different workstations of the line.

3.3.2.2 Fitness function

Several fitness functions have been proposed in the literature for the ALBP. For problems of type I, as it was mentioned before, 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 better guidance of the search process.

Kim et al (2000) use the minimisation of the *adjusted number of workstations*, which favours solutions that can, more likely, be improved. This function is computed by adding to the number of workstations (S) the ratio between the workload of the last workstation (W_S) and the cycle time (C), as shown in the following expression:

$$Fitness = S + \frac{W_S}{C} \qquad \textbf{(3.2)}$$

Falkenauer (1998) evaluates the *squared average deviation* from a full station load. The fitness function, to maximise, favours solutions with some well-filled and some nearly empty workstations as opposed to solutions where all workstations have similar workload. The reasoning is that in extremely unbalanced solutions is easier to eliminate workstations. The function is computed as follows:

$$Fitness = \frac{\sum_{k=1}^{S} (W_k / C)^2}{S} \quad \textbf{(3.3)}$$

where W_k is the workload of workstation k.

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 given by

$$Fitness = \sum_{k=1}^{S} \frac{(C - W_k)^2}{S} \quad \textbf{(3.4)}$$

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

$$Fitness = 2\sqrt{\frac{\sum_{k=1}^{S} (W_{\max} - W_k)^2}{S}} + \frac{\sum_{k=1}^{S} (W_{\max} - W_k)}{S} \quad \textbf{(3.5)}$$

where $W_{\max}$ is the maximum workload. The authors give the first term a higher importance and multiply it by two.

The problems of type II have as goal the minimisation of cycle time for a given number of workstations. Anderson and Ferris (1994) consider as fitness function the minimisation of the maximum workload added by a penalty for unfeasible solutions. Kim et al (1998) propose a fitness function that distributes the workload as equal as possible between the workstations and favours solutions with workstations with workloads close to the average workload ($\overline{W}$). It is given by:

$$Fitness = \frac{1}{S} \sum_{k=1}^{S} \sqrt{\left| W_k - \overline{W} \right|} \quad \textbf{(3.6)}$$

Multi-criteria approaches are proposed by Kim et al (1996) and Ponnambalam et al (2000) addressing several objectives like minimising the number of workstations, minimising the cycle time, balancing workloads and maximising line efficiency. The first authors introduce a performance measure, called index of *work relatedness*, which aims to

assign related tasks to the same workstation. This objective is considered together with the other goals when assessing the quality of a balancing solution.

3.3.2.3 Other features

Most of the GA applications to ALBP use standard approaches, as the ones described in section 3.3.1, when generating the initial population, defining selection and replacement strategies, setting crossover and mutation probabilities and stopping criteria.

An original variation, called Adam-Eve GA, is presented by Feyzbakhsh and Matsui (1999). The initial population has only two individuals, however, the population size increases during its evolution, as offspring are inserted as new individuals in the population instead of replacing the parents. Due to a new operator, each individual faces *death* after a few generations. Levitin et al (2006) introduce a phenomenon called *cataclysm* which consists in, at the end of each iteration, create a whole new population preserving only the best individual from the previous generation. This aims to avoid premature convergence.

3.4 Ant colony optimisation algorithms

3.4.1 Overview

Ant colony optimisation algorithms are population-based procedures inspired on the behaviour of real ant colonies. Ants are known for being able to find the shortest path between their nest and a food source, without making use of visual cues; only by following pheromone trails released by other ants. The more intense is the trail, the higher the probability of an ant to follow it and thus reinforce the trail with its own pheromone. So, it is the colony as a whole that coordinates the activities without a direct communication between individual ants, as an isolated ant basically moves at random.

Figure 3.12 presents an illustration of a typical result of the so-called double bridge experiment, adapted from Bonabeau et al (1999). In this experiment, a food source is separated from the nest by a double bridge with two branches of different lengths. Initially there is no pheromone in the branches, having all, therefore, the same probability of being selected by the ants. The first ants returning to the nest are those who selected the shortest

path twice (to go from the nest to the food source and to return to the nest), so that, immediately after these ants have returned, more pheromone is present in the short branches than in the long branches, stimulating other ants to select the short branches. Sooner the colony converges to the shortest path. The collective behaviour that emerges is a form of *autocatalytic* behaviour, i.e., *positive feedback*, where the more ants are following the trail, the more attractive that trail becomes for being followed.

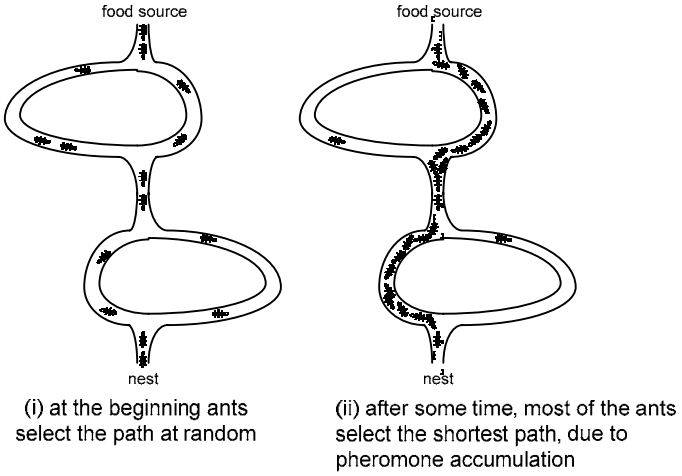

Figure 3.12 – The double bridge experiment

Ant algorithms were firstly presented by Dorigo et al (1991, 1996) as an approach to solve $\mathcal{NP}$-hard combinatorial optimisation problems. Although they have been originally applied to the travelling salesman problem (Dorigo and Gambardella, 1996, 1997), rapidly the scientific community showed a high curiosity and interest for this kind of approach, providing applications to other types of problem.

The Ant Colony Optimisation (ACO) meta-heuristic presented by Dorigo et al (1999) provides a unifying framework for most applications of ant algorithms to combinatorial optimisation problems. According to Stültze and Dorigo (1999), all ant algorithms previously developed fit into the ACO meta-heuristic, so they all can be called ACO algorithms.

The basic idea underlying ACO algorithms is to use a positive feedback mechanism, based on an analogy with the *pheromone-laying pheromone-following* behaviour of ants, to

reinforce good solutions of combinatorial optimisation problems. Each ant builds, step-by-step, a single solution. During this procedure the ant takes into account the information left by other ants (pheromone trails) and, eventually, other available information about the problem (heuristic information). By the end, good solutions emerge resulting from the indirect communication between the ants.

Artificial ants are different from real ants in the following aspects:

(i) they do not move continuously (the time is assumed to be discrete);

(ii) they have memory to store their past actions;

(iii) they are not completely blind as they possess some information about the problem to solve;

(iv) the amount of pheromone released by the ants is a function of the quality of the solution;

(v) the timing in pheromone laying is problem dependent and often it is very different from what happens with real ants (for example, when the pheromone is released only after the solution is completed).

These extra capabilities of the artificial ants increase their efficiency and effectiveness.

In order to illustrate how ACO algorithms work, its application to the travelling salesman problem (TSP) will now be described. The TSP is a path optimisation problem, so the ant colony metaphor is easily adapted. The goal is to find a closed tour of minimal length connecting n nodes where each node must be visited once. Each ant builds a solution to the TSP by moving on the problem graph from one node to another until it completes a tour. During an iteration of the algorithm, m ants build a tour executing n steps. At each step, an ant is in node i and it applies a probabilistic decision (state transition) rule to select the next node j to be visited. The edge (i,j) is then added to the tour under construction. For each ant, the transition from node i to node j depends on two factors:

- *Visibility* (η_{ij}) – Artificial ants are provided with some local information about the problem. In the TSP, visibility (or heuristic information) is related to the distance between two nodes, usually the inverse of the distance ($\eta_{ij}=1/d_{ij}$), which means that the lower the distance between nodes i and j, the higher the probability of going from i to j.

Visibility helps directing the search, although a constructive method based exclusively on heuristic information would produce low quality solutions.

- *Pheromone trail* (τ_{ij}) – The amount of virtual pheromone trail on edge (i,j) represents the learned desirability of selecting node j when in node i. The more ants have chosen edge (i,j) in previous iterations, the more intense will be the trail. The pheromone trail information is changed after each algorithm's iteration to reflect the experience acquired by the ants.

The state transition rule, i.e., the probability of ant k to go from node i to node j in the t^{th} iteration of the algorithm is called *random proportional transition rule* (Dorigo et al, 1991, 1996) and it is given by:

$$p_{ij}^{k}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{l \in A_i^k}\left[\tau_{il}(t)\right]^{\alpha}\left[\eta_{il}\right]^{\beta}} \qquad \textbf{(3.7)}$$

where A_i^k is the set of available nodes of ant k when in node i, and α and β are two adjustable parameters that determine the relative importance of pheromone intensity versus visibility. If α=0, the closest nodes are more likely to be selected, corresponding to a classic stochastic greedy heuristic with multiple starting points (since ants are initially distributed on the nodes at random). If β=0, only pheromone information is guiding the search, but this situation may lead to premature convergence of the algorithm. Therefore, it is necessary to establish a trade-off between both types of information.

Dorigo and Gambardella (1997) developed an enhanced version of the transition rule, called *pseudo-random proportional rule*, which allows a balance between the *exploration* of new edges and the *exploitation* of the currently best known edges. By applying this rule, an ant k in node i will select node j, in the t^{th} iteration, according to:

$$j = \begin{cases} J_1 = \arg\max_{j \in A_i^k}\left\{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}\right\} & \text{if } r \le r_0 \quad \text{(exploitation)} \\ J_2 : p_{iJ_2}^{k}(t) = \dfrac{\left[\tau_{iJ_2}(t)\right]^{\alpha}\left[\eta_{iJ_2}\right]^{\beta}}{\sum_{l \in A_i^k}\left[\tau_{il}(t)\right]^{\alpha}\left[\eta_{il}\right]^{\beta}} & \text{otherwise} \quad \text{(biased exploration)} \end{cases} \qquad \textbf{(3.8)}$$

where r is a random number uniformly distributed in the interval [0,1] and r_0 is a pre-defined parameter, which determines the relative importance of exploitation versus

exploration. Whenever an ant in node *i* has to select a node *j*, it samples a random number ($0 \leq r \leq 1$). If $r \leq r_0$ then the best node (J_1) is selected, otherwise a node (J_2) is selected according to its probability ($p_{iJ_2}^k(t)$).

After completing a tour, each ant *k* deposits an amount of pheromone $\Delta\tau_{ij}^k(t)$ on each visited edge (*i*,*j*) that depends on the quality of the solution (distance of the tour) and it is given by:

$$\Delta\tau_{ij}^k(t) = \begin{cases} Q / L^k(t) & \text{if } (i,j) \in T^k(t) \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{(3.9)}$$

where $T^k(t)$ is the tour built by ant *k* at iteration *t*, $L^k(t)$ is its length and *Q* is a pre-defined parameter. To be noticed that an iteration *t* of the algorithm is completed when all ants have done a tour, so that the pheromone released by ants in one iteration does not influence the decision of the other ants in the same iteration. For ease of implementation, all ants will release their pheromone simultaneously at the end of each iteration.

An important issue is *pheromone evaporation*. In order to ensure efficient solution space exploration and avoid stagnation, it is necessary to allow the decay of the trail intensity. This is implemented by the introduction of an evaporation coefficient ρ ($0 \leq \rho < 1$) which decreases the trail intensity of each edge. At the beginning of the algorithm, an initial amount of pheromone τ_0 is present on all edges.

The global pheromone update effect of all ants on each edge (*i*,*j*) in the t^{th} iteration is given by :

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \qquad \textbf{(3.10)}$$

Other pheromone update strategies are possible, like the use of *elitist ants*. An elitist ant is an ant which, in every iteration, reinforces the edges of the best tour found so far by the algorithm. The idea is that this reinforcement will direct the search of the other ants (in probability) towards a solution containing some edges of the best tour.

The Ant Colony System of Dorigo and Gambardella (1997) performs two types of pheromone update strategies: global and local. The global update is done, at the end of an iteration, solely by the ant that generated the best tour since the beginning of the algorithm.

Local updates are performed by the other ants while building the tours and not at the end of the iteration. When ant *k*, while building a tour, is in node *i* and selects node *j*, the pheromone intensity of edge (*i*,*j*) is updated as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\tau_0 \qquad \textbf{(3.11)}$$

The local update rule avoids the selection of a very good edge by all ants, as every time an edge is selected its pheromone level diminishes. This will favour exploration of not yet visited edges, preventing premature convergence of the algorithm.

3.4.2 ACO approaches for assembly line balancing

The use of ACO algorithms to solve the assembly line balancing problem follows the recent developments in combinatorial problem solving, that is influenced by techniques based on the behaviour of insect societies. An overview of the application of concepts inspired in colonies of social insects (ants and wasps) to solve manufacturing problems is presented in Cicirello and Smith (2001). A literature review of the application of ACO algorithms to several hard problems, like quadratic assignment, sequential ordering, job-shop scheduling, graph colouring, vehicle routing, generalized assignment, shortest common super sequence and network routing is provided by Dorigo et al (1999).

The literature reporting the use of ACO algorithms to solve assembly line balancing problems is scarce. Only two publications were found: (i) the conference paper of Bautista and Pereira (2002), who apply an ACO algorithm to solve the simple assembly line balancing problem and (ii) the paper of McMullen and Tarasewich (2003), reporting the use of ant techniques to address assembly line balancing problems with focus on the stochastic nature of task processing times. The main features of both works will be now briefly described.

The way artificial ants build an assembly line balancing solution in the approach proposed by Bautista and Pereira (2002) is straightforward: each ant iteratively selects a task for assignment using a constructive procedure. The probability of selecting a task *j* depends on the heuristic information about the task (η_j), in the form of a priority rule, and the pheromone trail intensity. The authors use thirteen priority rules available in the literature for the assembly line balancing problem (e.g., maximum processing time,

maximum number of immediate successors, maximum number of successors, etc.) and assign one priority rule to each ant.

Three pheromone release strategies are used:

(i) trail between consecutive assigned tasks – τ_{ij} is the trail intensity between tasks i and j;

(ii) trail between the task and the iteration in which it was assigned – τ_{ij} is the trail intensity between task j and its position i in the sequence of assigned tasks;

(iii) trail between the task and the workstation to which it was assigned – τ_{ij} is the trail intensity between workstation i and task j.

The probability of ant k to select task j is given by:

$$p_{ij}^{k} = \frac{\left[\tau_{ij}\right]^{\alpha}\left[\eta_{j}\right]^{\beta}}{\sum_{l \in A_i^k}\left[\tau_{il}\right]^{\alpha}\left[\eta_{l}\right]^{\beta}} \qquad \textbf{(3.12)}$$

where A_i^k is the set of available tasks (i.e., tasks that meet precedence and capacity constraints). τ_{ij} will depend on the pheromone release strategy and η_j will depend on the priority of each task for a given rule. The values of the different priority rules are linearly normalised between 1 and the number of available tasks, in order to eliminate the dissimilarity between the ranges of values of the different rules.

After all ants of an iteration of the algorithm have generated a balancing solution, a local search procedure is applied to the best solutions obtained. The search is guided by an objective function that minimises the idle time in the first workstations and maximises idle time in the last workstations, aiming to decrease the number of workstations of the solution. The neighbourhood is defined by (i) exchanging the workstation of two tasks or (ii) transferring a task to the previous workstation. Both of the movements are forced to build feasible solutions.

The updating of pheromone trails is performed exclusively by the best ants in each iteration and it takes into account the number of workstations of their solution (*BestSol*) and the number of workstations of the best solution found so far by the algorithm (*BestSolSoFar*). It is given by:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho \frac{BestSolSoFar}{BestSol} \quad \textbf{(3.13)}$$

In the procedures proposed by McMullen and Tarasewich (2003, 2006), the pheromone level associated with the assignment of a task j to the current workstation i is also the probability of task j being selected and it is given by:

$$ph_j = \left(metric_j \Big/ \sum_{l \in A_j} metric_l \right) + \left(M(j, I_j) \Big/ \sum_{l \in A_j} M(l, I_l) \right) \quad \textbf{(3.14)}$$

Depending on the strategic approaches selected by management, four different metrics are used to determine the overall attractiveness of task j to be assigned to the current workstation: (i) u_j, (ii) p_j, (iii) $u_j \times p_j$ and (iv) $u_j \times (1\text{-}p_j)$, where u_j is the utilisation of the current workstation after the assignment of task j and p_j is the probability of all tasks being completed on time if task j is assigned to the current workstation. The first term of expression (3.14) gives the relative desirability of task j (for a given metric) compared with the other available tasks. The second term is related with the traditional pheromone concept and it is explained as follows. M is a matrix that keeps the number of times that task j has been assigned to a workstation after a certain immediate predecessor I_j, since the beginning of the algorithm. If I_j is a frequent predecessor of task j in previous balancing solutions, then $M(j,I_j)$ will have a high value, incorporating, therefore, historical information in the task selection process.

After the assignment of all tasks, four solution quality measures are computed: (i) utilisation of assembly line layout, (ii) probability of all workstations to complete their tasks on-time (as task processing times are considered stochastic), (iii) composite measure of utilisation and on-time completion probability and (iv) design cost associated with the line layout. These objective functions will be used according to the strategic approach selected by the decision-maker.

3.5 Taboo search algorithms

3.5.1 Overview

Another popular meta-heuristic approach used to address the assembly line balancing problem is taboo search, introduced by Glover (1989, 1990). In this section only a glance at this search procedure will be given as well as brief references to literature publications on the subject.

Taboo search is a generalised local search procedure, for solving combinatorial optimisation problems, that uses information on the history of the search to overcome local optimality. It starts from an initial solution and iteratively moves to a neighbour solution which may or not lead to improvement. The decision of which neighbour solution should be visited is based on the examination of the whole neighbourhood or a subset of the neighbourhood of a solution. The best neighbour is selected, even if it is worse than the current solution. A neighbour solution is usually obtained by transferring tasks to different workstations or by swapping tasks from different workstations, similarly to the neighbour generation procedure of most simulated annealing approaches.

The underlying idea is to forbid some search directions at a determined iteration, in order to avoid cycling, by keeping some attributes of the last visited solutions in a structure called *taboo list* with a limited size. The use of 'short-term memory' avoids the procedure to be trapped at local optima while the use of 'long-term memory' allows the use of intensification and diversification strategies to refine the search process. *Intensification* aims at concentrating the search to a specific region of the solution space whereas *diversification* tries to lead the search direction into unvisited regions of the solution space.

3.5.2 Taboo search approaches for assembly line balancing

In the literature, there are several applications of taboo search to the assembly line balancing problem. Heinrici (1994) presents a comparison of simulated annealing and taboo search to solve the SALBP of type II. The initial solution is produced using a modified version of the ranked positional weight technique. The set of neighbour solutions is obtained by shifting tasks out of the workstation with the highest workload. If this does

not generate any feasible solution, then all possible transfers and swaps are performed. The reported computational tests showed that the taboo search performed equal or better than simulated annealing, for most of the tested problems.

Scholl and Voβ (1996) present taboo search algorithms for both type I and type II problems. These authors initially present a procedure to tackle the SALBP-II and then it is applied within a framework of a lower bound method to solve the SALBP-I. Initial solutions are obtained using heuristics based on priority rules and transitions to neighbour solutions are performed through transfer and swap moves. Good results of the computational experiments are reported.

Chiang (1998) presents four different versions of a taboo search procedure to address the SALBP-I. The initial solution is obtained via a constructive heuristic based on several priority rules. The transition to neighbour solutions is performed by λ-exchange moves, in which no more than λ tasks are exchanged for any two workstations. The performance is tested with set of test problems and the reported results are very good: except for a few cases, the procedure always finds the optimal solutions.

The only application of taboo search to assembly line balancing problems which reflects some operating conditions of real assembly lines is presented by Lapierre et al (2006). The developed algorithm allowed the exploration of unfeasible solutions, through cycle time violation, and uses two different neighbourhood structures: one focuses on reducing or increasing the 'half-empty' workstations and the other attempts to completely empty 'near-empty' workstations. The proposed taboo search procedure is applied to a real line with workstations located on both sides of the conveyor, with two possible conveyor heights.

3.6 Chapter conclusions

There is a growing interest in the use of meta-heuristics to solve combinatorial optimisation problems due to their capability to handle a wide range of problems with a relatively low algorithm complexity and to the good performance achieved in most cases. The analogy and inspiration from natural systems is also an extra aspect that motivates

researchers. Besides these characteristics, the motivation for using meta-heuristics in this particular work was their flexibility to incorporate complex characteristics of the problems.

The review of the application of these techniques to the assembly line balancing problem showed that the emphasis of the researchers is still on the definition of the technique's parameters instead of solving more complex problems. The simple assembly line balancing problem remains the most researched problem, mainly because it is a benchmark problem with a large number of data sets with known optimal solutions. This makes it easier to evaluate the performance of the developed procedures, as solutions can be compared with the optimal values.

The characteristics of real world assembly lines are much more complex than the ones addressed by most of the techniques reported in the literature. This represents a gap between research directions and industrial needs.

The present study is driven by the need to model the assembly line balancing problem in a way that reflects the operating conditions of real world assembly lines. Complex features of the problem like mixed-model production, use of parallel workstations, zoning constraints are included in the definition of the problem and meta-heuristic based procedures are developed to solve it. Also, some real assembly lines are studied in order to validate the assumptions of the proposed procedures and to better understand the real industrial problems.

The following chapters present the definition of the addressed problems – balancing mixed-model (i) straight lines, (ii) U-shaped lines and (iii) 2-sided lines – and describe the meta-heuristic based procedures developed to tackle them.

4

Balancing straight assembly lines

Contents

- **Chapter introduction**
- **Definition of the mixed-model ALBP with parallel workstations**
- **Simulated annealing based approach**
- **Genetic algorithm based approach**
- **Ant colony optimisation based approach**
- **Addressing the problem of type II**
- **Computational experience**
- **Chapter conclusions**

4.1 Chapter introduction

In this chapter, the addressed problem – the mixed-model straight assembly line balancing problem (MALBP) – is formally described using a mathematical programming model and the procedures developed to tackle it are presented. Three procedures based on meta-heuristics (simulated annealing, genetic algorithms and ant colony optimisation) were developed to address both type I and type II problems and their performance is compared through a set of computational experiments.

Although some of what is described in this chapter refers to previous developed work, namely, the mathematical programming model and the simulated annealing procedure for type I (Simaria, 2001), it was decided to include it in this document, with the purpose of better describing the whole research.[*]

4.2 Definition of the mixed-model ALBP with parallel workstations

4.2.1 Problem assumptions and constraints

As it was referred earlier, the recent market trends show that there is a growing demand for customised products, increasing the pressure for manufacturing flexibility. Mixed-model assembly lines are an adequate production system for companies to implement manufacturing postponement strategies, being an important piece of the supply chain.

In the addressed problem, the assembly line is configured to produce a set of similar models of a product (m=1,…,M), in any order or mix, over a pre-specified planning horizon, P. The forecasted demand, over the planning horizon, for model m is D_m, requiring the line to be operated with a cycle time given by:

$$C = P / \sum_{m=1}^{M} D_m \qquad \textbf{(4.1)}$$

[*] Parts of the work presented in this chapter are published in Simaria & Vilarinho (2001, 2004) and Vilarinho & Simaria (2002, 2006).

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

$$q_m = D_m / \sum_{p=1}^{M} D_p \qquad \textbf{(4.2)}$$

Each model has its own set of precedence relationships, but there is a subset of tasks common to all models. Hence, the precedence diagrams for all the models can be combined and the resulting one has N tasks (i=1,…,N are the task numbers of the tasks in the combined precedence diagram). 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).

The work of Bukchin et al (2002) and Bukchin and Rabinowitch (2005) states that in modern assembly lines workers are expected to be more versatile and one can assume that each worker is able to perform any task on the line. Following this assumption, they allow the assignment of the same task to different workstations when performed in different models. Although this idea seems adequate for the actual industry environment, there is no evidence of a successful implementation in real world assembly lines. So, in this model the traditional assumption of the use of specialised operators, trained to perform a small set of tasks, is maintained. This way, a task that is common to several models must be assigned to the same workstation, for the different models. The first set of decision variables is defined as:

$$x_{ik} = \begin{cases} 1, \text{ if task } i \text{ is assigned to workstation } k \\ 0, \text{ otherwise} \end{cases} \quad (i = 1,..., N; k = 1,..., LL) \qquad \textbf{(4.3)}$$

where LL is the line length, i.e., the number of workstations in the assembly line. The assignment of a task to only one workstation, regardless of the model being assembled, is guaranteed by the following set of constraints:

$$\sum_{k=1}^{LL} x_{ik} = 1 \quad (i = 1,...,N) \qquad \textbf{(4.4)}$$

To ensure that the precedence constraints of the assembly process are not violated, the set of constraints (4.5) is included in the model, taking into account, for each task i, the set successors of task i (Suc_i), i.e., the set of tasks that cannot be performed before task i is completed, which is derived from the combined precedence diagram. No successor of task

i will, then, be assigned to an earlier workstation than the workstation to which task i is assigned.

$$\sum_{k=1}^{LL} kx_{ik} - \sum_{k=1}^{LL} kx_{jk} \leq 0 \quad (i = 1,...,N;\ j \in Suc_i) \qquad \textbf{(4.5)}$$

The workload corresponding to the set of tasks assigned to a workstation cannot exceed the workstation's capacity, a crucial factor for the line production rate. Most of the techniques used to solve the ALBP require the assignment of each task to a single workstation and, consequently, the production rate is limited by the longest task time. This assumption can be relaxed by using parallel workstations in such a way that two or more replicas of a workstation can perform the same set of tasks on different assemblies. The introduction of parallel workstations not only allows for cycle times shorter than the longest task time, allowing an increase in the production rate, but also provides greater flexibility in designing the assembly line (Buxey, 1974). However, as the number of parallel workstations increases, so does the number of different tasks performed by each operator. If the replication of workstations is not controlled, one can lose one of the main advantages of using assembly lines: the use of low skilled labour that can easily be trained to perform a small number of tasks.

Most of the models for the ALBP with parallel workstations proposed in the literature base the decision to create parallel workstations in a trade-off between the incremental tooling/equipment cost of the duplicated workstation and the cost of hiring operators for the original line in order to satisfy the demand (e.g., Johnson, 1983, Pinto et al, 1975, 1981, Bard, 1989, Daganzo and Blumenfeld, 1994, Askin and Zhou, 1997). McMullen and Frazier (1997, 1998) allow the replication of a workstation as long as its utilisation increases. Schofield (1979) and Sarker and Shantikumar (1983) define a limit on the number of parallel workstations to control the replication process, while Buxey (1974) includes a limit on the number of tasks per workstation. In all these approaches, tasks with processing times much shorter than the cycle time can trigger the replication of workstations, which can lead to an excessive number of parallel workstations.

To address this issue, the proposed approach uses a mechanism to control the replication of workstations, based on the approach originally developed for the single-model assembly line balancing problem (Simaria and Vilarinho, 2001). The model

allows the decision-maker to establish the conditions under which a workstation can be replicated by defining a minimum processing time that triggers the replication process (*MRT* – minimum replication time). This means that only workstations that perform tasks with processing time higher than *MRT* for, at least, one of the models, are allowed to be replicated, that is, are allowed to have two or more operators working in parallel (in replicas of the workstation). 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:

$$R_k = \left\lceil \frac{\max\limits_{m=1,\ldots,M;i=1,\ldots,N} \{t_{im} x_{ik}\}}{MRT} \right\rceil \quad (k = 1,\ldots LL) \tag{4.6}$$

By using parallel workstations, it is necessary to distinguish between the total number of operators working on the line and the number of different workstations in the line (*LL*). This way, if some operators carry out the same set of tasks in parallel workstations, there will be more operators than different workstations. The total number of operators working on the assembly line (*S*) is computed by the sum of the number of replicas of all workstations, as follows:

$$S = \sum_{k=1}^{LL} R_k \tag{4.7}$$

The capacity of a workstation depends on the tasks assigned to it. Let task h be a candidate for assignment to workstation k. W_{km} is the workload of workstation k for model m, after the assignment of 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:

$$W_{km} = \sum_{i=1}^{N} t_{im} x_{ik} + t_{hm} \quad (k = 1,\ldots LL; m = 1,\ldots, M) \tag{4.8}$$

The capacity constraints are defined as follows: if workstation k performs a task with a processing time higher than *MRT*, for at least one of the models, or if task h has a processing time higher than *MRT*, for at least one of the models, then constraint (4.9) holds. In this case, the capacity of the workstation is the required to perform the task with processing time higher than *MRT*.

$$W_{km} \leq R_k \cdot C \quad (k = 1,\ldots,LL; m = 1,\ldots,M) \tag{4.9}$$

If all tasks in workstation k and task h have processing times not higher than MRT, constraint (4.10) must hold, i.e, the capacity of the workstation is equal to the cycle time.

$$W_{km} \leq C \qquad (k = 1,...,LL; m = 1,...,M) \tag{4.10}$$

The idle time of a workstation is the difference between the capacity of the workstation and its workload. s_{km} is idle time of workstation k due to model m and it is computed by the following set of equations:

$$\sum_{i=1}^{N} t_{im} x_{ik} + s_{km} = R_k \cdot C \quad (k = 1,...LL; m = 1,...,M) \tag{4.11}$$

Zoning constraints may also be included in the problem. Positive zoning constraints force pairs of tasks to be assigned to the same workstation and are defined by the set of constraints (4.12), where ZP represents the set of pairs of tasks that must be assigned to the same workstation.

$$\sum_{k=1}^{LL} kx_{ik} - \sum_{k=1}^{LL} kx_{jk} = 0 \quad ((i, j) \in ZP) \tag{4.12}$$

The set of constraints (4.13) defines the negative zoning constraints, which forbid the assignment of pairs of tasks to the same workstation. ZN is the set of pairs of incompatible tasks.

$$\sum_{k=1}^{LL} kx_{ik} - \sum_{k=1}^{LL} kx_{jk} \neq 0 \quad ((i, j) \in ZN) \tag{4.13}$$

4.2.2 Objective function

The main goal of ALBP of type I is to minimise the number of workstations for a given cycle time. So, the first approach to address this goal was the use of an objective function that minimised the number of the workstation to which the last task of the precedence diagram was assigned. This function was given by:

$$Minimise \quad \sum_{k=1}^{K} kx_{Nk} \tag{4.14}$$

where K is an upper bound of the number of workstations, as when formulating the problem one does not know how many workstations will have the line ($K \geq LL$). However,

this function is only adequate when the precedence diagram converges to one final task (the N^{th} task), which is not the case, for many diagrams.

Another way to address this goal is through the minimisation of the idle time of the line, because a line with a lower number of workstations will necessarily have a lower idle time. To cope with the mixed-model nature of the problem, an objective function called weighted idle time (*WIT*) was developed. It minimises the weighted sum (considering each model production share, q_m) of the idle time of the workstations in the line and it is given by:

$$Minimise \quad WIT = \sum_{k=1}^{LL} \sum_{m=1}^{M} q_m \left(R_k \cdot C - \sum_{i=1}^{N} t_{im} x_{ik} \right) \qquad \textbf{(4.15)}$$

The values of *WIT* are different from problem to problem, as they vary according to the cycle time and task processing times of the problem instance. An alternative measure, which is always within a fixed range of values, is the weighted line efficiency (*WE*). It varies between 0 and 1 and it gives a direct idea of the efficiency of the assembly line, regardless of the data of the problem instance: the more close to 1 (or 100%) the less idle time has the line. *WE* is an objective function to maximise and it is computed as follows:

$$Maximise \quad WE = \sum_{m=1}^{M} q_m \left(\frac{\sum_{i=1}^{N} t_{im}}{S \cdot C} \right) \qquad \textbf{(4.16)}$$

where S is the total number of operators of the line. This objective function is adequate for problems of type I and of type II. The only difference is in what is given as input and what is incognita. While for type I C is given and S is unknown, for type II the opposite occurs, i.e., S is given and C is unknown.

Besides the minimisation of the number of workstations (or the minimisation of cycle time, for problems of type II), additional goals, concerning workload smoothing, are also envisaged. The objective function B_b aims to balance the workload between workstations, i.e., for each model the idle time is distributed across workstations as equally as possible, and it is given by:

$$Minimise \quad B_b = \frac{LL}{LL-1}\sum_{k=1}^{LL}\left(\frac{\sum_{m=1}^{M} q_m s_{km}}{WIT} - \frac{1}{LL}\right)^2 \tag{4.17}$$

The value of function B_b varies between a maximum of 1, when the weighted idle time of the line is equal to the idle time of one of the workstations, and a minimum of 0, when *WIT* is equally distributed by all workstations in the line. A demonstration of these values is presented in Appendix 1.

Due to the mixed-model nature of the problem, each task processing time may vary among the different models and, so, the workload assigned to a workstation may also vary. In order to ensure that each operator performs approximately the same amount of work for each model being assembled, it is desirable to balance the workload within each workstation. To achieve this goal the objective function B_w was developed, which aims at smoothing the workload balance within each workstation and it is computed as follows:

$$Minimise \quad B_w = \frac{M}{LL(M-1)}\sum_{k=1}^{LL}\sum_{m=1}^{M}\left(\frac{q_m s_{km}}{S_k} - \frac{1}{M}\right)^2 \tag{4.18}$$

where S_k is the weighted idle time of workstation k, given by:

$$S_k = \sum_{m=1}^{M} q_m s_{km} \tag{4.19}$$

The value of function B_w varies between a maximum of 1, when the idle time of each workstation is only accountable to one model, and a minimum of 0, when it is equally distributed by all models in every workstation. (A demonstration of these values is presented in Appendix 1.) An important note is that workstations with no idle time for any model (i.e., with S_k=0) are not considered in the computation of B_w.

Figure 4.1 shows the distribution of the overall idle time of an assembly line in five different balancing scenarios, all with 4 models and 4 workstations. In this figure each matrix cell represents the idle time of workstation k due to model m (s_{km}) The model production shares are q_1=0.2, q_2=0.2, q_3=0.4 and q_4=0.2.

Scenario 1 represents the perfect balancing solution, with both functions B_b and B_w reaching their minimum values. The idle time of the line is equally distributed between all

workstations and proportionally split by all models at each workstation. In scenario 2 the balance between workstations is perfect, but the balance within workstations is at its worse, with a single model causing the whole idle time.

The increase of functions B_b and B_w depends on the degree of balance between and within workstations, respectively. For instance, in scenario 3 half of the workstations have their idle time completely unbalanced, so B_w is around 0.5 (it is not exactly 0.5 because in workstation 4 the idle times are not perfectly balanced). A similar reasoning is applied to scenario 4, where 75% of the workstations are completely unbalanced leading to a value of B_w around 0.75.

Finally, scenario 5 shows a line perfectly balanced within workstations, but in which there is only one workstation with idle time. Although this is a situation for B_b=1 and, thus, a worst case situation, in practice, it can be seen as an opportunity to decrease the number of workstations of the line. In fact, by slightly increasing the cycle time of the line it could be possible to reassign the tasks of the last workstation to other workstations, eliminating, this way, workstation 4.

Scenario 1

k	s_{k1}	s_{k2}	s_{k3}	s_{k4}
1	6	6	3	6
2	6	6	3	6
3	6	6	3	6
4	6	6	3	6
B_b=0 B_w=0				

Scenario 2

k	s_{k1}	s_{k2}	s_{k3}	s_{k4}
1	24	0	0	0
2	0	24	0	0
3	0	0	12	0
4	0	0	0	24
B_b=0 B_w=1				

Scenario 3

k	s_{k1}	s_{k2}	s_{k3}	s_{k4}
1	12	0	0	0
2	0	12	0	0
3	6	6	3	6
4	6	6	9	18
B_b=0.13 B_w=0.52				

Scenario 4

k	s_{k1}	s_{k2}	s_{k3}	s_{k4}
1	12	0	0	0
2	0	12	0	0
3	0	0	0	12
4	12	12	12	12
B_b=0.25 B_w=0.76				

Scenario 5

k	s_{k1}	s_{k2}	s_{k3}	s_{k4}
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	24	24	12	24
B_b=1 B_w=0				

Figure 4.1 – Example of the variation of functions B_b and B_w for different scenarios

4.2.3 Complete mathematical programming model

The functions and constraints described in the previous sections are part of a mathematical programming model developed to formally describe the mixed-model assembly line balancing problem, presented globally in Figure 4.2. The objective function takes into account the values of *WE*, B_b and B_w, however it is obvious that *WE* is the most important goal because it directly addresses either the minimisation of the number of workstations or the minimisation of the cycle time. For this reason, it is multiplied by a user defined parameter (λ) that should be set $\lambda>1$. As the criterion of the global objective

function is maximisation, the symmetric values of functions B_b and B_w (defined by equations (4.17) and (4.18), respectively) had to be considered.

The model constraints are interpreted as follows:

(i) constraints ensuring that each task is assigned to only one workstation of the station interval (assignment constraints);

(ii) constraints ensuring that no successor of a task is assigned to an earlier station than the workstation to which is assigned that task (precedence constraints);

(iii) constraints ensuring that each workstation capacity is not exceeded, where the capacity of a workstation depends on whether or not it performs tasks with processing times for, at least, one model, higher than the minimum replication time, *MRT* (capacity constraints);

(iv) positive zoning constraints;

(v) negative zoning constraints,

(vi) set of constraints computing the line length (*LL*) in which the auxiliary binary variable y_k equals one, if the k^{th} workstation is used for assembly and zero, otherwise (in this set of constraints, $\mathcal{M}$ is a very large positive integer);

(vii) set of constraints defining the decision variables domain.

The large number of constraints and binary variables makes the proposed model highly complex, preventing it from being solved to optimality, at least for real world problems. It is, however, a very useful tool to formally describe the problem.

The use of a mathematical programming model to optimally solve a mixed-model assembly line balancing problem, with no parallel workstations or zoning constraints, was proposed by Göcken and Erel (1997, 1998). The computational experiments conducted by these authors revealed that the model was capable of solving problems with up to 40 tasks in the combined precedence diagram. For larger sized problems, it would be too large to obtain optimal solutions. It is clear that the model proposed in this section is more complex than the one proposed by these authors, because the addressed problem has additional characteristics that better reflect the operating conditions of real assembly lines (e.g., parallel workstations and zoning constraints). This way, the approach to solve the problem was based on the development of heuristic procedures that are able to efficiently search the solution space, providing good solutions in reasonable computation times.

Three meta-heuristic based procedures (simulated annealing, genetic algorithms and ant colony optimisation) were developed to tackle the MALBP described in this section and will be presented in the following sections.

$$\textit{Maximise} \quad \lambda WE - B_b - B_w$$

subject to :

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

$$\sum_{k=1}^{K} k x_{ik} - \sum_{k=1}^{K} k x_{jk} \leq 0 \qquad (i = 1,...,N;\ j \in Suc_i) \qquad (ii)$$

$$\sum_{i=1}^{N} t_{im} x_{ik} + s_{km} = \left\lceil \frac{\max_{m=1,...,M;i=1,...,N} \{t_{im} x_{ik}\}}{MRT} \right\rceil \cdot C \qquad (k = 1,...K; m = 1,...,M) \qquad (iii)$$

$$\sum_{k=1}^{K} k x_{ik} - \sum_{k=1}^{K} k x_{jk} = 0 \qquad ((i,j) \in ZP) \qquad (iv)$$

$$\sum_{k=1}^{K} k x_{ik} - \sum_{k=1}^{K} k x_{jk} \neq 0 \qquad ((i,j) \in ZN) \qquad (v)$$

$$\sum_{i=1}^{N} x_{ik} \leq \mathcal{M} y_k \qquad (k = 1,...K) \qquad (vi \text{ a})$$

$$\sum_{i=1}^{N} x_{ik} \geq y_k \qquad (k = 1,...K) \qquad (vi \text{ b})$$

$$LL = \sum_{k=1}^{K} y_k \qquad (vi \text{ c})$$

$$s_{km} \geq 0 \qquad (k = 1,...K; m = 1,...,M) \qquad (vii \text{ a})$$

$$x_{ik} \in \{0,1\} \qquad (i = 1,...,N; k = 1,...K) \qquad (vii \text{ b})$$

$$y_k \in \{0,1\} \qquad (k = 1,...K) \qquad (vii \text{ c})$$

$$LL > 0 \qquad LL \text{ is integer} \qquad (vii \text{ d})$$

Figure 4.2 – Mathematical programming model for the mixed-model ALBP with parallel workstations

4.3 Simulated annealing based approach

A two-stage procedure that uses the simulated annealing technique (described in section 3.2) was developed to tackle the MALBP of type I. In the first stage the procedure looks for a sub-optimal solution for the problem's main goal – the minimisation of the number of workstations. In the second stage, the additional goals of workload balancing are envisaged. In both stages a simulated annealing approach is used. The framework of this procedure is presented in Figure 4.3.

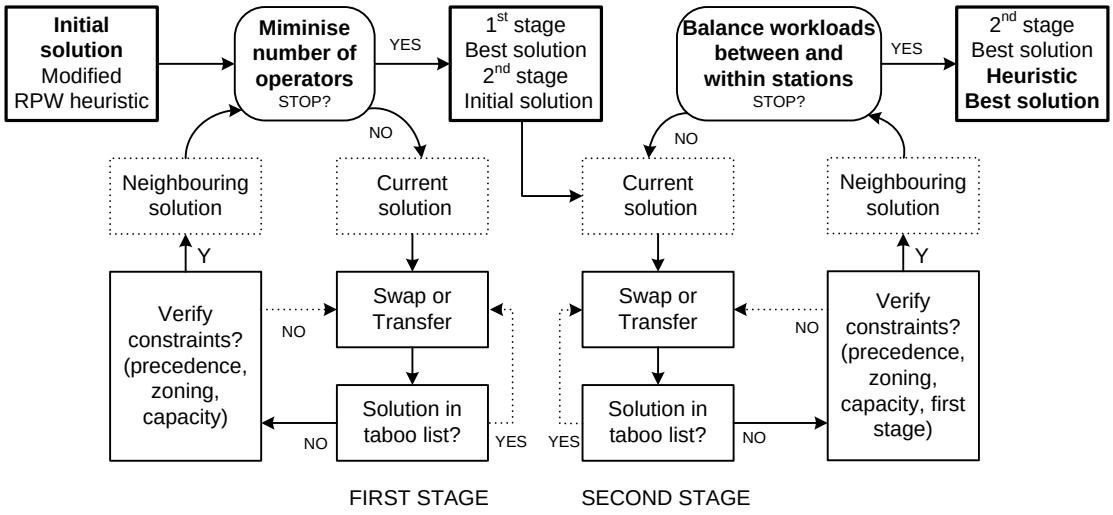

Figure 4.3 – The two-stage simulated annealing based procedure

4.3.1 The first stage

4.3.1.1 Initial solution

The initial solution is obtained using a version of the Ranked Positional Weight (RPW) heuristic proposed by Helgeson and Birnie (1961). The original RPW version only addresses the simple assembly line balancing problem, where one single model is assembled and no parallel workstations are allowed. The positional weight of a task in a mixed-model assembly line is the cumulative weighted average task time associated with itself and its successors. The weighted average task time of task i is the sum of the processing times of that task for each model weighted by the respective production share. The weighted average time of task i is then given by:

$$\bar{t}_i = \sum_{m=1}^{M} q_m t_{im} \ (i = 1,..., N) \qquad \textbf{(4.20)}$$

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 the original version of the RPW heuristic the cumulative duration of the tasks in a workstation cannot exceed the cycle time (hence the concept of feasible workstation) and

thus does not account for parallel workstations. The version of the RPW heuristic used to obtain the initial solution in the first stage of the proposed procedure redefines the concept of feasible workstation: if a workstation performs a task i with processing time larger than MRT, for, at least, one model, its time capacity is $C \cdot \left\lceil \frac{\max_m \{t_{im}\}}{MRT} \right\rceil$, otherwise is C.

The implemented version of the RPW heuristic also checks if the task to be assigned is not incompatible with any of the tasks already allocated to the workstation and merges tasks with positive zoning constraints, i.e., that need to be processed in the same workstation, previously, so that they are treated as only one task.

4.3.1.2 Solution evaluation criterion

In the first stage the procedure looks for the solution that minimises the number of workstations in the assembly line, so the weighted line efficiency, as defined in equation (4.16), is used as objective function.

4.3.1.3 Neighbouring solutions

A neighbouring solution can be generated by one of the following actions: (i) swapping two tasks in different workstations or (ii) transferring a task to another workstation. The tasks to be swapped, as well as the task and the workstation for the transfer, are randomly chosen. For any of these actions to result in a new neighbouring solution, the precedence, zoning and capacity constraints must be fulfilled. When this is not the case, a new swap or transfer must be attempted.

Only transfer movements may contribute to reduce the number of workstations, thus maximising line efficiency. Nevertheless, swap procedures are also required to ease the generation of successful transfer movements. So, the probability of performing a transfer procedure must be higher than for the swap procedure and, by default, probabilities of 75% and 25% were respectively set, although the user can set different values.

In both stages of the proposed procedure a taboo list is used to maintain information about the most recently generated neighbouring solutions, in order to avoid cycling.

4.3.2 The second stage

The goal of the second stage is to balance simultaneously the workloads between and within workstations, for the number of workstations obtained in the first stage. The initial solution of the second stage is the final solution found in the first stage. The criterion used to evaluate the neighbouring solutions generated in this second stage derives directly from the objective functions B_b and B_w computed by equations (4.17) and (4.18), respectively.

4.3.2.1 Neighbouring solutions

The generation of neighbouring solutions in the second stage also employs swap and transfer movements, but the tasks and workstations involved in these movements are selected to foster improving solutions, i.e., to improve workload smoothing. The steps of the swap and transfer movements are described as follows:

(i) Swap movement

STEP 1. Let Z be a randomly selected workstation and X the model whose idle time for that workstation has the highest deviation from the workstation average idle time $\left(X : \Delta s_{ZX} = \max_m \left\{ \left| s_{Zm} \cdot q_m - \frac{S_Z}{M} \right| \right\} \right)$.

STEP 2. If $s_{ZX} > \frac{S_Z}{M}$, then go to STEP 3, else, go to STEP 5.

STEP 3. Select the task assigned to workstation Z with the lowest processing time for model X $\left(T_1 : t_{T_1 X} = \min_i \{ t_{iX} \} \wedge i \in O_Z \right)$, where O_z is the set of tasks assigned to workstation Z.

STEP 4. From the set of tasks performed on model X that are not assigned to workstation Z and whose task time is higher than the task time of T_1, randomly select one $\left(T_2 : t_{T_2 X} > t_{T_1 X} \wedge T_2 \notin O_Z \right)$. Go to STEP 7.

STEP 5. Select the task assigned to workstation Z with the highest processing time for model X $\left(T_1 : t_{T_1 X} = \max_i \{ t_{iX} \} \wedge i \in O_Z \right)$.

STEP 6. From the set of tasks performed on model *X* that are not assigned to workstation *Z* and whose task time is smaller than the task time of T_1, randomly select one $\left(T_2 : t_{T_2X} < t_{T_1X} \wedge T_2 \notin O_Z\right)$.

STEP 7. If precedence, zoning, capacity and number of workstations constraints are met, swap tasks T_1 and T_2, else, go to STEP 1.

(ii) Transfer movement

STEP 1. Let *Z* be a randomly selected workstation and *X* the model whose idle time for that station has the highest deviation from the workstation average idle time $\left(X : \Delta s_{ZX} = \max_m \left\{\left| s_{Zm} \cdot q_m - \frac{S_Z}{M}\right|\right\}\right)$.

STEP 2. If $s_{ZX} > \frac{S_Z}{M}$, then go to STEP 3, else, go to STEP 5.

STEP 3. Select a task not assigned to workstation *Z* with processing time for model *X* higher than for the other models $\left(T_1 : t_{T_1X} = \max_m \left\{t_{T_1m}\right\} \wedge T_1 \notin O_Z\right)$.

STEP 4. If precedence, zoning, capacity and number of workstations constraints are met, transfer task T_1 to workstation *Z*, else go to STEP 1.

STEP 5. Select the task assigned to workstation *Z* with the highest processing time for model *X* $\left(T_1 : t_{T_1X} = \max_i \left\{t_{iX}\right\} \wedge i \in O_Z\right)$.

STEP 6. Randomly select a workstation (*W*) where the workload for model *X* is lower than the workstation average idle time $\left(W : s_{WX} < \frac{S_W}{M}\right)$.

STEP 7. If precedence, zoning, capacity and number of workstations constraints are met, transfer task T_1 to workstation *W*, else, go to STEP 1.

As the goal in this second stage is to balance the workloads, swap movements are more likely to contribute towards this end (probabilities of 75% for swap and 25% for transfer moves are set as the default). If after a predefined number of attempts neither swap nor transfer movements lead to a neighbouring solution, tasks or workstations involved in these movements will be randomly selected to force a new neighbouring solution.

4.3.3 Parameter settings

A common annealing schedule was used for both stages of the procedure, in which the following control parameters were defined:

(i) Initial temperature (T_0): Computational experience showed that the values of the objective functions never changed by more than 10% between two neighbouring solutions. So, for an initial temperature of 50 it is guaranteed that at least 80% of the inferior solutions are accepted (for T=50, $p = e^{-\frac{10}{50}} = 0.82$).

(ii) Temperature reduction function: The geometric function with a temperature reduction factor of 0.9 (T_i=0.9T_{i-1}) was used at each stage.

(iii) Length of each temperature level (L): A dominant factor on the computational effort associated with the solution of the problem is the number of tasks (N). So, in order to restrict the computational effort to the first order of the dominant factor, the number of solutions searched at each temperature level was set to φN, where φ is a user defined constant (φ=1 is the value suggested by default).

(iv) Stopping criteria: Two alternative criteria were set. In the first one, a freezing temperature of 10 is set, which means that 16 temperature levels are used ($T_0 a_i^{15}$=50(0.9^{15})=10.29). In the second one, it is admitted that, if in five consecutive temperature levels 85% of the generated solutions are rejected, then the probability of replacing the best solution found is very small and the procedure is then terminated.

4.3.4 Numerical illustration

A numerical example, with the following characteristics, is used to illustrate the proposed procedure.

- Two models, A and B, are simultaneously assembled on a line over a planning horizon of 480 t.u. (time units). The demand for each model is, respectively, 20 and 28 units (then, the cycle time is C=10, q_A=42% and q_B=58%).
- The combined precedence diagram, with 25 tasks, is depicted in Figure 4.4, where each node represents a task and each arc represents a precedence relation between a pair of tasks.

- The task processing times for the two models (t_A and t_B) are shown in Table 4.1.
- Tasks 9 and 10 cannot be executed on the same workstation (negative zoning constraints).
- Only workstations performing tasks with a processing time greater than the line cycle time can be replicated (*MRT*=*C*).

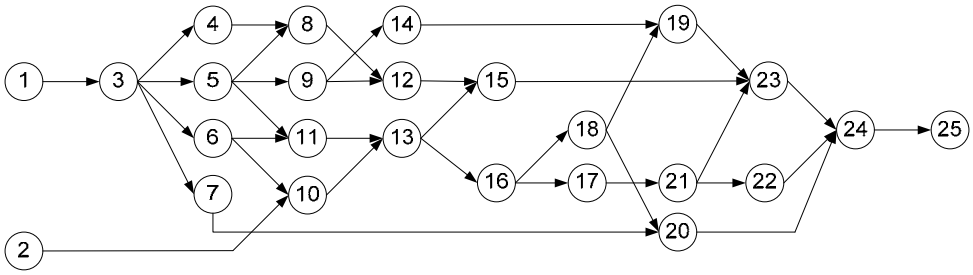

Figure 4.4 – Combined precedence diagram of the numerical example

The initial solution is determined by the modified version of the RPW heuristic described in section 4.3.1.1. The weighted average processing times ($\bar{t}$) and average positional weights (*PW*) of each task are also shown in Table 4.1.

Table 4.1 – Processing times and average positional weights for the numerical example

Task	t_A	t_B	$(\bar{t})$	*PW*	Task	t_A	t_B	$(\bar{t})$	*PW*
1	0	2.0	1.2	115.4	**14**	1.3	0	0.5	19.7
2	7.7	7.7	7.7	54.4	**15**	5.5	5.5	5.5	23.4
3	7.3	7.3	7.3	114.2	**16**	1.9	2.0	2.0	44.2
4	15.0	15.0	15.0	46.6	**17**	3.7	0	1.6	26.3
5	8.8	8.8	8.8	85.3	**18**	9.4	9.4	9.4	33.8
6	6.2	0	2.6	66.2	**19**	1.3	1.3	1.3	19.2
7	3.6	0	1.5	15.8	**20**	0	9.0	5.2	14.3
8	0	2.0	1.2	31.6	**21**	2.0	2.0	2.0	24.8
9	6.6	6.6	6.6	38.9	**22**	4.7	4.7	4.7	13.8
10	2.5	2.5	2.5	46.7	**23**	9.6	8.2	8.8	17.9
11	5.5	5.5	5.5	61.1	**24**	4.1	3.7	3.9	9.1
12	7.1	7.1	7.1	30.5	**25**	12.5	0	5.3	5.2
13	5.9	5.9	5.9	55.6					

Figure 4.5 illustrates some steps of the procedure applied to the numerical example. In each of the tables shown in the figure a line balancing solution is shown. To simplify the schema, only the workstations where changes occurred are represented in the neighbouring solutions. The content of each column in these tables is the following: (*K*) workstation index, (*Tasks*) set of tasks assigned to the workstation and (*R*) number of replicas of the workstation. The last line of each table shows the total number of operators required by the solution, *S*, and the between (B_b) and within (B_w) workstation workload balancing values.

The initial solution requires a total of 18 operators (including operators working in parallel workstations). After a number of swap and transfer movements, starting from the initial solution, an intermediate solution is obtained. From this solution, the heuristic is able to reduce the number of workstations, performing the transfer procedures shown in the figure. The best solution found for the first stage of the heuristic indicates that 16 workstations, including replicas, are required (for this solution B_b=0.05 and B_w=0.22). The best solution found in the first stage of the procedure is used as the initial solution for the second stage. In this stage, the number of workstations remains constant, while the workload balancing value (B_b+B_w) is reduced. The final solution shows an improvement of about 30%.

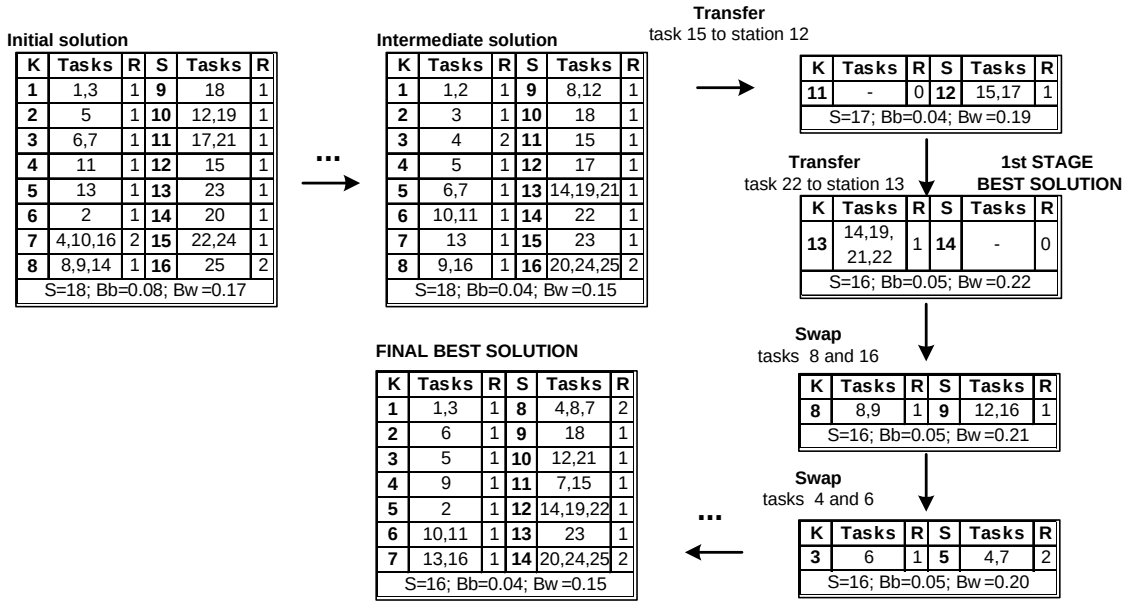

Figure 4.5 – Application of the SA based procedure to the numerical example

4.4 Genetic algorithm based approach

Genetic algorithms and its main concepts have been previously characterised in chapter 3. The structure of the proposed genetic algorithm based procedure to tackle the mixed-model assembly line balancing problem of type I is a standard one, with its main steps presented in Figure 4.6. The detailed application to the addressed problem is described in the following sections.

4.4.1 Representation of solutions

The encoding of solutions in the proposed procedure is of type 'one-to-one' (Falkenauer, 1998), which means that each solution is represented exactly by one chromosome and the decoding of each chromosome results in exactly one solution for the problem. A standard encoding scheme is used in which the chromosome is a string of length *N*. Each element of the chromosome represents a task and the value of each element represents the workstation to which the corresponding task is assigned. An example of this type of encoding scheme was already presented in Figure 3.6 of chapter 3.

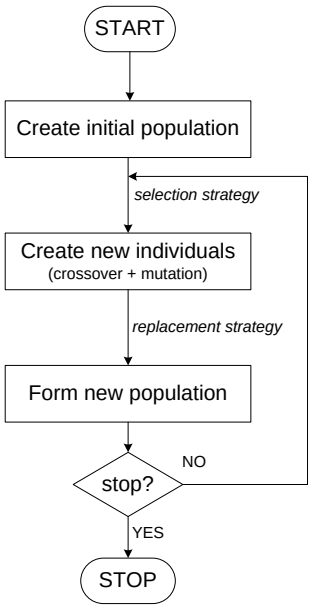

Figure 4.6 – Global structure of the genetic algorithm based approach

4.4.2 Initial population and fitness

The initial population is composed by a set of individuals (or chromosomes), each of them representing a solution for the MALBP of type I, described in section 4.2. The individuals of the initial population are generated via a simple constructive heuristic, which uses some common priority rules in assembly line balancing problems, namely:

- *maximum processing time* for all models ($\max_m \{t_{im}\}$);
- *maximum average processing time* – the average processing time of a task is the sum of the processing times of that task for each model weighted by the respective production share ($\bar{t}_i = \sum_m q_m t_{im}$);
- *maximum ranked positional weight* – in a mixed-model assembly line, the positional weight of a task is the cumulative average task processing time associated with itself and its successors;
- *maximum number of direct successors* – the number of direct successors of each task i is the number of tasks in set Suc_i, as defined in section 4.2;
- *maximum total number successors* of the combined precedence diagram.

Each time a task must be selected for assignment, from the set of available tasks, the heuristic randomly selects the priority rule to be used. As stated in the problem definition, a workstation capacity depends on the type of tasks that it performs. If it performs a task with a processing time for a model t_{im} higher than MRT, then its capacity is $C \cdot \left\lceil \frac{\max_m \{t_{im}\}}{MRT} \right\rceil$, otherwise is C. The heuristic also checks for positive and negative zoning constraints.

Since the procedure chooses randomly the priority rules to be used for assigning tasks to workstations, it ensures that different individuals will be created. The size of the population is fixed during all generations and was set to 50, a typical figure used by many researchers (Falkenauer, 1998).

The goal of genetic algorithms is to find the most fit individual over a set of generations. The fitness function is then, typically, a maximisation function. In this procedure, the fitness function is a combination of the objectives to achieve for the MALBP-I, namely, the maximisation of the weighted line efficiency (WE) and the

smoothing of workloads between (B_b) and within (B_w) workstations. The fitness function is thus computed as follows:

$$Maximise\ F = \lambda WE - B_b - B_w \qquad \textbf{(4.21)}$$

The lower the number of operators and the values of functions B_b and B_w (given by equations (4.17) and (4.18), respectively) the higher the value of F. As WE, B_b and B_w are within the value range [0,1], the term λWE is dominant for $\lambda>1$, so, the procedure minimises the number of workstations before the secondary goals become active.

4.4.3 Selection and genetic operators

The selection of the individuals for mating is done using tournament, a very popular strategy that aims to imitate mutual competition of individuals during casual meetings, already described in section 3.3, with the typical value of 2 for the tournament size.

The main genetic operator is the crossover, which has the role to combine pieces of information from different individuals in the population. Two parents (P_1 and P_2) are selected from the tournament list and a crossover point (cp), an integer randomly generated from [1, LL], is selected. The combination of P_1 and P_2 will produce two offspring (O_1 and O_2). To generate offspring O_1 (O_2), the assignment of workstations 1 to cp is copied from P_1 (P_2) and the remaining positions are copied from the assignment of workstations cp+1 to LL from P_2 (P_1). Figure 4.7 illustrates a crossover, for the numerical example presented in section 4.3.3.

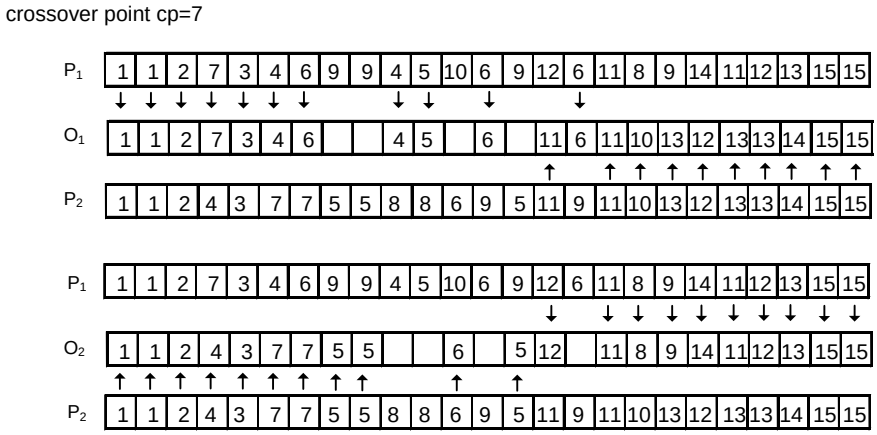

Figure 4.7 – Generation of two offspring through crossover

As it is shown in Figure 4.7, although precedence constraints are verified, the crossover produces some tasks without any workstation associated (tasks 8, 9, 12 and 14 for O_1 and tasks 10, 11, 13 and 16 for O_2). These tasks must, therefore, be reassigned in order to achieve feasible individuals. Other two types of tasks must also be reassigned: (i) tasks that violate zoning constraints and (ii) tasks assigned to workstations with low workload (Kim et al, 2000). If it is possible a reassignment of these last tasks to other workstations, then the workstation to which they were previously assigned will disappear, reducing the total number of workstations. Thus, if a workstation has an average workload, given by equation (4.22), inferior to a minimum workload (set to $C/2$, by default), all of its tasks must be reassigned.

$$\overline{W_k} = \sum_{i=1}^{N} x_{ik}\bar{t}_i \quad (k = 1,...,LL) \qquad \textbf{(4.22)}$$

The reassignment procedure aims to allocate the tasks to workstations in such a way that precedence and zoning constraints are satisfied and, if possible, the number of workstations is reduced. For each task i to be reassigned (starting with the tasks which have no precedent tasks to be reassigned), the procedure computes the earliest (E_i) and the latest (L_i) workstations to which task i can be assigned (Scholl 1999), according to the precedence relationships between tasks. From the range of workstations [E_i, L_i] task i is assigned to the first one that meets the capacity and zoning constraints. When it is not possible to find a feasible workstation within [E_i, L_i], a new workstation is opened to perform the task. Figure 4.8 shows an example of the reassignment procedure for the balancing solution corresponding to offspring O_1 of Figure 4.7.

Tasks to reassign: 8, 9, 12, 14

K	Tasks	R	W_A	W_B
1	1,2	1	7.7	7.7
2	3	1	7.3	7.3
3	5	1	8.8	8.8
4	6,10	1	8.7	2.5
5	7,11	1	9.1	5.5
6	13	1	5.9	5.9
7	4	2	15.0	15.0
8				
9				
10	18	1	9.4	9.4
11	15,17	1	9.2	5.5
12	20	1	0.0	9.0
13	19,21,22	1	8.0	8.0
14	23	1	9.6	8.2
15	24,25	2	16.6	3.7

→ task 8: E_8=7; L_8=11 assigned to station 7

K	Tasks	R	W_A	W_B
1	1,2	1	7.7	7.7
2	3	1	7.3	7.3
3	5	1	8.8	8.8
4	6,10	1	8.7	2.5
5	7,11	1	9.1	5.5
6	13	1	5.9	5.9
7	4,**8**	2	15.0	17.0
8				
9				
10	18	1	9.4	9.4
11	15,17	1	9.2	5.5
12	20	1	0.0	9.0
13	19,21,22	1	8.0	8.0
14	23	1	9.6	8.2
15	24,25	2	16.6	3.7

→ task 9: E_9=3; L_9=11 assigned to station 8

K	Tasks	R	W_A	W_B
1	1,2	1	7.7	7.7
2	3	1	7.3	7.3
3	5	1	8.8	8.8
4	6,10	1	8.7	2.5
5	7,11	1	9.1	5.5
6	13	1	5.9	5.9
7	4,8	2	15.0	17.0
8	**9**	1	6.6	6.6
9				
10	18	1	9.4	9.4
11	15,17	1	9.2	5.5
12	20	1	0.0	9.0
13	19,21,22	1	8.0	8.0
14	23	1	9.6	8.2
15	24,25	2	16.6	3.7

→ task 12: E_{12}=8; L_{12}=11 assigned to station 9

K	Tasks	R	W_A	W_B
1	1,2	1	7.7	7.7
2	3	1	7.3	7.3
3	5	1	8.8	8.8
4	6,10	1	8.7	2.5
5	7,11	1	9.1	5.5
6	13	1	5.9	5.9
7	4,8	2	15.0	17.0
8	9	1	6.6	6.6
9	**12**	1	7.1	7.1
10	18	1	9.4	9.4
11	15,17	1	9.2	5.5
12	20	1	0.0	9.0
13	19,21,22	1	8.0	8.0
14	23	1	9.6	8.2
15	24,25	2	16.6	3.7

→ task 14: E_{14}=8; L_{14}=13 assigned to station 8

K	Tasks	R	W_A	W_B
1	1,2	1	7.7	7.7
2	3	1	7.3	7.3
3	5	1	8.8	8.8
4	6,10	1	8.7	2.5
5	7,11	1	9.1	5.5
6	13	1	5.9	5.9
7	4,8	2	15.0	17.0
8	9,**14**	1	7.9	6.6
9	12	1	7.1	7.1
10	18	1	9.4	9.4
11	15,17	1	9.2	5.5
12	20	1	0.0	9.0
13	19,21,22	1	8.0	8.0
14	23	1	9.6	8.2
15	24,25	2	16.6	3.7

Figure 4.8 – An application of the reassignment procedure

A mutation operator, which randomly disturbs genetic information, performing small changes in a single parent in order to produce a new offspring, is also included. Considering the structure of the problem and the encoding of solutions, the most adequate mutation operator is the one used by Kim et al (1996, 2000). This operator was adapted to the characteristics of the addressed problem and it works as follows. A parent is selected to undergo mutation, according to a mutation probability, and a small set of tasks is randomly selected. These tasks will be reassigned applying the reassignment procedure earlier described and a new offspring is created. The mutation probability is set by default to 0.02, a typical value used in this technique (Leu et al, 1994, Sabuncuoglu et al, 2000) and the number of tasks involved in mutation is, at maximum, 10% of the total number of tasks in the combined precedence diagram.

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. Comparing each offspring with one of its parents, the offspring always replace the parent except when the fitness value of the offspring is lower than the worst fitness value of the individuals in the previous generations – in this case, the probability of the parent to continue in the population is set to a high value (0.8 by default). In order to always keep the best individual found so far, the individual in the new population with the lowest fitness is replaced by the individual from the previous generation with the highest fitness.

4.4.4 Stopping criteria

To determine the stopping criteria of the procedure, a simple convergence study was performed for each of the tested problems. Figure 4.9 shows the variation of the fitness function (setting λ=10) in five runs of two test problems, one with 25 tasks and two models and another with 70 tasks and three models. The leap from the lower level to the upper level is due to the reduction of the number of operators in the best balancing solution. Further increases are due to the improvement in the workload balance. The value of the fitness function remained unchanged after the 20^{th} and 90^{th} iterations (populations of individuals) for the first and second problems, respectively. Before this scenario the decision was to select over-engineered parameters, meaning that a greater amount of time

will be spent running the procedure ensuring that a good solution is found, instead of spending a smaller amount of time at a cost of the solution quality.

A trade-off between convergence and execution time is defined as the stopping criterion. This is a popular criterion used in GA based approaches (Leu et al, 1994). The procedure will stop when one of the following conditions is achieved:

(i) the fitness function of the best solution does not improve more than 1% after a pre-determined number of consecutive iterations (this value is set to 50 by default);

(ii) the total number of iterations exceeds a maximum number (200 is the value set by default).

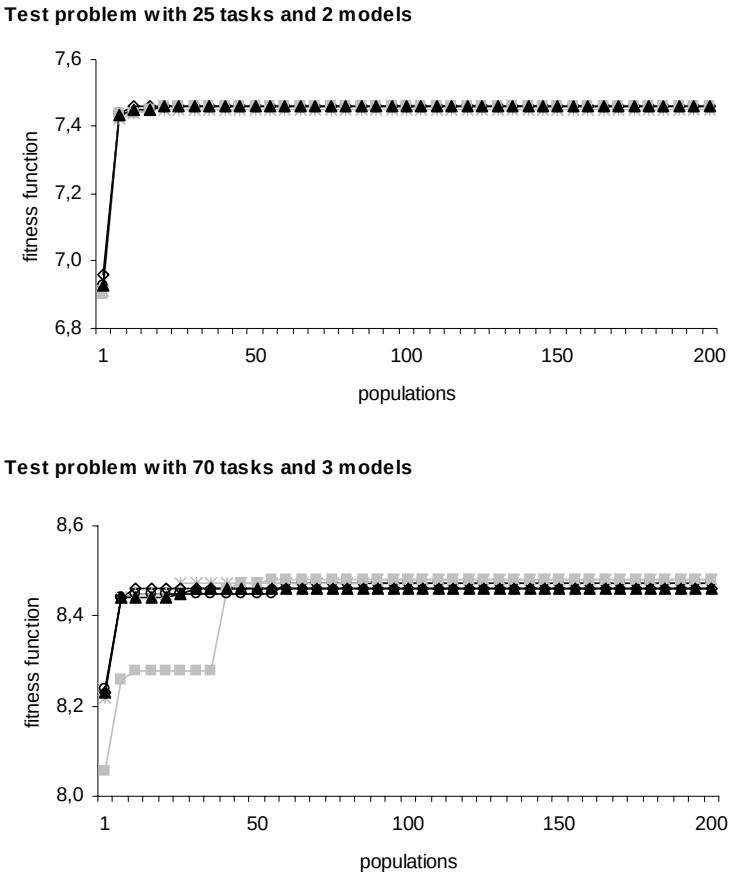

Figure 4.9 – Variation of the fitness function in GA for two test problems

4.5 Ant colony optimisation based approach

The ant colony optimisation (ACO) based approach developed to address the MALBP-I was named ANTBAL. The initial version of ANTBAL showed a bad performance, so a modification was implemented in order to correct the observed problems. In the next section the initial version of ANTBAL will be described and in section 4.5.2 the modifications made to improve it will be reported.

4.5.1 Initial version of ANTBAL

The outline of the initial version of ANTBAL is shown in Figure 4.10.

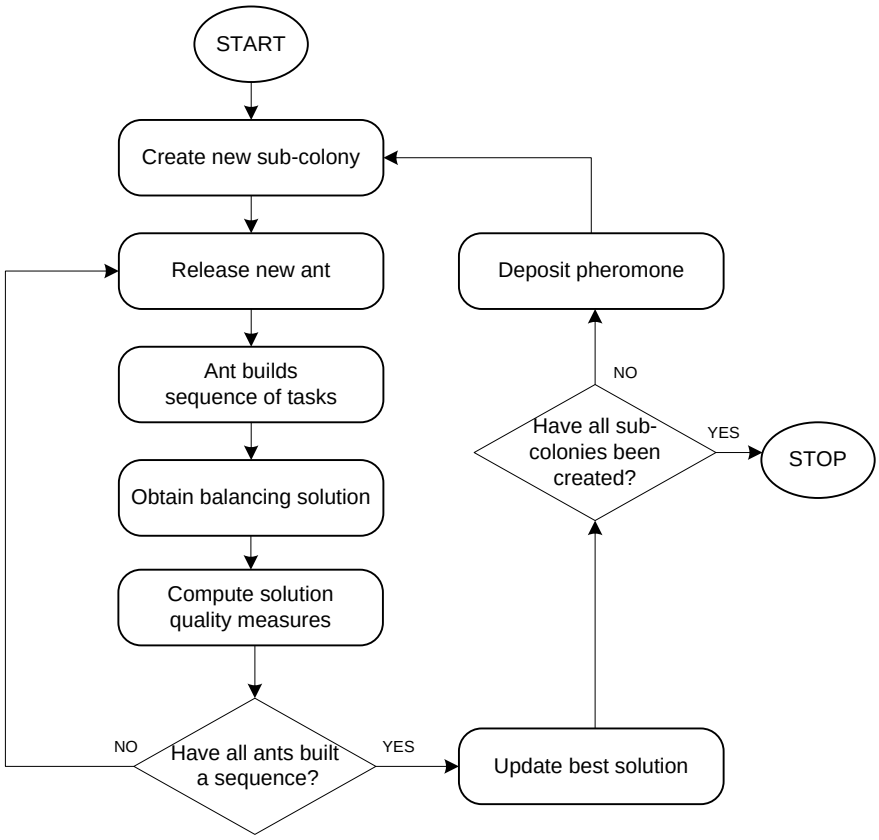

Figure 4.10 – Outline of the first version of ANTBAL

In ANTBAL, the mission of an ant is to analyse the precedence diagram of the tasks required to assemble a given product and build a sequence according to which the tasks will be performed. After the sequence of tasks is completed, a procedure is applied in order to turn the sequence into a feasible balancing solution, taking into account the problem's

capacity and zoning constraints. For each solution obtained, quality measures are computed, according to the defined goals.

In each sub-colony there are N_A ants. After all ants of a sub-colony have generated a solution, they release an amount of pheromone according to its solution quality. Pheromone trails are kept in a matrix *task*×*task*. If task *j* is selected to join the sequence immediately after task *i*, then an amount of pheromone is released between task *i* and task *j*. This way, pheromone trails exist in the paths that ants used to build the whole sequence.

The procedure is repeated for every sub-colony within the ant colony. The best solution found by the procedure is updated after each sub-colony iteration. In the following sections the main features of ANTBAL will be described in more detail.

4.5.1.1 How does an ant build a sequence of tasks?

The sequence of tasks must be feasible in terms of the precedence constraints, so it is built according to the combined precedence diagram. Each ant has access to a list of available tasks that it can choose from to include in the sequence. A task is considered available if it has no predecessors or if all its predecessors are already in the task sequence. The probability of selecting a task, from the list of available tasks, is a function of the pheromone trail intensity between the previously selected task and each available task and each available task's heuristic information.

ACO algorithms are based on the behaviour of real ants but they also provide artificial ants with additional skills that make them more effective. For example, to address the travelling salesman problem, the selection of the cities of the tour uses both pheromone trails and the known distance between the cities, additional information that a real ant would not own. In assembly line balancing problems the additional information about the problem, called heuristic information, is usually given by priority rules.

When the ants of a sub-colony are generated, different priority rules are assigned to them. This way, while an ant is building its sequence, the heuristic information of a task is simply the priority rule value known by the ant. The procedure uses the priority rules also used in the genetic algorithms based procedure, namely: (i) maximum processing time for all models, (ii) maximum average processing time, (iii) maximum ranked positional

weight, (iv) maximum number of direct successors and (v) maximum total number of successors.

The probability with which an ant *n* selects task *j* after it had selected task *i* is given by:

$$p^{n}{}_{(i,j)} = \frac{\left[\tau_{(i,j)}\right]^{\alpha} \cdot \left[\eta_{j}\right]^{\beta}}{\sum_{u \in A_i^n} \left[\tau_{(i,u)}\right]^{\alpha} \cdot \left[\eta_{u}\right]^{\beta}} \qquad \textbf{(4.23)}$$

where $\tau_{(i,j)}$ is the pheromone trail intensity in the path 'selecting task *j* after selecting task *i*', η_j is the heuristic information of task *j* (i.e., the priority rule value for task *j*), A_i^n is the set of available tasks for ant *n* after the selection of task *i* and α and β are parameters that determine the relative importance of pheromone intensity versus heuristic information. At each iteration, a random number is generated and a task is selected according to its probability.

4.5.1.2 Procedure to obtain a balancing solution

When a sequence of tasks is completed, it is necessary to convert it into a feasible balancing solution. Tasks are assigned to workstations exactly by their order in the sequence. The proposed procedure, whose structure is shown in Figure 4.11, allows the generation of solutions to the MALBP-I with the characteristics described in section 4.2.

A task is assigned to a workstation if and only if the resulting assignment verifies both zoning and capacity constraints, as the precedence constraints are guaranteed by the sequence already built by the ant. If a problem has positive zoning constraints the tasks that need to be allocated to the same workstation are merged previously and treated as only one task. This is done in the precedence diagram, prior to the start of ANTBAL. Negative zoning constraints are handled while building the balancing solution, as we can see in Figure 4.11. If a task is to be assigned in a workstation where there is already a task with which it is incompatible, then, the current workstation is closed and the task is assigned to a new workstation.

When assigning a task from the sequence built by the ant, capacity constraints, as described in by equations (4.8), (4.9) and (4.10) are taken into account. If the assignment of the task violates capacity constraints, then, the task is not assigned to the current

workstation and a new one is opened. When all tasks in the sequence have been assigned to workstations, the balancing solution is completed and solution quality measures are computed, as described in the following section.

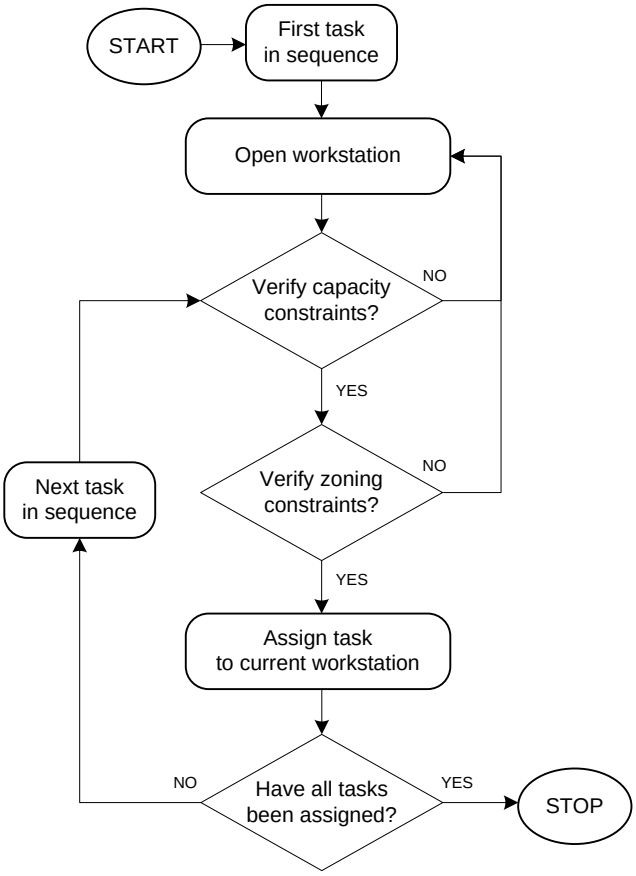

Figure 4.11 – Procedure to convert a sequence of tasks into a balancing solution

4.5.1.3 Solution quality

The objective function used in ANTBAL is the one of the mathematical programming model presented in Figure 4.2, i.e., the maximisation of $Z=\lambda WE-B_b-B_w$. The selection of this particular expression was due to the fact that, typically, in ACO approaches, the amount of pheromone released by the ants depends on the quality of the corresponding solution. In order to ease the pheromone amount calculation process, it was decided to use exactly the same value of the objective function. This way, the criterion had to be maximisation, because, the better the solution, the higher the pheromone trail. Also, the range of values of Z would not depend on the problem instance.

4.5.1.4 Pheromone release strategy

The pheromone release strategy is based on the one used by Dorigo et al (1996). At the beginning of the procedure, an initial amount of pheromone (τ_0) is released in every path, i.e., between every pair of tasks. At the end of each sub-colony iteration, all balancing solutions provided by the ants have their objective function values computed. It is at this point that the pheromone trail intensity is updated. First, a portion of the existing pheromone value is evaporated in all paths, according to:

$$\tau_{(i,j)} \leftarrow (1-\rho)\cdot\tau_{(i,j)} \qquad \textbf{(4.24)}$$

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

$$\Delta\tau^{n}{}_{(i,j)} = \begin{cases} Z, \text{if in the solution built by ant } n \text{ task } j \text{ is performed immediatly after task } i \\ 0, \text{otherwise} \end{cases} \qquad \textbf{(4.25)}$$

The overall pheromone update effect of all ants in each path *(i,j)* is then:

$$\tau_{(i,j)} \leftarrow \tau_{(i,j)} + \sum_{n=1}^{N_A} \Delta\tau^{n}{}_{(i,j)} \qquad \textbf{(4.26)}$$

4.5.2 Modifications of ANTBAL

4.5.2.1 Problems with the initial version of ANTBAL

To test the performance of ANTBAL, a set of instances of MALBP-I was solved and the results were compared with the results already obtained using the simulated annealing procedure, described in section 4.3. The results showed that the number of operators of the solutions obtained with ANTBAL was, for almost every instance, higher that the ones obtained using the simulated annealing procedure, which indicated a very bad performance of ANTBAL. In order to understand the causes of this performance, an analysis to the algorithm was made and, rapidly, the reasons were found.

The problem with ANTBAL was the fact that many workstations would have unnecessary idle time, in order to preserve, in the balancing solution, the sequence built by the ants. If a task did not fit the current workstation, the procedure would close the current workstation and open a new one to assign that task. The procedure did not allow other tasks, forward in the sequence, to be assigned to the current workstation, even if they would verify the capacity constraints. An illustration of such a situation is presented in Figure 4.12.

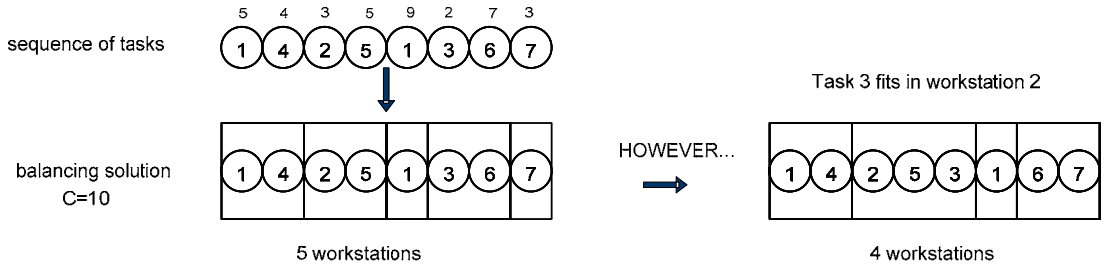

Figure 4.12 – Problems with the initial version of ANTBAL

The existence of a rigid task sequence to keep was providing very bad balancing solutions, so, to tackle this problem, the role of the ants was modified, as it is explained in the following section.

4.5.2.2 New role of the ants

Instead of just making a sequence of tasks, the new role of the ants is to build a complete balancing solution. The structure of modified version of ANTBAL is presented in Figure 4.13.

Each ant in the sub-colony builds a feasible balancing solution, i.e., an assignment of tasks to workstations that satisfies precedence, zoning and capacity constraints. For each feasible solution obtained, a measure of its quality is computed, according to the problem's objective function.

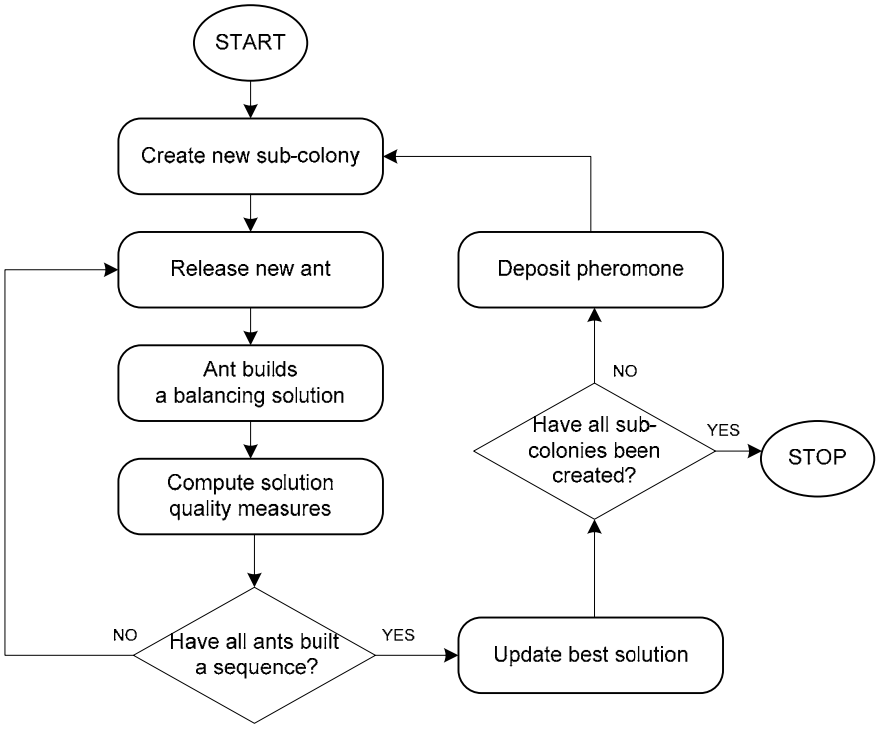

Figure 4.13 – Outline of the modified version of ANTBAL

4.5.2.3 New pheromone release strategy

After all ants of a sub-colony have generated a solution, they release a certain amount of pheromone according to the quality of the solution. The characteristic of a balancing solution that make it better or worse than another is the assignment of tasks to workstations. In straight assembly lines, the sequence in which tasks are performed within a workstation is not relevant, as long as it meets precedence constraints. So, for this particular problem, it was considered more adequate to keep pheromone trails in the assignment of tasks to workstations than between consecutive tasks.

In the new version of ANTBAL pheromone trails are kept in a matrix *workstation×task*: if task *j* is assigned to workstation *i*, then a certain amount of pheromone is released between workstation *i* and task *j*. An initial amount of pheromone (τ_0) is released in every path, i.e., between every pair *workstation-task*. At the end of each sub-colony iteration, the pheromone trail intensity is updated. First, a portion of the existing pheromone value is evaporated in all paths, according to:

$$\tau_{(i,j)} \leftarrow (1-\rho)\cdot\tau_{(i,j)} \quad \textbf{(4.27)}$$

where ρ is the evaporation coefficient ($0 \leq \rho \leq 1$). Then, each ant n releases an amount of pheromone in the assignments of tasks to workstation that it has made, according to the corresponding balancing solution quality. This amount of pheromone is given by:

$$\Delta\tau^{n}_{(i,j)} = \begin{cases} Z, \text{if in the solution built by ant } n \text{ task } j \text{ is assigned to workstation } i \\ 0, \text{otherwise} \end{cases} \quad \textbf{(4.28)}$$

The total pheromone level in the assignment of task j to workstation i (τ_{ij}) is then:

$$\tau_{(i,j)} \leftarrow \tau_{(i,j)} + \sum_{n=1}^{N_A} \Delta\tau^{n}_{(i,j)} \quad \textbf{(4.29)}$$

4.5.2.4 Building a balancing solution

The procedure carried out by each ant to build a feasible balancing solution is depicted in Figure 4.14. An ant begins by determining the available tasks for assignment to the current workstation, taking into account the problems constraints (precedence, zoning and capacity). Then, from the set of available tasks, selects one of these tasks. When there are no available tasks to assign to the current workstation, a new workstation is opened. This procedure is repeated until all the tasks have been assigned.

The procedure for selecting a task for assignment was also modified, in order to better guide the search of the solution space. The probability of a task being selected, from the set of available tasks, is a function of (i) the pheromone trail intensity between the current workstation and each available task and (ii) the information provided by the heuristic for each available task. This information is a priority rule that is randomly assigned to each ant when the respective sub-colony is generated. The procedure uses the same static priority rules of the initial version and a new dynamic called 'last task becoming available', especially developed for this algorithm and which deals with the work relatedness issue. Related tasks are directly connected in the precedence diagram and a common procedure used by assembly line managers is to assign them to the same workstations, in order to improve work efficiency. Therefore, this rule aims to favour the assignment of the direct

successors of a task immediately after that task has been assigned, by attributing them the highest priority value in the subsequent assignment iteration.

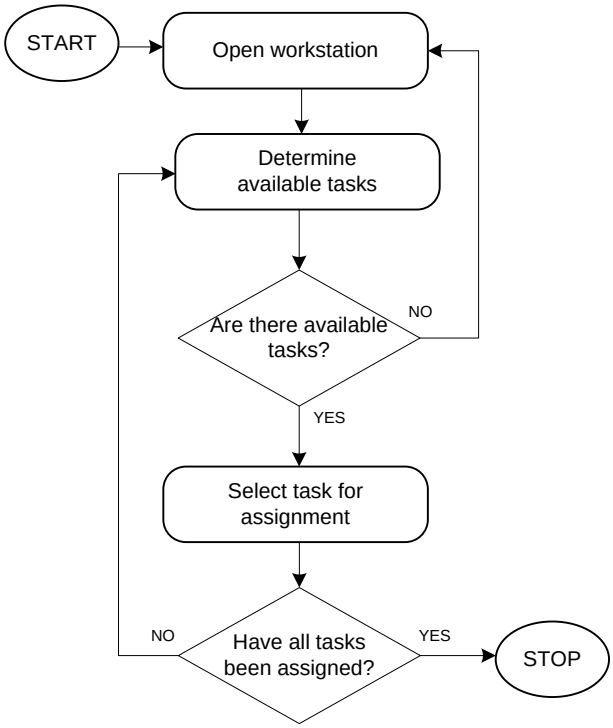

Figure 4.14 – Procedure carried out by an ant to build a feasible solution

The values of the priority rules will vary between 1 for the task with lowest priority and N (number of tasks) for the task with highest priority, and will be the heuristic information used by the ants to select the tasks.

Let r be a random number between 0 and 1 and r_1, r_2 and r_3 three user-defined parameters such that $0 \le r_1, r_2, r_3 \le 1$ and $r_1 + r_2 + r_3 = 1$. An ant n will select task j to be assigned to the current workstation i by applying the following rule:

$$j = \begin{cases} J_1 = \arg\max_{j \in A_i^n} \left\{ [\tau_{(i,j)}]^\alpha [\eta_j]^\beta \right\} & \text{if } r \le r_1 \quad (\textit{exploitation}) \\ J_2 : p_{(i,J_2)} = \dfrac{[\tau_{(i,J_2)}]^\alpha [\eta_{J_2}]^\beta}{\sum_{j \in A_i^n} [\tau_{(i,j)}]^\alpha [\eta_j]^\beta} & \text{if } r_1 \le r \le r_2 \quad (\textit{biased exploration}) \\ J_3 : \text{random selection of } j \in A_i^n & \text{if } r_2 \le r \le r_3 \quad (\textit{random selection}) \end{cases} \qquad \textbf{(4.30)}$$

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

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

- Exploitation: it determines the selection of the best task according to the values of $[\tau_{(i,j)}]^{\alpha}[\eta_j]^{\beta}$.
- Biased exploration: a task is selected with a probability of $p_{(i,j)}$ as given by J_2 in equation (4.30).
- Random selection: from the set of available tasks, the ant selects one at random.

The first two strategies are based on the Ant Colony System state transition rule proposed by Dorigo and Gambardella (1997). After the task is selected, the ant assigns it to the current workstation. When all tasks have been assigned to workstations, the balancing solution is completed and solution quality measures are computed, as described in the initial version.

4.5.2.5 Parameter settings

The following values of the numeric parameters used in ANTBAL were obtained by a set of experimental tests:

- Initial pheromone level: According to Dorigo and Gambardella (1997) a rough approximation of the optimal value of the objective function is a reasonable value for τ_0. A perfectly balanced line would have an efficiency of 100% (setting λ=10, λWE=10) and equally distributed workloads (B_b=0 and B_w=0). Considering that such a situation would hardly occur in a real-world assembly line, the value of τ_0 is set, by default, to 9.0.
- Pheromone evaporation coefficient: ρ = 0.2.
- Relative importance of pheromone intensity versus heuristic information: α=0.2, β=1.0 (it was observed that higher values of α lead to premature convergence of the algorithm).

- Task selection strategy: r_1=0.6, r_2=0.3, r_3=0.1.

To determine the total number of iterations of the algorithm, a simple convergence study was carried out for each of the tested problems. Figure 4.15 shows the variation of the objective function value of the best solution in five runs of two test problems. The value of the objective function did not improve after the 150th and 120th sub-colonies for the first and second problems, respectively. Before this scenario, and to ensure a good solution is found, ANTBAL will have 200 sub-colonies with 50 ants each.

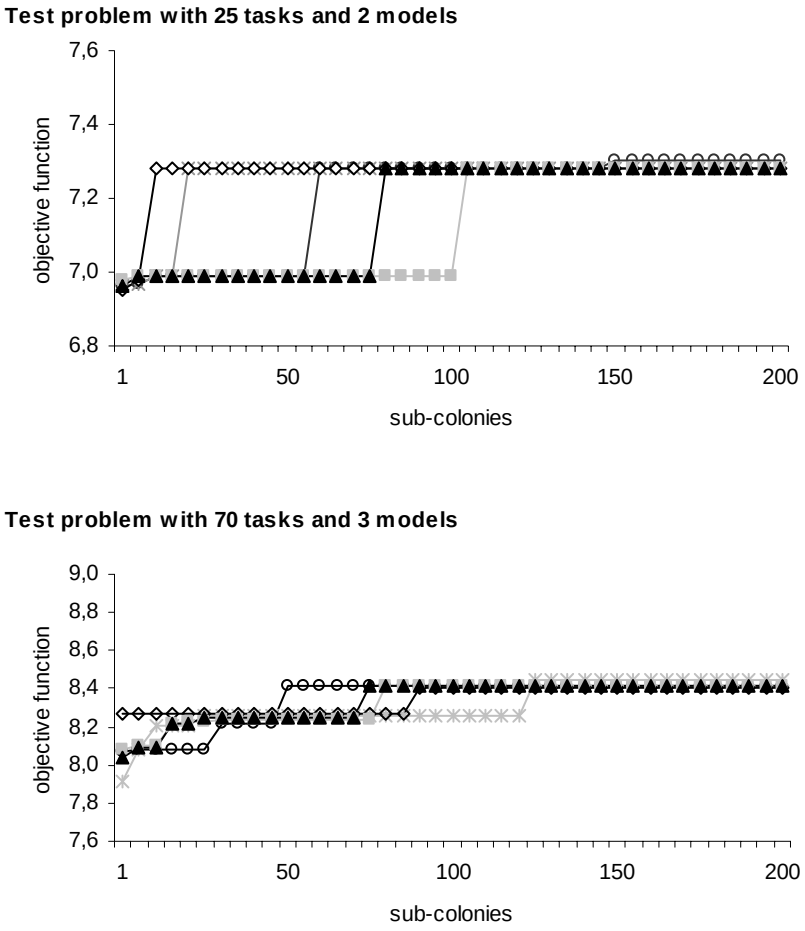

Figure 4.15 – Variation of the objective function in ANTBAL for two test problems

4.6 Addressing the problem of type II

Assembly line balancing problems of type II deal with the maximisation of the production rate of an existing assembly line, i.e., the goal is to minimise the cycle time of the line for a given number of operators. While type I problems are more frequently used in the design of a new assembly line for which the demand can be easily forecasted and consequently, the production rate, has to be pre-specified, problems of type II are applied when, for example, changes in the assembly process or in the product range require the line to be redesigned.

The research work on the assembly line balancing problem of type II has been devoted, almost exclusively, to single-model lines.

Some of the methods proposed to solve the single-model version of the assembly line balancing problem of type II (SALBP-II) explore the duality relationship between type I and type II problems, and a solution for the SALBP-II is found by iteratively solving type I problems for several trial cycle times, in order to check if a feasible assignment of a pre-determined number of workstations exists. Heuristic procedures that use this strategy are proposed by, for example, Hackman et al (1989), Rachamadugu and Talbot (1991) and Scholl and Voβ (1996).

Meta-heuristics have also been proposed to solve the SALBP-II. Genetic algorithms are used by Anderson and Ferris (1994) and Kim et al (1998), both aiming to smooth the workload between the specified number of workstations and hence minimise the cycle time. Scholl and Voβ (1996) developed a taboo search procedure with the goal of improving an initial feasible solution through shift and swap movements.

Klein and Scholl (1996) propose SALOME-2, an optimising approach based on the branch-and-bound method, which directly solves the SALBP-II.

Liu et al (2003) propose two bi-directional heuristic procedures to minimise both cycle time and the mean absolute deviation of workloads, but only for single-model assembly lines.

4.6.1 First approach

The first approach developed to address the mixed-model assembly line balancing problem of type II (MALBP-II) is outlined in Figure 4.16. It iteratively solves problems of type I for different cycle times. It starts by computing a lower bound for the value of the cycle time and then it uses this value to solve a problem of type I, using an appropriate procedure, as the ones described in sections 4.3, 4.4 and 4.5.

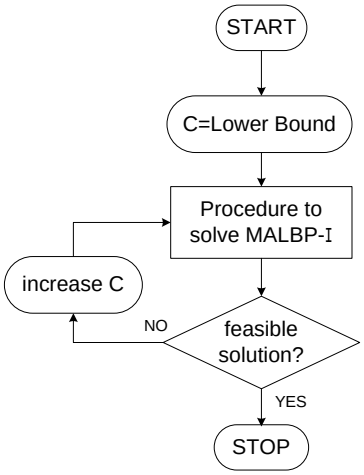

Figure 4.16 – First approach to address MALBP-II

A straightforward lower bound for the cycle time in a MALBP-II can be computed from the ratio between the sum of the task processing times and the pre-defined number of operators. But, for the problem described in section 4.2, this lower bound can be fine-tuned taking into account the value of *MRT*. As no task with a processing time higher than *MRT* can be processed in a non-replicated workstation, *MRT* can improve the lower bound for the cycle time defined above. So the lower bound for the cycle time is given by:

$$LB = \max_{m=1,\ldots,M} \left\{ \sum_{i=1}^{N} t_{im} \Big/ S \, , MRT \right\} \qquad \textbf{(4.31)}$$

If the total number of operators derived from solving the MALBP-I is greater than the preset number of operators for the original MALBP-II problem (i.e., if it is a non-feasible solution), the cycle time is increased by a problem specific increasing unit and another MALBP-I is solved. This procedure continues until a solution with the preset number of

operators, *S*, is found. An application of the genetic algorithm based procedure to solve the MALBP-II was presented in Simaria and Vilarinho (2004).

4.6.2 Second approach

Although the computational experiments to test the performance of this approach provided good results, it was observed that, in some cases, the value of the cycle time could still be improved. As the approach would stop when the pre-specified number of operators was reach it did not attempt to improve solutions with the same number of operators. This way, a second approach to address the MALBP-II was developed. It adds to the first approach a simulated annealing (SA) smoothing procedure that aims to perform swapping and transferring of tasks between workstations, in order to try to decrease the cycle time of the line configuration. The outline of this procedure is presented in Figure 4.17.

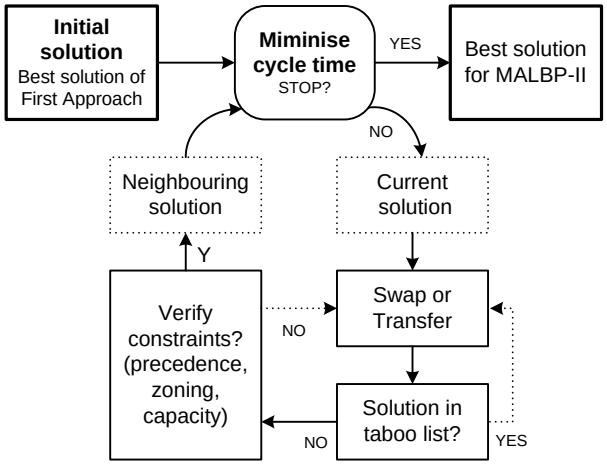

Figure 4.17 – SA smoothing procedure for the MALBP-II

The initial solution of the SA procedure is the best solution obtained by the first approach, which iteratively solves problems of type I until a solution with the required number of operators is obtained. The objective function to minimise is the real cycle time of the line, i.e., the maximum workload observed among the workstations, given by:

$$C = \max_{k=1,\ldots,K}\left[\max_{m=1,\ldots,M}\left\{W_{km}/R_k\right\}\right] \quad \textbf{(4.32)}$$

where W_{km} is the workload of workstation k due to model m and R_k is the number of replicas of workstation k.

A neighbouring solution can be obtained by one of the following actions:

- Random swap: two tasks are randomly selected and their positions are swapped.
- Random transfer: one task is randomly selected and transferred to a randomly selected workstation.
- 'Intelligent' swap: selects one task from the workstation with maximum workload and swaps it with another task with inferior processing time.
- 'Intelligent' transfer: selects one task from the workstation with maximum workload and transfers it to another workstation.

'Intelligent' movements are more likely to contribute to the goal of the procedure, so, higher probabilities should be set to this type of actions. However, if after a predefined number of attempts neither 'intelligent' swap or transfer movements lead to a neighbouring solution, due to the constraints of the problem, random movements will be performed.

The annealing schedule defined for this procedure is similar to the one described in section 4.3, as the characteristics of the addressed problems are similar.

4.7 Computational experience

4.7.1 Type I

The procedures described in this chapter were coded in C and run on a 2.8 GHz Pentium 4 computer. The performance of the three meta-heuristic approaches was compared using a set of 20 mixed-model assembly line balancing problems with parallel workstations and zoning constraints, whose main characteristics are presented in Table 4.2, namely, the number of tasks of the combined precedence (N), the number of models (M), the sum of task times (in time units, t.u.) for each model ($Sum\ t_i$) and the production share of each model (q_m).

Table 4.2 – Main characteristics of the MALBP data set with typical task times

			1st model		2nd model		3rd model	
Problem	**N**	**M**	**Sum t_i**	**q_m**	**Sum t_i**	**q_m**	**Sum t_i**	**q_m**
1	8	2	35.8	0.42	33.1	0.58	-	-
2	8	3	34.5	0.33	47.0	0.50	53.1	0.17
3	11	2	60.3	0.42	42.0	0.58	-	-
4	11	3	46.0	0.33	57.3	0.50	57.3	0.17
5	21	2	130.1	0.42	106.0	0.58	-	-
6	21	3	116.2	0.33	119.6	0.50	124.8	0.17
7	25	2	132.2	0.42	116.2	0.58	-	-
8	25	3	114.8	0.33	126.8	0.50	127.4	0.17
9	28	2	176.9	0.42	185.0	0.58	-	-
10	28	3	162.5	0.33	166.6	0.50	173.4	0.17
11	30	2	140.9	0.42	139.9	0.58	-	-
12	30	3	164.8	0.33	157.2	0.50	169.0	0.17
13	32	2	135.5	0.42	155.0	0.58	-	-
14	32	3	160.9	0.33	147.0	0.50	161.0	0.17
15	35	2	193.6	0.42	190.8	0.58	-	-
16	35	3	200.0	0.33	206.2	0.50	208.6	0.17
17	45	2	221.8	0.42	210.4	0.58	-	-
18	45	3	230.0	0.33	235.3	0.50	212.0	0.17
19	70	2	372.8	0.42	389.8	0.58	-	-
20	70	3	375.8	0.33	384.3	0.50	376.6	0.17

The precedence diagrams used for the test problems were taken from Scholl (1993), except for problems 7 and 8, where the one shown in Figure 4.4 was used. The task processing times for each problem were randomly generated taking into account the different task types that might be present in a real world mixed-model assembly process, in which the processing time of a task may vary from model to model but within certain limits. They will be called 'typical' task times. Considering t_{im} the processing time of task i for model m and C the cycle time of the line, the generation of the task processing times uses the following rules:

(i) Task i is
- performed in the first model (m=1) with t_{i1}>0, which can be higher, equal or lower than C or
- not performed for the first model (t_{i1}=0).

(ii) The processing time of task i for the other models is

- equal to t_{i1},
- higher or lower than t_{i1} within pre-specified limits ($(1-\delta)t_{i1} \leq t_{im} \leq (1+\delta)\ t_{i1}$, $0 \leq \delta \leq 1$) or
- null.

For every problem of type I, a cycle time of 10 t.u. was considered. The full details of the test problems are provided in Appendix 2.

The test problems were solved using the three proposed procedures: simulated annealing (SA), genetic algorithms (GA) and the ant colony optimisation algorithm (ANTBAL). For this set of problems the number of operators (S) of the solutions provided by each procedure was compared with the lower bound of the total number of operators (LB_{pmix}), especially developed for this type of problems and whose details are presented in Appendix 3. The values shown in Table 4.3 are the best of ten runs, however the observed variance of the results was nearly null.

Table 4.3 – Computational results for the MALBP-I data set

			SA		GA		ANTBAL		Best
Problem	**Opt**	**LB_{pmix}**	**S**	**D(%)**	**S**	**D(%)**	**S**	**D(%)**	**WE (%)**
1	4	4	4	0	4	0	4	0	85.6
2	8	6	8	0	8	0	8	0	54.9
3	7	7	7	0	7	0	7	0	71.0
4	7	6	7	0	7	0	7	0	76.5
5	-	14	16	14.3	16	14.3	16	14.3	72.6
6	-	13	15	15.4	15	15.4	15	15.4	79.6
7	-	14	16	14.3	16	14.3	16	14.3	76.8
8	-	14	15	7.1	14	0	14	0	87.9
9	-	19	21	10.5	20	5.3	20	5.3	90.8
10	-	18	20	11.1	20	11.1	20	11.1	83.2
11	-	15	16	6.7	16	6.7	16	6.7	86.6
12	-	17	19	11.8	19	11.8	19	11.8	83.4
13	-	16	19	18.8	19	18.8	19	18.8	77.3
14	-	17	19	11.8	19	11.8	19	11.8	81.0
15	-	20	24	20.0	23	15.0	23	15.0	83.5
16	-	21	24	14.3	24	14.3	24	14.3	85.2
17	-	23	25	8.7	24	4.3	24	8.7	85.4
18	-	24	28	16.7	27	12.5	26	8.3	84.4
19	-	41	44	7.3	43	4.9	43	4.9	87.0
20	-	39	44	12.8	44	12.8	44	12.8	86.0

For problems 1 to 4 the optimal solution is known – it was obtained by solving the mathematical programming model using the CPLEX (1999) optimiser. The difference between the solutions obtained by each procedure and the lower bound (or the optimal solution) is depicted in the correspondent column *D*(%) and it is computed as follows:

$$D(\%) = \frac{S - LB_{pmix}}{LB_{pmix}} \times 100 \qquad \textbf{(4.33)}$$

Some conclusions can be drawn from the results of this experience. Considering the number of total operators, GA and ANTBAL outperformed the SA procedure. They improved the solution in five problem instances (problems 8, 15, 17, 18 and 19), reaching the lower bound in problem 8, thus guaranteeing the optimum. ANTBAL improved the solution provided by GA in problem 18. While the GA procedure found a minimum of 27 operators, ANTBAL was able to find a line configuration with 26 operators. So, ANTBAL was the best procedure for this computational experience. The worst performance of this heuristic was for problem 13, where the difference between the best solution obtained and the lower bound is 18.8%. However, as the calculation of the lower bound does not take into account the precedence and zoning constraints, one is lead to consider that the results are fairly good. This conclusion is reinforced by the values for the line efficiency shown in column *WE*(%), where a high line usage rate can be perceived, particularly for the largest sized problems.

Considering the average computational time, all procedures are similar for small and medium sized problems. For large sized problems GA and ANTBAL are slower than SA. This is explained by the fact that the number of solutions generated by these in each iteration is much higher that in the SA procedure. However, the maximum computational time was around 2 minutes, a perfectly acceptable value, considering the strategic nature of the problem under analysis.

Another set of computational experiments was conducted using the MALBP data set but with different task processing times. The task times used for this experience were randomly generated. In order to allow the creation of parallel workstations with a minimum replication time of *MRT*=*C*, each task processing time was randomly generated between the limits [0,2*C*], where *C* is 10 t.u.. Table 4.4 presents the values of the sum of

task times and production share per model for each problem instance. The full details are provided in Appendix 2.

Table 4.4 – MALBP data set with random processing times

			1st model		2nd model		3rd model	
Problem	**N**	**M**	**Sum t_i**	**q_m**	**Sum t_i**	**q_m**	**Sum t_i**	**q_m**
1	8	2	66.9	0.42	96.5	0.58	-	-
2	8	3	69.0	0.33	58.8	0.50	56.1	0.17
3	11	2	59.0	0.42	85.9	0.58	-	-
4	11	3	83.3	0.33	85.9	0.50	120.4	0.17
5	21	2	220.7	0.42	167.6	0.58	-	-
6	21	3	250.4	0.33	176.4	0.50	147.0	0.17
7	25	2	296.1	0.42	257.0	0.58	-	-
8	25	3	283.9	0.33	277.9	0.50	192.3	0.17
9	28	2	284.5	0.42	267.0	0.58	-	-
10	28	3	248.1	0.33	262.9	0.50	257.6	0.17
11	30	2	303.2	0.42	269.9	0.58	-	-
12	30	3	302.0	0.33	312.3	0.50	342.3	0.17
13	32	2	299.7	0.42	341.1	0.58	-	-
14	32	3	296.3	0.33	342.2	0.50	343.6	0.17
15	35	2	343.9	0.42	358.7	0.58	-	-
16	35	3	291.8	0.33	350.9	0.50	395.4	0.17
17	45	2	423.8	0.42	485.7	0.58	-	-
18	45	3	489.8	0.33	508.3	0.50	470.8	0.17
19	70	2	683.3	0.42	643.3	0.58	-	-
20	70	3	705.0	0.33	638.2	0.50	750.1	0.17

Once again the test problems were solved using the three meta-heuristic based procedures and the best results of ten runs, for each procedure and problem instance, are presented in Table 4.5. The outcome of this experiment confirmed the conclusions of the previous one. For five problems both GA and ANTBAL improved the SA solutions in one or more operators (for problem 20 the improvement was of six operators). Comparing GA with ANTBAL, the following comments can be made:

(i) For one instance (problem 4) GA equalised the solution of SA while ANTBAL was able to improve it.

(ii) For one instance (problem 19) ANTBAL equalised the solution of SA while GA was able to improve it.

(iii) For one instance (problem 16) GA improved the solution of SA and ANTBAL improved the solution of GA.

Table 4.5 – Computational results for the MALBP-I data set with random times

		SA		GA		ANTBAL		Best
Problem	**LB_{pmix}**	**S**	**D(%)**	**S**	**D(%)**	**S**	**D(%)**	**WE (%)**
1	11	11	0.0	11	0.0	11	0.0	76.4
2	7	11	57.1	11	57.1	11	57.1	56.1
3	9	11	22.2	11	22.2	11	22.2	67.8
4	14	17	21.4	17	21.4	16	14.3	56.8
5	26	29	11.5	29	11.5	29	11.5	65.5
6	28	35	25.0	35	25.0	35	25.0	55.9
7	34	40	17.6	40	17.6	40	17.6	68.4
8	36	40	11.1	40	11.1	40	11.1	66.3
9	30	37	23.3	35	16.7	35	16.7	78.4
10	28	34	21.4	34	21.4	34	21.4	75.6
11	36	39	8.3	38	5.6	38	5.6	74.7
12	40	50	25.0	50	25.0	50	25.0	62.8
13	35	50	42.9	50	42.9	50	42.9	64.7
14	36	54	50.0	54	50.0	54	50.0	60.6
15	38	47	23.7	47	23.7	47	23.7	75.0
16	41	54	31.7	53	29.3	52	26.8	65.2
17	50	60	20.0	59	18.0	59	18.0	77.9
18	56	78	39.3	78	39.3	78	39.3	63.6
19	69	89	29.0	88	27.5	89	29.0	75.0
20	80	110	37.5	104	30.0	104	30.0	65.3

The best efficiency values (*WE*) were considerably lower than the one obtained in the first computational experiment. This is explained by the nature of the task processing times of the problem instances. While in the first data set there was a high number of short tasks (when compared with the value of the cycle time), which allowed a better combination of tasks within the workstations, in the second data set the generation of task times was completely random, making it high the number of long tasks. When building a balancing solution, the procedures create the workstations with high idle times, as it is more difficult to combine tasks. Idle times cause low efficiency of the assembly line.

To extend the computational experience, a series of comparative tests were carried out by adapting the three procedures to the conditions under which the benchmark problems

proposed by Scholl (1993) were originally set. The number of tasks of the problems from this data set ranges from 25 to 297. Scholl's test problems are for single-model balancing problems without parallel workstations, so the heuristics were run setting *MRT* to a value higher than the longest task processing time, in order to prevent from the creation of parallel workstations. For the 168 instances analysed the optimal solution is known for 166. Table 4.6 summarises the number of optimal solutions obtained by each procedure and the maximum deviation from the optimal solution.

Table 4.6 – Number of optimal solutions and maximum deviation obtained for Scholl's data set

	SA	**GA**	**ANTBAL**
Number of optimal solutions	73	97	97
Maximum deviation from optimal	14%	14%	14%

ANTBAL and GA clearly outperformed the SA procedure, considering the number of optimal solutions found for this data set. However, for all the procedures the maximum deviation from the optimal solutions was only 14% and it occurred in the same problem instance: the optimal solution had 7 operators and the best solution obtained by the three procedures had 8 operators. The performance of ANTBAL and GA was similar as both procedures provided solutions with the same number of operators for every problem instance.

This set of computational experiments showed that the overall performance of ANTBAL and GA is superior to the SA heuristic. The results of ANTBAL were slightly better than GA's results for the two MALBP-I data sets.

4.7.2 Type II

The set of computational experiments to address the balancing problem of type II, in which the goal is to minimise the cycle time of the assembly line for a given number of operators, consisted in using the GA and ANTBAL procedures (the best procedures to address type I, according to the results of the computational experience of the previous section) within the framework proposed in Figure 4.16. With this method, the problem of type I is iteratively solved for different values of the cycle time. Starting with a lower

bound for the cycle time, its value is successively increased until a feasible solution is achieved, i.e., a solution with the pre-specified number of operators. Then the SA smoothing procedure was applied in order to try to improve the cycle time values of the balancing solutions.

Problems 7, 8, 9, 10, 11, 17, 18 and 19 of the set of the problems with the characteristics described in Table 4.2 were used in this computational experience. For each of them, different values of the minimum replication time (*MRT*) and the total number of operators (*S*) were given as input. Table 4.7 presents the results of this experience, for each problem instance, in which the values of the cycle time are the best of ten runs. Columns '*SA imp(%)*' show the average improvement of the SA smoothing procedure and the best values of the cycle time are compared with the lower bound, computed by equation (4.31). The deviation from the lower bound (*LB*) is shown in column '*D*(%)' and the weighted efficiency of the best solution for each problem instance is given in the last column.

According to the results of the computational experience one can state that the performance of both GA and ANTBAL procedures was similar, when addressing this set of MALBP-II problem instances, being GA slightly superior to ANTBAL considering the minimisation of the cycle time.

The values of the average improvement of the cycle time after running the SA smoothing procedure were very low, which means that the iterative approach itself is a good way to tackle type II problems. Nevertheless, it may be useful to perform small changes in the resulting solutions, through swap or transfer movements, in order to better level the workloads among workstations.

Table 4.7 – Computational results for the MALBP-II data set

						GA			ANTBAL			
Problem	**N**	**M**	**S**	**MRT**	**LB**	**C**	**SA imp(%)**	**D(%)**	**C**	**SA imp(%)**	**D(%)**	**Best WE (%)**
A	25	2	16	5.9	8.3	8.3	0.0	0.0	8.4	0.0	1.2	92.6
				6.6	8.3	8.7	0.1	4.8	8.4	0.0	1.2	91.5
				8.8	8.8	9.3	0.3	5.7	9.6	0.1	9.1	82.6
				9.0	9.0	9.3	0.1	3.3	9.8	0.0	8.9	82.6
				9.4	9.4	9.4	0.9	0.0	9.8	0.0	4.3	81.7
B	28	2	21	4.2	8.9	9.1	0.0	2.2	9.1	0.0	2.2	95.0
				6.6	8.9	9.5	0.0	6.7	9.6	0.1	7.9	91.0
				7.7	8.9	9.4	0.0	5.6	9.6	0.3	7.9	92.0
				9.2	9.2	9.4	0.1	2.2	9.6	0.1	4.3	92.0
				9.8	9.8	9.8	0.0	0.0	9.8	0.0	0.0	88.2
C	30	2	16	4.8	8.9	9.1	0.0	2.2	9.1	0.0	2.2	96.4
				6.5	8.9	9.1	0.1	2.2	9.2	0.1	3.4	96.4
				7.8	8.9	9.3	0.1	4.5	9.2	0.1	3.4	95.3
				8.7	8.9	9.5	0.0	6.7	9.2	1.1	3.4	95.3
				9.9	9.9	9.9	0.0	0.0	9.9	0.0	0.0	88.6
D	45	2	25	4.8	8.9	9.0	0.2	1.1	9.2	0.1	3.4	95.6
				5.7	8.9	9.1	0.0	2.2	9.2	0.3	3.4	94.6
				7.3	8.9	9.4	0.1	5.6	9.3	0.3	4.5	92.6
				9.6	9.6	9.6	0.3	0.0	9.6	0.2	0.0	89.7
E	70	2	44	5.3	8.9	9.1	0.3	2.2	9.1	0.6	2.2	95.6
				7.4	8.9	9.5	0.2	6.7	9.5	0.2	6.7	91.5
				9.9	9.9	9.9	0.0	0.0	10.2	0.1	3.0	87.8
F	25	3	15	4.1	8.5	8.7	0.2	2.4	8.7	0.0	2.4	94.2
				6.8	8.5	8.6	0.0	1.2	8.7	0.0	2.4	95.3
				7.8	8.5	8.8	0.1	3.5	9.0	0.1	5.9	93.1
				8.5	8.5	8.7	0.0	2.4	9.0	0.0	5.9	94.2
G	28	3	20	4.7	8.7	9.0	0.4	3.4	8.9	0.2	2.3	93.5
				8.6	8.7	9.1	0.3	4.6	9.1	0.1	4.6	91.4
				9.3	9.3	9.3	0.0	0.0	9.3	0.0	0.0	89.5
H	45	3	28	5.6	8.5	8.8	0.2	3.5	8.9	0.4	4.7	93.2
				6.6	8.5	9.0	0.1	5.9	9.0	0.3	5.9	91.1
				7.5	8.5	9.0	0.0	5.9	9.1	0.2	7.1	91.1

4.7.3 Additional goals

The additional goals of balancing the workloads between and within workstations (functions B_b and B_w, respectively) are only envisaged after the maximisation of the weighted line efficiency (*WE*), in all of the three proposed meta-heuristic based procedures. As the objective function gives a higher importance to *WE*, the secondary goals B_b and B_w only become active when *WE* is maximum. Figure 4.18 presents the

variation of these different goals during the run of one of the procedures for a test problem. Computational experiments showed that this represents a typical variation of the workload balance values among the tested problems and that the performance of the three procedures, concerning the additional goals, was very similar.

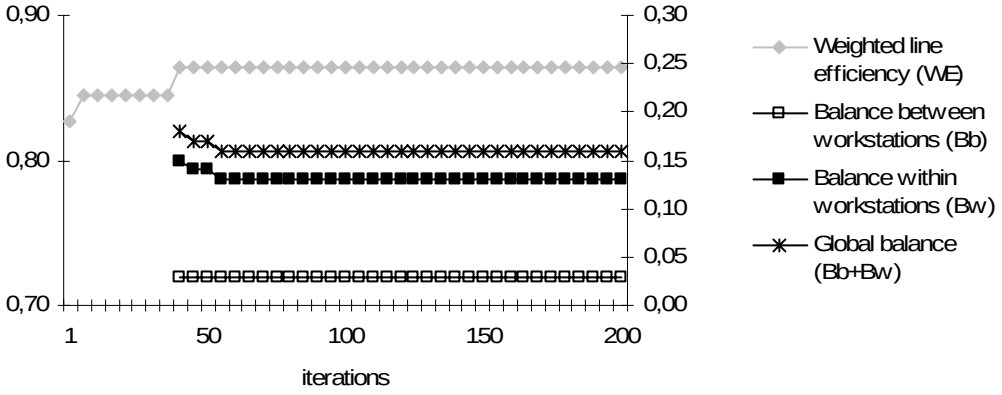

Figure 4.18 – An example of the variation of the workload balance functions

4.8 Chapter conclusions

In this chapter, a mixed-model assembly line balancing problem with special characteristics that reflect the operating conditions of real assembly lines was defined. The mathematical programming model was only used as a formal description of the problem, as it helps to describe the underlying principles of the proposed procedures. Due to its extreme complexity, its resolution was only possible for very small problems. To address larger sized problems three procedures based on the meta-heuristics simulated annealing (SA), genetic algorithms (GA) and ant colony optimisation (ANTBAL) were developed.

Computational results for type I problems showed that GA and ANTBAL clearly outperform SA while ANTBAL is slightly superior to GA. Considering type II problems, the approach of solving iteratively problems of type I for different values of the cycle time, until a solution with the pre-defined number of operators is found, showed a good performance.

The next step of the study was to address other assembly line balancing problems, namely balancing U-shaped and 2-sided lines. As it was said before, the approach is to solve complex problems and not to refine the techniques. So, to continue the study, only one meta-heuristic based procedure was selected to be adequately modified in order to solve these other balancing problems.

A conclusion from the results of the computational experiments presented in this chapter is that GA and ANTBAL have similar performances. So, the selection of only one of them to carry on the study was based on additional reasons. The development of ant colony optimisation algorithms is very recent and most of the researchers working on this area are still addressing traditional problems, like the travelling salesman problem, with which the analogy of the paths followed by the ants is much stronger.

On the opposite, genetic algorithms, and other similar evolutionary computational approaches, have been more widely applied. Particularly, the assembly line balancing problem, although in its most simple version, has been a frequent object of study from researchers on this area.

The contribution to the scientific knowledge would be more meaningful if the ACO algorithms were applied to more complex problems. So, ANTBAL was selected to continue the study.

5

Balancing U-shaped assembly lines

Contents

- **Chapter introduction**
- **Characteristics of U-shaped assembly lines**
- **Definition of the mixed-model U-ALBP**
- **U-ANTBAL: an ant colony optimisation based approach**
- **Computational experience**
- **Chapter conclusions**

5.1 Chapter introduction

In this chapter, the mixed-model U-shaped assembly line balancing problem (U-MALBP) is addressed. First, the main characteristics of U-shaped assembly lines are described and a brief review of existing techniques to tackle the single-model version of the line balancing problem for this type of assembly lines is provided. Then, the impact of mixed-model production on this type of lines is focused and a formal description of the addressed problem (U-MALBP) is presented, using a mathematical programming model. The ant colony optimisation algorithm developed to solve balancing problems in straight assembly lines, described in the previous chapter, was adapted in order to balance U-shaped assembly lines. This new procedure is presented and illustrated with a numerical example and its performance is tested through a set of computational experiments.

5.2 Characteristics of U-shaped assembly lines

The implementation of business philosophies such as *just-in-time* (JIT) is a way that companies have to cope with the constant changes in the external competitive environment. JIT suggests the use of multi-skilled workers and efficient facility layouts, so many companies are rearranging their traditional straight assembly lines into a U-shaped layout (Monden, 1993, Scholl and Klein, 1999, Aase et al, 2004). In a U-line, workers can move between the two legs of the 'U' to perform combinations of tasks that would not be allowed in a straight line. The space at the centre of the 'U' is a shared area where operators can communicate, help each other and learn one another's skills. (A graphical depiction of a U-shaped line was previously shown in Figure 2.5.)

Cheng et al (2000) and Miltenburg and Wijngaard (1994) summarise the main benefits and factors that favour the use of U-shaped assembly lines and explain its popularity among JIT practitioners. The main advantages of U-lines are the following:

(i) Operator flexibility and job enrichment: Operators are involved in different parts of the assembly process enlarging their skills. As they understand the relationships between tasks, they are better suited to make improvements in the

assembly process. The acquisition of multiple skills leads to higher motivation, improved product quality and increased flexibility.

(ii) Visibility and teamwork: The compact size and configuration of a U-line makes operators work closer to each other, improving visibility and communication between them. Quality problems can be more quickly detected and solved. Also it enhances teamwork making it easier for operators to help each other in cases of congestion.

(iii) Volume flexibility: The output of a U-shaped assembly line may need to be adjusted as a consequence of JIT principles, in which the production rate changes frequently. To achieve the desired production rate, the number of operators working on the line must be increased or decreased and a rapid reassignment of tasks among operators must be made. This level of flexibility is more difficult to attain in straight assembly lines due to its narrowly trained operators.

(iv) Number of workstations: The number of workstations required on a U-shaped line is never superior to that required on a straight line, because there are more possibilities of grouping tasks into workstations on a U-line. (The same workstation can perform tasks from the beginning and from the end of the precedence diagram, while in straight lines this is not possible.) This flexibility enables JIT companies to potentially reduce the total number of workers in their facility, creating a more efficient facility layout.

(v) Material handling: Usually, sub-assemblies are moved between workstations by the assembly line operators instead of using material handling equipment (such as conveyors or special material handling operators).

Miltenburg (2001) provides a review of the theory and practice on U-shaped production lines. In his study, a set of US and Japanese companies which changed their straight lines to U-lines is examined. The results show impressive benefits of the adoption of U-shaped configurations: productivity improvement of 76%, reduction of work-in-process inventory of 86%, decrease of lead time of 75% and defective rates reduction of 83%, on average. However, the U-shaped production lines of Miltenburg's study are mainly composed by machines operated by a small set of operators, making them different from the concept of assembly lines, whose work is essentially manual.

5.2.1 Literature review of approaches to solve the U-ALBP

The main focus of the research on U-lines has been on the development of techniques to solve the single-model U-shaped assembly line balancing problem (U-ALBP). Miltenburg and Wijngaard (1994), the first authors to study this problem, developed a dynamic programming exact procedure able to solve instances with up to 11 tasks. To address larger problems they proposed a set of single-pass heuristic procedures, able to solve instances with up to 111 tasks.

The integer programming formulation proposed by Urban (1998) managed to solve to optimality problems with up to 45 tasks. Scholl and Klein (1999) developed a branch-and-bound based heuristic called ULINO, which was adapted from a previous algorithm they had developed for balancing straight lines. The computational experience involved a large set of problems with up to 297 tasks and proved a good performance of the procedure, especially for the objective of minimising the number of workstations.

The problem of balancing a U-line facility with several U-lines connected by multi-line workstations was addressed by Miltenburg (1998) through the development of a dynamic programming formulation. The formulation was able to optimally solve problems with any number of U-lines as long as precedence diagrams for individual U-lines did not have more than 22 tasks.

Aase et al (2003) proposed a set of branch-and-bound procedures with different design elements (branching strategies, fathoming criteria, etc.) to solve the U-ALBP. These procedures are experimentally compared with other algorithms available in the literature. Significant improvements over the existing methods are reported by the authors when solving problem instances of reasonable application size for U-shaped layouts (problems with up to 50 tasks).

A goal programming approach to simultaneously consider several conflicting objectives was presented by Gökçen and Agpak (2006). The authors use a pre-emptive approach in which different goals, like number of workstations, cycle time and number of tasks per workstation, are ranked by some priority order. No comparison with other algorithms is provided, as the computational experience was only dedicated to the study of the multi-criteria version of the problem.

Guerriero and Miltenburg (2003) developed a mathematical model and recursive algorithms to solve the U-ALBP with stochastic task processing times. Computational experiments showed that the algorithms are able to solve problems of practical size.

Erel et al (2001) developed a simulated annealing based procedure to address the U-ALBP. A description of the most important features of this procedure was already provided on section 3.2.2 of this document. Computational experience yielded very good results for the set of tested problem instances.

Other studies on U-shaped lines have focused on the optimal worker allocation problem (Nakade and Ohno, 1999), product quality (Cheng et al, 2000), the effect of breakdowns (Miltenburg, 2000) and the impact on labour productivity (Aase et al, 2004).

All these studies have confirmed that the U-ALBP is a very significant problem for modern assembly systems. However they only deal with single-model assembly lines. The mixed-model U-shaped assembly line balancing problem is a more complicated problem to solve, but much more relevant within a context of increasing pressure for manufacturing flexibility and growing demand for customised products.

5.3 Definition of the mixed-model U-ALBP

The key difference between the straight assembly line balancing problem and the U-shaped assembly line balancing problem (U-ALBP) is related with the set of assignable tasks. In straight assembly lines, the set of assignable tasks at each moment is the set of tasks whose predecessors have already been assigned, in order to meet precedence constraints. In a U-shaped assembly line, the set of assignable tasks is the union of the set of tasks whose predecessors have already been assigned and the set of tasks whose successors have already been assigned.

The problem of balancing a U-shaped assembly line to produce a set of models of a product is the mixed-model U-ALBP (U-MALBP) and it was first described by Sparling and Miltenburg (1998). An additional and very important issue of mixed-model U-lines, when compared with single-model ones, is the fact that in the same cycle a workstation may perform its tasks in two different models, one at each leg of the line. This situation is illustrated in Figure 5.1. The line produces three models in the sequence ABC. In a

determined cycle, the operator of workstation 2 performs task 2 on model C at the front of the line and then crosses to the back to complete task 8 on model B.

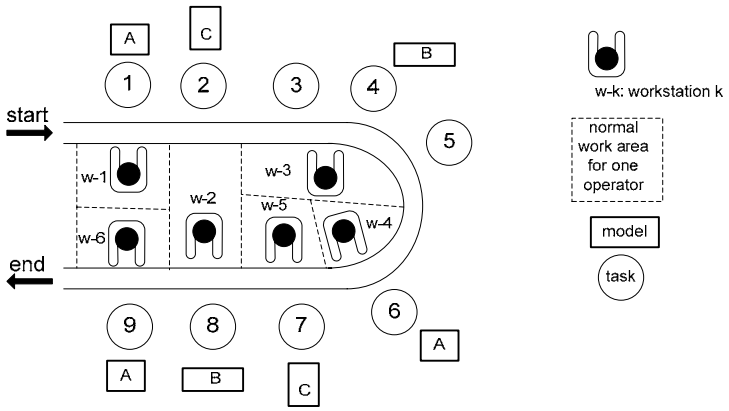

Figure 5.1 – Mixed-model production on a U-shaped assembly line

A very common practice when dealing with the mixed-model nature of the assembly process is to use average task processing times to assign tasks to workstations. A task is assigned to a workstation as long as the sum of its weighted average processing time with the current workstation workload does not exceed the cycle time. Then, finding the right model sequence (the sequence in which the models are launched to the line) is highly important, in order to allow a good workload balance within the workstations. In the U-MALBP this issue becomes even more important, not only because the models may be different from cycle to cycle, but because they also may be different within the same cycle.

All the existing approaches to the U-MALBP find an initial assignment of tasks based on their weighted average processing time and use some kind of procedure to reduce the unbalance of the initial balancing solution, using the task processing times for each model.

Sparling and Miltenburg (1998) use the combined precedence diagram and the weighted average task processing times to create a single-model balancing problem and, using a branch-and-bound algorithm, an optimal solution for this problem is obtained, called initial balance. Several unbalance measures, regarding the mixed-model nature of the original problem, are defined and computed for the initial balance. Then, a smoothing algorithm is applied in order to reduce the unbalance. This algorithm exchanges tasks between

workstations so that the value of the selected unbalance measure decreases. The unbalance measures used by these authors are (i) binary variables indicating whether the total time in a workstation exceeds the cycle time, (ii) amount of the total time, if any, that exceeds the cycle time in a workstation and (iii) absolute deviation of total time from a determined goal in a workstation. An important aspect of this approach is that the sequence in which the models are launched in the U-shaped line must be known, as it directly influences the values of the unbalance measures.

Kim et al (2000) address simultaneously the problems of balancing and sequencing mixed-model U-lines, as both the line balance and the model sequence influence the performance measure used by the authors: the absolute deviation of workloads. These authors propose a cooperative co-evolutionary algorithm which maintains two sets of populations, one to represent solutions of the line balancing problem and the other to represent solutions of the model sequencing problem. Each individual in a population has a matching pair in the other population and fitness (based on the absolute deviation of workloads) is computed for the pair of individuals. To generate new individuals, different genetic operators are defined for the each of the populations. Computational experiments proved a good performance of the procedure when compared with that of the hierarchical approach and of two other co-evolutionary algorithms for the same set of test problems.

Miltenburg (2002) also considers the problems of balancing the line and sequencing the models simultaneously, however the goal to achieve is the generation of level production schedules for other production facilities operating in JIT environment. It takes into account the number of parts, from each of the different production facilities, which each model requires to be assembled. A genetic algorithm approach was used to address the problem.

The following section describes the characteristics of the addressed U-MALBP.

5.3.1 Problem assumptions and constraints

The existing procedures to solve the U-MALBP have demonstrated a great influence of the model sequence in the line balances obtained. However, like for the straight MALBP (described in chapter 4), the goal of this work was to study only the balancing problem, so the approach was to try to find good line balances able to cope with any model sequence. In the particular case of U-shaped assembly lines, the model sequence interferes with the

mix of models within each workstation in the same cycle. To address this issue, the assignment of tasks to workstations is performed using each possible model combination for each cycle in a workstation. This way, regardless of the model sequence, the workload of each workstation never exceeds the required cycle time.

In the proposed approach, a set the of similar models of a product (m=1,…,M) are produced in a U-shaped assembly line, in any order or mix, over a pre-specified planning horizon, P. The forecasted demand, over the planning horizon, for model m is D_m, requiring the line to be operated with a cycle time given by

$$C = P/\sum_{m=1}^{M} D_m \qquad \textbf{(5.1)}$$

The production share of each model m is computed by

$$q_m = D_m / \sum_{p=1}^{M} D_p \quad (m = 1,..., M) \qquad \textbf{(5.2)}$$

The combined precedence diagram for all models has N tasks (numbered i=1,…,N) and t_{im} is the time required to perform task i on model m.

As it was referred earlier, an operator in a U-line may perform tasks at both legs of the line, so it is necessary to identify which tasks are performed at the front and which tasks are perform at the back of the line. Therefore, the following decision variables of the mathematical programming model are defined:

$$x_{ik}^{F} = \begin{cases} 1, \text{if task } i \text{ is assigned to workstation } k \text{ at the front of the U - line} \\ 0, \text{otherwise } (i = 1,..., N; k = 1,..., S) \end{cases} \qquad \textbf{(5.3)}$$

$$x_{ik}^{B} = \begin{cases} 1, \text{if task } i \text{ is assigned to workstation } k \text{ at the back of the U - line} \\ 0, \text{otherwise } (i = 1,..., N; k = 1,..., S) \end{cases} \qquad \textbf{(5.4)}$$

where S is the number of workstations (operators) of the assembly line. The assignment of a task to only one workstation, regardless of the model being assembled, is guaranteed by the following set of constraints:

$$\sum_{k=1}^{S} \left(x_{ik}^{F} + x_{ik}^{B} \right) = 1 \quad (i = 1,...,N) \qquad \textbf{(5.5)}$$

To verify the precedence constraints of the assembly process, it is necessary to guarantee that a task can only be assigned to a workstation if *either* all its predecessors *or* all its successors have been already assigned. If all the predecessors of a task *i* have been previously assigned to workstations at the front of the U-line, then task *i* can be assigned at the front of the line. This case is ensured by the set of constraints (5.6), in which $Pred_i$ is the set of predecessors of task *i*. The index of the workstation (*k*) to which task *i* is to assigned cannot be inferior to the index of the workstations to which its predecessors are assigned.

$$\sum_{k=1}^{S} kx_{hk}^{F} - \sum_{k=1}^{S} kx_{ik}^{F} \leq 0 \quad (i = 1,...,N; h \in Pred_i) \tag{5.6}$$

If all the successors of a task *i* have been previously assigned to workstations at the back of the U-line, then task *i* can be assigned at the back of the line. This case is ensured by the set of constraints (5.7), in which Suc_i is the set of successors of task *i*. The index of the workstation (*k*) to which task *i* is to be assigned cannot be inferior to the index of the workstations to which its successors are assigned, considering that they are assigned at the back of the line.

$$\sum_{k=1}^{S} kx_{ik}^{B} - \sum_{k=1}^{S} kx_{jk}^{B} \leq 0 \quad (i = 1,...,N; j \in Suc_i) \tag{5.7}$$

When included in the mathematical programming model these will be sets of disjunctive constraints, as only one is verified at a time.

The workload of a workstation will depend on the models that it performs at the front and at the back of the line in each cycle. Let *m* and *n* be two of the models to be assembled on the U-line. The workload of workstation *k* when model *m* is produced at the front and model *n* is produced at the back of the line is computed by:

$$W_{kmn} = \sum_{i=1}^{N} \left(t_{im} x_{ik}^{F} + t_{in} x_{ik}^{B} \right) (k = 1,...,S;\ m,n = 1,...,M) \tag{5.8}$$

In order to ensure that the cycle time is never exceeded, regardless of the pairs of models produced at the front and back of the line by each workstation on each cycle, the following set of constraints must hold for every workstation:

$$W_{kmn} \leq C \qquad (k = 1,...,S;\ m,n = 1,...,M) \tag{5.9}$$

The idle time of a workstation is the difference between the capacity of the workstation and its workload. s_{kmn} is idle time of workstation k when it performs its tasks on model m at the front and on model n at the back of the line and is computed by the set of equations (5.10).

$$\sum_{i=1}^{N} \left(t_{im} x_{ik}^{F} + t_{in} x_{ik}^{B} \right) + s_{kmn} = C \quad (k = 1,...S; m, n = 1,..., M) \tag{5.10}$$

Zoning constraints may also be included in the problem. Positive zoning constraints force pairs of tasks to be assigned to the same workstation and are defined by:

$$\sum_{k=1}^{S} k(x_{ik}^{F} + x_{ik}^{B}) - \sum_{k=1}^{S} k\left(x_{jk}^{F} + x_{jk}^{B} \right) = 0 \quad ((i, j) \in ZP) \tag{5.11}$$

where ZP is the set of pairs of tasks that must be assigned to the same workstation.

Negative zoning constraints are defined by:

$$\sum_{k=1}^{S} k(x_{ik}^{F} + x_{ik}^{B}) - \sum_{k=1}^{S} k(x_{jk}^{F} + x_{jk}^{B}) \neq 0 \quad ((i, j) \in ZN) \tag{5.12}$$

where ZN is the set of pairs of incompatible tasks.

5.3.2 Objective function

Similarly to the MALBP (defined in section 4.2), the goals of the U-MALBP are the following:

(i) minimisation of the number of workstations, for a given cycle time (for type I problems) or minimisation of the cycle time for a given number of workstations (for type II problems), both equivalent to the minimisation of the idle time of the line;

(ii) smoothing workloads between workstations;

(iii) smoothing workloads within workstations.

Given the particular characteristics of the U-MALBP, the expressions used to address these goals are different from the ones used for the straight MALBP.

To cope with the U-line mixed-model production, an objective function, called U-line idle time (WIT^U), was developed. It minimises the sum of the weighted idle times of each workstation, considering the probability of occurrence of each pair of models on the front and back of the line, q_{mn}, and it is given by:

$$Minimise \quad WIT^U = \sum_{k=1}^{S}\sum_{m=1}^{M}\sum_{n=1}^{M} q_{mn}\left(C - \sum_{i=1}^{N}\left(t_{im}x_{ik}^{F} + t_{in}x_{ik}^{B}\right)\right) \qquad \textbf{(5.13)}$$

As the sequence in which the models are launched into the line is not known, it is not possible to precisely determine the value of q_{mn}. To assemble M models in a U-line there are M^2 possible combinations of models in a workstation working on both legs of the line. An example with three models is shown in Figure 5.2. As the line balance must be feasible for all the possible sequence of models, including random sequences, it is reasonable to set equal probabilities of occurrence of each pair of models (model m at the front and model n at the back of the line). So, q_{mn} is set to $1/M^2$ for every pair (m,n).

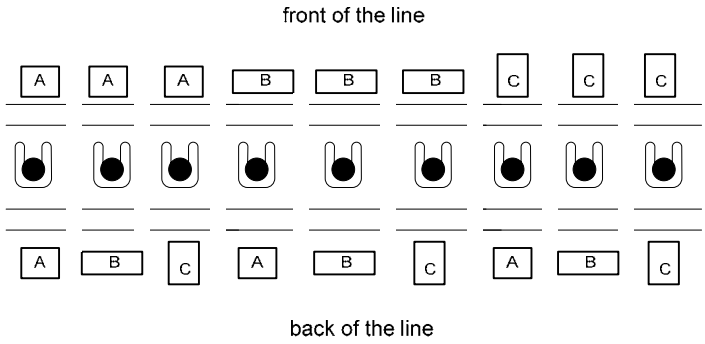

Figure 5.2 – Possible combinations of models in a workstation in the same cycle

In order to have a measure independent from the data of each problem instance, an alternative objective function called weighted U-line efficiency (WE^U) was defined. WE^U varies between 0 and 1. The more close to 1 (or 100%) the less idle time has the line. WE^U is an objective function to maximise and it is computed as follows:

$$Maximise \quad WE^U = \sum_{m=1}^{M}\sum_{n=1}^{M}\frac{1}{M^2}\left(\frac{\sum_{k=1}^{S}\sum_{i=1}^{N}\left(t_{im}x_{ik}^F + t_{in}x_{ik}^B\right)}{S \cdot C}\right) \tag{5.14}$$

Besides the minimisation of the number of workstations (or the minimisation of cycle time, for problems of type II), additional goals, concerning workload smoothing, are also envisaged. The objective function B_b^U aims to balance the workload between workstations, i.e., for each model the idle time is distributed across workstations as equally as possible, and it is given by:

$$Minimise \quad B_b^U = \frac{S}{S-1}\sum_{k=1}^{S}\left(\frac{S_k}{WIT^U} - \frac{1}{S}\right)^2 \tag{5.15}$$

where, S_k is the average idle time of workstation k computed by:

$$S_k = \sum_{m=1}^{M}\sum_{n=1}^{M}\frac{1}{M^2}s_{kmn} \tag{5.16}$$

The value of function B_b^U varies between a maximum of 1, when the average idle time of the line is equal to the idle time of one of the workstations, and a minimum of 0, when WIT^U is equally distributed by all workstations in the line. A demonstration of these values is presented in Appendix 4.

Due to the mixed-model nature of the problem and also the U-shaped configuration, each task processing time may vary among the different models and within each cycle a workstation may have to work on two models (one at the front and another at the back of the line). In order to ensure that each operator performs approximately the same amount of work regardless of the models being assembled, it is desirable to balance the workload within each workstation. To achieve this goal the objective function B_w^U was developed, which aims at smoothing the workload balance within each workstation and it is computed as follows:

$$Minimise \quad B_w^U = \frac{M^2}{S(M^2-1)}\sum_{k=1}^{S}\sum_{m=1}^{M}\sum_{n=1}^{M}\left(\frac{q_{mn}s_{kmn}}{S_k} - \frac{1}{M^2}\right)^2 \tag{5.17}$$

The value of function B_w^U varies between a maximum of 1, when the idle time of each workstation is only accountable to one combination of models (*m*,*n*), and a minimum of 0, when it is equally distributed by all combinations of models in every workstation. (A demonstration of these values is presented in Appendix 4.)

5.3.3 Complete mathematical programming model

The global objective function is then composed by three terms, each of which addressing one of the goals stated in the previous section. The complete mathematical programming model for the U-shaped mixed-model assembly line balancing problem is presented in Figure 5.3. The model constraints are interpreted as follows:

(i) constraints ensuring that each task is assigned to only one workstation of the station interval (assignment constraints);

(ii) disjunctive constraints ensuring that a task can be assigned to a workstation if *either* all its predecessors (*ii* a) *or* all its successors (*ii* b) have been assigned to the same or to an earlier workstation (in this set of constraints, u_i is an auxiliary binary variable and $\mathcal{M}$ is a very large positive integer);

(iii) constraints ensuring that each workstation capacity is not exceeded, as the use of parallel workstations was not accounted for in this model, the capacity of a workstation is the cycle time;

(iv) positive zoning constraints;

(v) negative zoning constraints,

(vi) set of constraints computing the number of operators required by the line (*S*) in which the auxiliary binary variable y_k equals one, if the k^{th} workstation is used for assembly and zero, otherwise (in this set of constraints, *K* is an upper bound for the number of workstations and $\mathcal{M}$ is a very large positive integer);

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

The proposed mathematical programming is only used as a means to formally describe the problem, as its high complexity makes it impossible to be solved to optimality. The following section describes U-ANTBAL, an ant colony optimisation based approach developed to find solutions for the U-MALBP.

$$\textit{Maximise} \quad \lambda WE^U - B_b^U - B_w^U$$

subject to :

$$\sum_{k=1}^{K}\left(x_{ik}^F + x_{ik}^B\right) = 1 \qquad (i = 1,...,N) \qquad (i)$$

$$\sum_{k=1}^{K} kx_{hk}^F - \sum_{k=1}^{K} kx_{ik}^F \le \mathcal{M}u_i \qquad (i = 1,...,N; h \in Pred_i) \qquad (ii \text{ a})$$

$$\sum_{k=1}^{K} kx_{ik}^B - \sum_{k=1}^{K} kx_{jk}^B \le \mathcal{M}(1 - u_i) \qquad (i = 1,...,N;\ j \in Suc_i) \qquad (ii \text{ b})$$

$$\sum_{i=1}^{N}\left(t_{im}x_{ik}^F + t_{in}x_{ik}^B\right) + s_{kmn} = C \qquad (k = 1,...K; m,n = 1,...,M) \qquad (iii)$$

$$\sum_{k=1}^{K} k(x_{ik}^F + x_{ik}^B) - \sum_{k=1}^{K} k\left(x_{jk}^F + x_{jk}^B\right) = 0 \qquad ((i,j) \in ZP) \qquad (iv)$$

$$\sum_{k=1}^{K} k(x_{ik}^F + x_{ik}^B) - \sum_{k=1}^{K} k(x_{jk}^F + x_{jk}^B) \ne 0 \qquad ((i,j) \in ZN) \qquad (v)$$

$$\sum_{i=1}^{N}\left(x_{ik}^F + x_{ik}^B\right) \le \mathcal{M}y_k \qquad (k = 1,...K) \qquad (vi \text{ a})$$

$$\sum_{i=1}^{N}\left(x_{ik}^F + x_{ik}^B\right) \ge y_k \qquad (k = 1,...K) \qquad (vi \text{ b})$$

$$S = \sum_{k=1}^{K} y_k \qquad (vi \text{ c})$$

$$s_{kmn} \ge 0 \qquad (k = 1,...K; m,n = 1,...,M) \qquad (vii \text{ a})$$

$$x_{ik}^F, x_{ik}^B \in \{0,1\} \qquad (i = 1,...,N; k = 1,...K) \qquad (vii \text{ b})$$

$$u_i \in \{0,1\} \qquad (i = 1,...,N) \qquad (vii \text{ c})$$

$$y_k \in \{0,1\} \qquad (k = 1,...K) \qquad (vii \text{ d})$$

$$S > 0 \qquad S \text{ is integer} \qquad (vii \text{ e})$$

Figure 5.3 – Mathematical programming model for the U-MALBP

5.4 U-ANTBAL: an ant colony optimisation based approach

The ant colony optimisation based approach developed to tackle the straight mixed-model assembly line balancing problem, described in section 4.5 was modified in order to address the U-shaped problem. This new procedure is called U-ANTBAL and its main steps are presented in Figure 5.4.

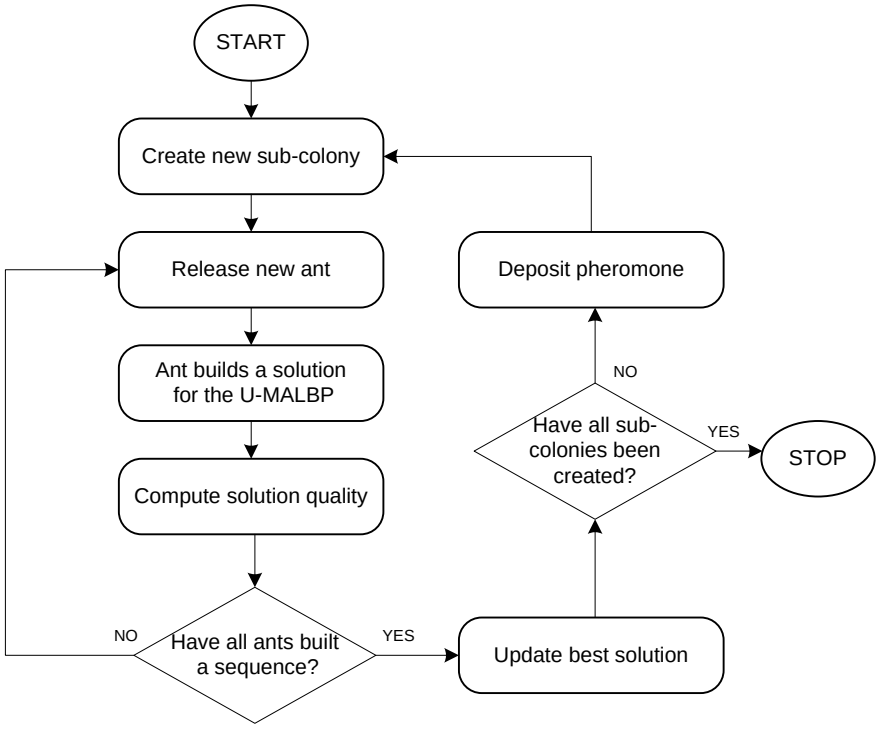

Figure 5.4 – Outline of U-ANTBAL

The features and parameters of U-ANTBAL are similar to the ones of ANTBAL, except the following:

(i) When building a balancing solution, an ant must determine the set of available tasks, i.e., the set of tasks that can be assigned to the current workstation. A task is available if it verifies (i) capacity constraints, defined by equations (5.10), (ii) zoning constraints, defined by equations (5.11) and (5.12) and (iii) precedence constraints, defined by equations (5.6) or (5.7) determining which tasks are assignable to the front and which tasks are assignable to the back of the line, respectively. The way an ant builds a balancing solution of a U-MALBP is depicted in Figure 5.5.

(ii) The objective function used to guide the search in U-ANTBAL is the one of the mathematical programming model of the previous section: $Z = \lambda WE^U - B_b^U - B_w^U$. This quality measure is also used as the amount of pheromone released by the ants, as it was described in section 4.5.1.4.

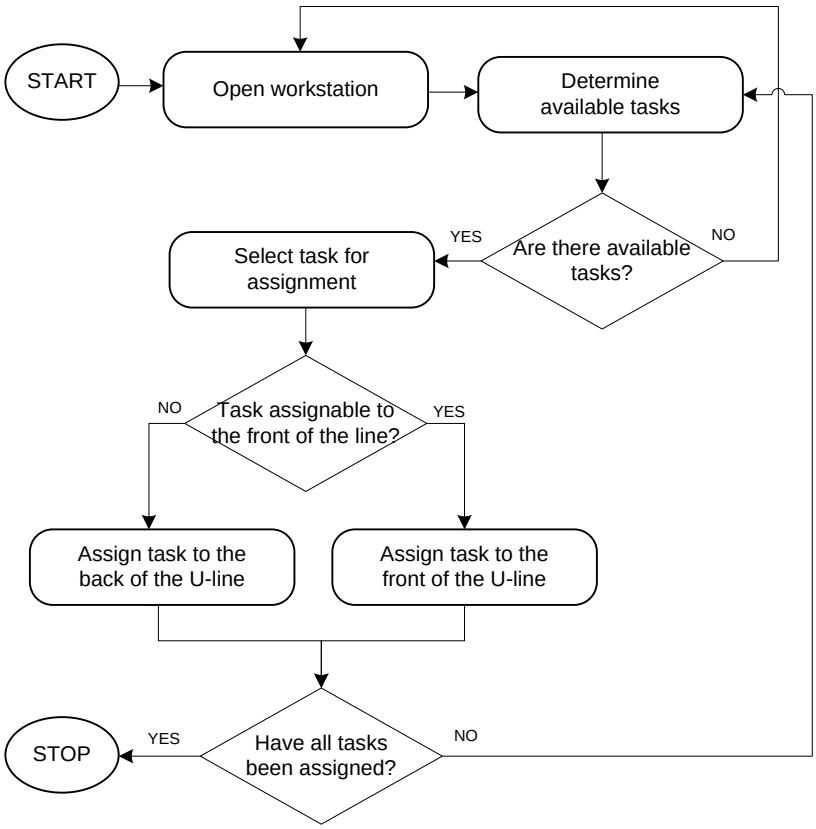

Figure 5.5 – Building a balancing solution in U-ANTBAL

5.4.1 Use of parallel workstations

Although in the definition of the U-MALBP the capacity of each workstation was the cycle time, the implemented ANTBAL algorithm allows the creation of parallel workstations in a similar way as it was described for straight assembly lines (in section 4.2). This was due to the need of using the same data set of mixed-model assembly line balancing problems that was used in the computational experiments for straight assembly lines.

This way, U-ANTBAL allows the replication of workstations that perform tasks with processing time higher than *MRT* (minimum replication time) for, at least, one of the models. These workstations will have two or more operators working in parallel (in replicas of the workstation). 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:

$$R_k = \left\lceil \frac{\max\limits_{m=1,...,M;i=1,...,N} \left\{ t_{im} \left(x_{ik}^F + x_{ik}^B \right) \right\}}{MRT} \right\rceil \quad (k = 1,...LL) \qquad \textbf{(5.18)}$$

where *LL* will be the line length, i.e., the number of different workstations. The capacity of a workstation *k* will then depend on the tasks it performs: if all its tasks are not greater than *MRT*, then its workload must not exceed cycle time and the set of constraints defined in (5.9) must be verified. Otherwise, the capacity constraints are given by:

$$W_{kmn} \leq R_k \cdot C \qquad (k = 1,...,LL; m, n = 1,...,M) \qquad \textbf{(5.19)}$$

where W_{km} is the workload of workstation *k* when it works on model *m* at the front and on model *n* at the back of the line, already defined by expression (5.8).

The physical implementation of parallel workstations in U-shaped assembly lines may be possible with an adequate material handling system and/or an agile operator's positioning along the line.

5.4.2 Numerical illustration

The goal of this section is to show the differences in the balancing solutions obtained for straight line configurations and U-shaped configurations, for the numerical example of section 4.3.3. The best solution for a straight configuration has 16 operators working on 14 different workstations (two workstations are replicated) and it is presented in the upper side of Figure 5.6. Using U-ANTBAL, the same problem was solved and U-shaped balancing solutions with 15 operators (13 different workstations) were obtained. Two of these solutions are depicted in the lower part of Figure 5.6. The reduction of one workstation was due to the more flexible nature of the precedence constraints in U-lines, as tasks from different parts of the assembly process can be performed by the same operator at the front and at the back of the line.

The two U-line configurations are solutions obtained when running U-ANTBAL. Solution 1 is an intermediate solution while solution 2 is the best solution provided by the algorithm. Both have 15 operators, but have different assignments of tasks to workstations, leading to different values of the workload balance functions B_b^U and B_w^U.

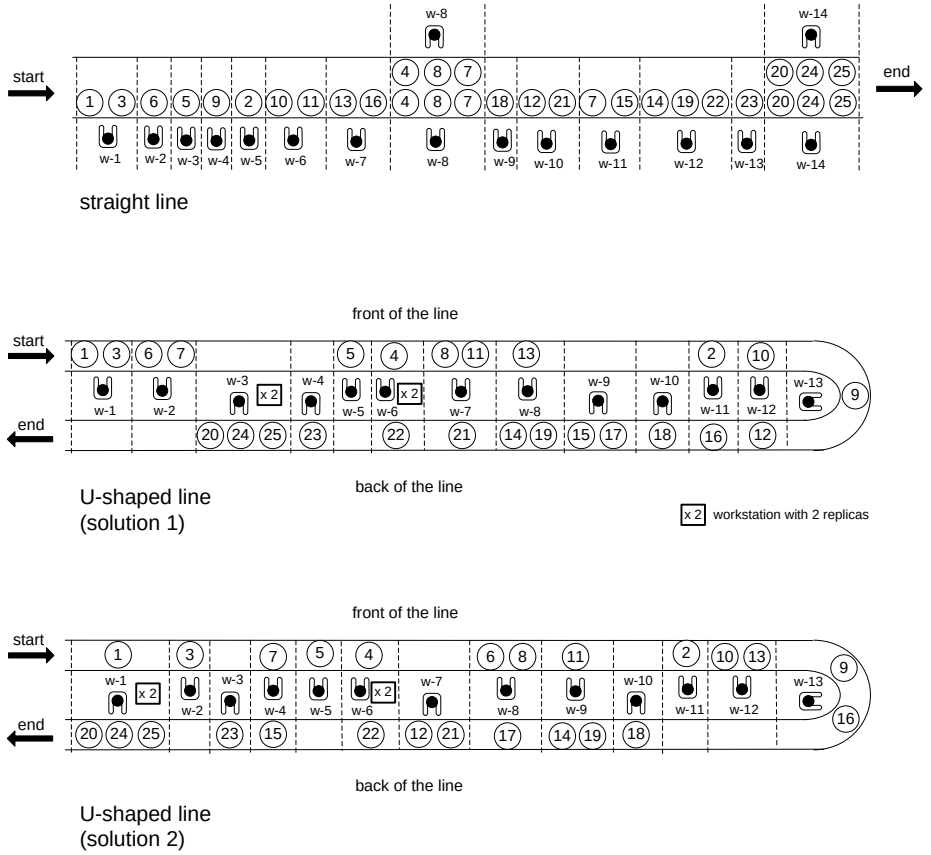

Figure 5.6 – Straight and U-shaped line configurations for the numerical example

Table 5.1 presents, for each solution, the set of tasks assigned to the front and to the back of each workstation and the workloads for every model combination (AB is the workload of a workstation when it works on model A at the front and on model B at the back of the line). It also presents the values of S_k, i.e., the workstation's average idle time, computed by expression (5.16). For workstations with two replicas (workstations 3 and 6 of solution 1 and workstations 1 and 6 of solution 2) the idle time is computed by the difference between twice the cycle time and the workload.

For solution 1, the balance between workstations has a value of B_b^U =0.06 and the balance within workstations has a value of B_w^U =0.07. For the same number of operators, U-ANTBAL tries to improve the global workload balance ($B_b^U + B_w^U$). This way, solution 2, the best solution obtained for this problem, presents better workload balance values: B_b^U =0.03 and B_w^U =0.04, which shows a considerable improvement the global balance.

Table 5.1 – Task assignments and workload values for the two U-line solutions

SOLUTION 1

	Tasks		Workload				
k	**Front**	**Back**	**AA**	**AB**	**BA**	**BB**	**S_k**
1	1,3		7.3	7.3	9.3	9.3	1.7
2	6,7		9.9	9.9	0	0	5.1
3		20,24,25	16.6	12.7	16.6	12.7	5.4
4		23	9.6	8.2	9.6	8.2	1.1
5	5		8.8	8.8	8.8	8.8	1.2
6	4	22	19.7	19.7	19.7	19.7	0.3
7	8,11	21	7.5	7.5	9.5	9.5	1.5
8	13	14,19	8.5	7.2	8.5	7.2	2.2
9		15,17	9.2	5.5	9.2	5.5	2.7
10		18	9.4	9.4	9.4	9.4	0.6
11	2	16	9.6	9.7	9.6	9.7	0.4
12	10	12	9.6	9.6	9.6	9.6	0.4
13	9		6.6	6.6	6.6	6.6	3.4

SOLUTION 2

	Tasks		Workload				
k	**Front**	**Back**	**AA**	**AB**	**BA**	**BB**	**S_k**
1	1	20,24,25	16.6	12.7	18.6	14.7	4.4
2	3		7.3	7.3	7.3	7.3	2.7
3		23	9.6	8.2	9.6	8.2	1.1
4	7	15	9.1	9.1	5.5	5.5	2.7
5	5		8.8	8.8	8.8	8.8	1.2
6	4	22	19.7	19.7	19.7	19.7	0.3
7		12,21	9.1	9.1	9.1	9.1	0.9
8	6,8	17	9.9	6.2	5.7	2.0	4.1
9	11	14,19	8.1	6.8	8.1	6.8	2.6
10		18	9.4	9.4	9.4	9.4	0.6
11	2		7.7	7.7	7.7	7.7	2.3
12	10,13		8.4	8.4	8.4	8.4	1.6
13	9,16		8.5	8.6	8.5	8.6	1.5

5.5 Computational experience

U-ANTBAL was coded in C and run on a 2.8 GHz Pentium 4 computer. To test its performance the two sets of mixed-model assembly line balancing problems used in the computational experiments of section 4.7.1 were solved and the number of operators of the solutions obtained by U-ANTBAL was compared with the best ones obtained for straight lines. The results, presented in Table 5.2, show that for some problem instances it is possible to reduce the number of operators of the line just by changing its configuration from straight to U-shaped. For the first data set, U-ANTBAL improved the solution of nine of the 20 instances while for the second it was only able to improve the solution of two problem instances. This performance was somehow predictable, due to the random nature of the task times of the second data set. There is a high number of large tasks that cannot be combined in the same workstation, so the advantage of using a U-shaped configuration is not so high as it is for problems with tasks with typical times, as in the first data set.

Table 5.2 – Computational results (number of operators) of U-ANTBAL for the two MALBP data sets

	MALBP data set with typical times		**MALBP data set with random times**	
Problem	**straight**	**U-shaped**	**straight**	**U-shaped**
1	4	4	11	11
2	8	8	11	11
3	7	7	11	10
4	7	6	16	16
5	16	14	29	29
6	15	13	35	35
7	16	15	40	40
8	14	14	40	40
9	20	20	35	35
10	20	19	34	34
11	16	16	38	38
12	19	19	50	50
13	19	17	50	49
14	19	18	54	54
15	23	23	47	47
16	24	23	52	52
17	24	24	59	59
18	26	26	78	78
19	43	43	88	88
20	44	43	104	104

5.6 Chapter conclusions

In this chapter, the mixed-model U-shaped assembly line balancing problem was address. A mathematical programming model was used to formally describe the problem and an ant colony optimisation algorithm was developed to solve it. A distinctive feature of this approach from existing ones is the fact that it does not depend on the sequence in which the models are launched into the line. The line configurations provided by the proposed procedure are adequate for every sequence of models that might occur. This flexibility is very important for companies operating under lean production philosophies such as JIT.

The results of the computational experiments carried out in this study showed that the proposed procedure is able to decrease the number of operators of an assembly line by using a U-shaped configuration rather than a straight line configuration. Also, the possibility of having replicated workstations allows the line to increase its production rate, when the required cycle time is lower than some of the task processing times.

6

Balancing 2-sided assembly lines

Contents

6.1 Chapter introduction

In this chapter, the 2-sided mixed-model assembly line balancing problem (2-MALBP) is addressed. First, the main characteristics of 2-sided assembly lines are described and a brief review of existing techniques to tackle the single-model version of the line balancing problem for this type of lines is provided. Then, a formal description of the addressed problem (2-MALBP) is presented, using a mathematical programming model and the ant colony optimisation algorithm developed to solve it is presented. The procedure is illustrated with a numerical example and its performance is tested through a set of computational experiments.

6.2 Characteristics of 2-sided assembly lines

Typically, 2-sided assembly lines are used in the production of large-sized products, such as trucks and buses (Kim et al, 2000). The assembly process of this type of products may be different from the assembly of small products, as some assembly tasks are required to be performed on a specific side of the product or at both sides of the product simultaneously (by different operators).

The structure of a 2-sided assembly line is depicted in Figure 6.1. The line has two sides, left and right, and, in most cases, at each position there is a pair of workstations directly facing each other. The two opposite operators perform, in parallel, different tasks but on the same individual item. This is different from the concept of parallel workstations, where different operators perform the same tasks but on different items.

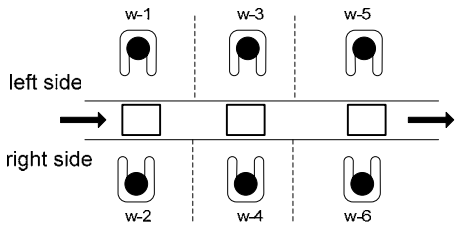

Figure 6.1 – Configuration of a 2-sided assembly line

According to Bartholdi (1993), in practice a 2-sided line can provide several advantages over a one-sided line, like the reduction of (i) the number of operators, (ii) the throughput time, (iii) the cost of tools and fixtures, as they can be shared by the operators of both sides and (iv) material handling costs.

The main difference between the assignment of tasks in one-sided lines and in 2-sided lines is in the relevance of the sequence in which the tasks are performed. In one-sided lines the sequence of the tasks within a workstation is not important as long as it verifies precedence constraints. However, in 2-sided assembly lines, this is a crucial factor for an efficient assignment of tasks. Tasks at opposite sides of the line can interfere with each other through precedence constraints which might cause idle time if a workstation needs to wait for a predecessor task to be completed at the opposite side of the line. This phenomenon is called *interference* and it is illustrated in Figure 6.2.

The precedence diagram of the tasks required to assemble a product is shown in the right side of Figure 6.2. Some tasks have to be performed on a specific side of the line (L-left side, R-right side) while others may be processed on either side (E). Let task 1, with processing time of 2 time units (t.u.), be assigned to the left side and task 2, with processing time of 8 time units, be assigned to the right side of the line. Task 3 can be assigned to workstation 1 right after task 1, as its unique predecessor is task 1. Task 2 is completed, in workstation 2, after 8 t.u., however, it is necessary to wait for the completion of task 3 before any other task becomes available. Workstation 2 is obliged to remain idle for 1 t.u., so task 3 *interferes* with the next task to be assigned.

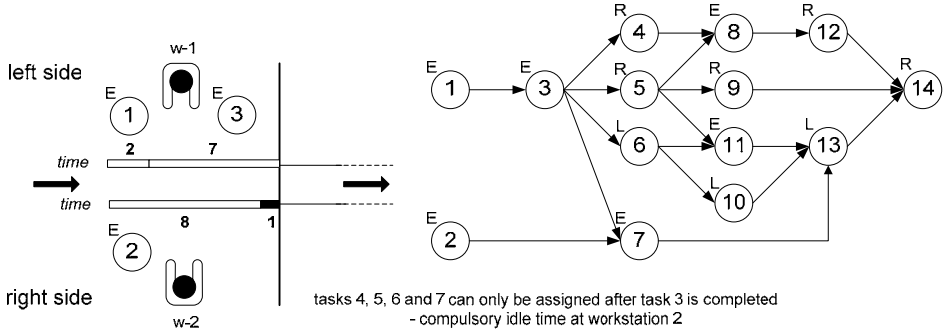

Figure 6.2 – Interference in 2-sided assembly lines

6.2.1 Literature review of approaches to solve the 2-ALBP

The literature on the 2-sided assembly line balancing problem (2-ALBP) is scarce. Bartholdi (1993) was the first author to address the 2-ALBP. His work comprehends an interactive computer program embodied with a balancing algorithm, based on the 'first fit' heuristic, that enables line managers to rapidly refine the solutions provided by the algorithm. Kim et al (2000) present a genetic algorithm approach for 2-ALBP, while Lee et al (2001) propose a group assignment procedure focusing on the maximisation of work relatedness and work slackness. An industrial case study is presented by Lapierre and Ruiz (2004), in which an enhanced priority-based heuristic is applied to balance a 2-sided assembly line. This study was extended to the application of a taboo search procedure in Lapierre et al (2006).

All these studies report on the 2-ALBP for single-model assembly lines, but this type of line is not suited for high levels of product customisation, a crucial factor for companies to be competitive under current market trends and essential to address in the final stage assembly lines of the automotive industry. The present work addresses the problem of balancing mixed-model 2-sided assembly lines. A definition of this problem is presented in the following section.

6.3 Definition of the mixed-model 2-ALBP

In a mixed-model 2-sided assembly line, a set of similar models of a product is assembled, in any order and mix, by workers that perform assembly tasks on a set of assembly stations, each of which has a pair of workstations directly opposite each other (left and right side workstations). In each cycle, the two operators working at the different sides of the line, at each position, perform their tasks in the same individual item, thus in the same model. So, the approach to address the mixed-model nature of the problem is similar to the one of the straight mixed-model assembly line.

The particularity of 2-sided lines is concerned with sequencing the tasks within each workstation, at both sides of the line, in a way that minimises the compulsory idle time due to the phenomenon of interference.

6.3.1 Problem assumptions and constraints

In the proposed approach, the assembly line has two sides, left and right, with a set of workstations positioned at each side of the line. A set of similar models of a product (numbered m=1,…,M) is produced in the 2-sided assembly line, in any order or mix, over a pre-specified planning horizon, P. The forecasted demand, over the planning horizon, for model m is D_m, requiring the line to be operated with a cycle time given by

$$C = P / \sum_{m=1}^{M} D_m \qquad \textbf{(6.1)}$$

The overall proportion of the number of units of model m being assembled is computed by

$$q_m = D_m / \sum_{p=1}^{M} D_p \quad (m = 1,...,M) \qquad \textbf{(6.2)}$$

The combined precedence diagram for all models has N tasks (numbered i=1,…,N) and t_{im} is the time required to perform task i on model m. Also, the side of the line in which a task is performed is defined in the assembly process. Tasks can be:

(i) performed on either side of the line: S_E is the set of tasks that can be performed on either side of the line;

(ii) required to be performed on a specific side of the line: S_L (S_R) is the set of tasks that must be performed on the left (right) side of the line;

(iii) required to be performed simultaneously on both sides of the line, so that a pair of operators can collaborate: these tasks are called synchronous tasks and each one calls the other mated-task. S_C is the set of pairs of synchronous tasks.

In 2-sided lines there are workstations at both sides of the line, so it is necessary to identify the workstation and side at which tasks are performed. Therefore, the following decision variables are defined:

$$x_{ikb} = \begin{cases} 1, \text{ if task } i \text{ is assigned to workstation } k \text{ at side b (b = L [left], R [right])} \\ 0, \text{ otherwise } (i = 1,..., N; k = 1,..., LL) \end{cases} \qquad \textbf{(6.3)}$$

In this case, LL will be the length of the 2-sided line, considering that at each position k there will be two operators working, one at each side of the line.

The assignment of a task to only one workstation, regardless of the model being assembled, is guaranteed by the following set of constraints:

$$\sum_{b=L}^{R}\sum_{k=1}^{LL} x_{ikb} = 1 \quad (i = 1,...,N) \tag{6.4}$$

The assignment of tasks to a specific side of the line, as required by the assembly process, must also be assured. The set of constraints (6.5) assigns left-side tasks to the left side of the line while the set of constraints (6.6) forces the assignment of right-side tasks to the right side of the line.

$$\sum_{k=1}^{LL} x_{ikL} = 1 \quad (i \in S_L) \tag{6.5}$$

$$\sum_{k=1}^{LL} x_{ikR} = 1 \quad (i \in S_R) \tag{6.6}$$

To deal with the interference issue it is necessary to establish the sequence, within a workstation, in which the tasks are going to be performed. T_i, the starting time of task i, will be a decision variable of the model. T_i represents the time instant at which a workstation begins to process task i and its value is within the range $0 \leq T_i < C$.

The assignment of tasks to workstations at both sides of the line must take into account the precedence constraints of the problem. A task can only be processed when all its predecessors are completed. Let task i be a predecessor of task j. In order to verify precedence constraints, the starting time of task j must never be earlier that the starting time of task i added by the processing time of task i, as defined by the set of constraints (6.7), where $\max_m\{t_{im}\}$ is the maximum processing time of task i, considering all models.

$$\sum_{b=L}^{R}\sum_{k=1}^{LL} kx_{ikb}T_i + \max_m\{t_{im}\} - \sum_{b=L}^{R}\sum_{k=1}^{LL} kx_{jkb}T_j \leq 0 \quad (i = 1,..., N;\ j \in Suc_i) \tag{6.7}$$

Synchronous tasks must be performed simultaneously, one at each side of the line. The set of constraints (6.8) ensures that these pairs of tasks are assigned to workstations directly facing each other (with the same index but one at each side) and will have the same starting time.

$$\sum_{k=1}^{LL} kx_{ikL} T_i - \sum_{k=1}^{LL} kx_{jkR} T_j = 0 \quad ((i,j) \in S_C) \tag{6.8}$$

When there are no precedence relationships or synchronism constraints between tasks it is still necessary to determine the scheduling of tasks within workstations, preventing overlapping of tasks. Considering tasks *i* and *j* one of two situations can occur: either (i) task *i* is assigned before task *j* or (ii) task *i* is assigned after task j. The first situation is modelled by the set of constraints (6.9) while the second is modelled by the set (6.10).

$$\sum_{b=L}^{R} \sum_{k=1}^{LL} kx_{ikb} T_i + \max_m \{t_{im}\} - \sum_{b=L}^{R} \sum_{k=1}^{LL} kx_{jkb} T_j \le 0 \quad (i = 1,...,N; j \notin Suc_i) \tag{6.9}$$

$$\sum_{b=L}^{R} \sum_{k=1}^{LL} kx_{jkb} T_j + \max_m \{t_{jm}\} - \sum_{b=L}^{R} \sum_{k=1}^{LL} kx_{ikb} T_i \le 0 \quad (i = 1,...,N; j \notin Suc_i) \tag{6.10}$$

When included in the mathematical programming model these will be sets of disjunctive constraints, as only one is verified at a time.

The idle time of a workstation is computed by the difference between its capacity (the cycle time) and the sum of the processing times of the tasks that it performs. s_{kbm} is the idle time of workstation *k* of side *b* when it works on model *m*, and it is given by:

$$s_{kbm} = C - \sum_{i=1}^{N} x_{ikb} t_{im} \quad (k = 1,...,LL; b = L,R; m = 1,...,M) \tag{6.11}$$

However these set of constraints are not enough to ensure that the capacity of a workstation is not exceeded, because there may exist idle time between two consecutive tasks within a workstation (due to interference). Therefore the set of constraints (6.12) must also be included in the mathematical model in order to guarantee that the completion time instant of a task is never higher than the cycle time.

$$x_{ikb} \left(T_i + \max_m \{t_{im}\} \right) \le C \quad (i = 1,...,N; k = 1,...,LL; b = L,R) \tag{6.12}$$

Positive and negative zoning constraints may also be included in the problem. The first are verified by the set of constraints (6.13) while the second are guaranteed by the set (6.14).

$$\sum_{k=1}^{LL} kx_{ikb} - \sum_{k=1}^{LL} kx_{jkb} = 0 \quad ((i,j) \in ZP; b = L,R) \tag{6.13}$$

$$\sum_{k=1}^{LL} kx_{ikb} - \sum_{k=1}^{LL} kx_{jkb} \neq 0 \quad ((i,j) \in ZN; b = L,R) \tag{6.14}$$

6.3.2 Objective function

Similarly to the MALBP and the U-MALBP the goals of the 2-MALBP are: (i) minimisation of the idle time of the line, (ii) smoothing workloads between workstations and (iii) smoothing workloads within workstations.

The weighted idle time of a 2-sided assembly line is given by:

$$Minimise \quad WIT^{2s} = \sum_{m=1}^{M} q_m \sum_{k=1}^{LL} \sum_{b=L}^{R} \left(C - \sum_{i=1}^{N} t_{im} x_{ikb} \right) \tag{6.15}$$

The weighted line efficiency allows a measure of the line efficiency always within the value range [0,1]. The goal is to maximise this function, which is computed as follows.

$$Maximise \quad WE^{2s} = \sum_{m=1}^{M} q_m \left(\frac{\sum_{i=1}^{N} t_{im}}{S \cdot C} \right) \tag{6.16}$$

where S is the total number of workstations, i.e., the sum of operators working in the different sides of the line.

The objective function B_b^{2s} aims to balance the workload between workstations, i.e., for each model the idle time is distributed across workstations as equally as possible, and it is given by:

$$Minimise \quad B_b^{2s} = \frac{S}{S-1} \sum_{k=1}^{LL} \sum_{b=L}^{R} \left(\frac{S_{kb}}{WIT^{2s}} - \frac{1}{S} \right)^2 \tag{6.17}$$

where, S_{kb} is the average idle time of workstation k on side b computed by:

$$S_{kb} = \sum_{m=1}^{M} q_m s_{kbm} \tag{6.18}$$

The value of function B_b^{2s} varies between a maximum of 1, when the average idle time of the line is equal to the idle time of one of the workstations, and a minimum of 0, when WIT^{2s} is equally distributed by all workstations in the line.

In order to ensure that each operator performs approximately the same amount of work regardless of the models being assembled, the objective function B_w^{2s} is used, aiming to smooth the workload balance within each workstation. It is computed as follows:

$$Minimise \quad B_w^{2s} = \frac{M^2}{S(M^2-1)} \sum_{k=1}^{LL} \sum_{b=L}^{R} \sum_{m=1}^{M} \left(\frac{q_m s_{kbm}}{S_{kb}} - \frac{1}{M^2} \right)^2 \qquad \textbf{(6.19)}$$

The value of B_w^{2s} varies between a maximum of 1, when the idle time of each workstation is only due to one model and a minimum of 0, when it is equally distributed by all models in every workstation.

6.3.3 Complete mathematical programming model

The global objective function is composed by three terms, each of which addressing one of the goals stated in the previous section. The complete mathematical programming model for the 2-sided mixed-model assembly line balancing problem is presented in Figure 6.3. The model constraints are interpreted as follows:

(i) constraints ensuring that each task is assigned to only one workstation of the station interval (assignment constraints);

(ii) constraints ensuring that tasks required to be performed at a specific side of the line are assigned to the correct side (task side constraints);

(iii) constraints ensuring that no task is assigned before all its predecessors are completed (precedence constraints);

(iv) constraints ensuring that synchronous tasks are performed simultaneously by two operators, one at each side of the line;

(v) disjunctive constraints ensuring that a correct sequencing of tasks is made, i.e., a task *i* can be assigned *either* before task *j* (*ii* a) *or* after task *j* (*ii* b) as long as tasks *i* and *j* do not have any precedence relation (in this set of constraints, u_i is an auxiliary binary variable whose value determines which situation will occur and $\mathcal{M}$ is a very large positive integer);

(vi) constraints ensuring that each workstation capacity is not exceeded (capacity constraints);

(vii) positive zoning constraints;

(viii) negative zoning constraints,

(ix) set of constraints computing the total number of operators of the line (S) in which the auxiliary binary variable y_{kb} equals one, if the k^{th} workstation of side b of the line is used for assembly and zero, otherwise (in this set of constraints, K is an upper bound for the number of workstations on each side and $\mathcal{M}$ is a very large positive integer);

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

The proposed mathematical programming is only used as a means to formally describe the problem, as its high complexity makes it impossible to be solved to optimality. The following section describes 2-ANTBAL, an ant colony optimisation based approach developed to find solutions for the 2-MALBP.

6.4 2-ANTBAL: an ant colony optimisation based approach

In the proposed ACO algorithm for the 2-sided mixed model assembly line balancing problem, 2-ANTBAL, two ants 'work' simultaneously, one at each side of the line. They will be called left-ant and right-ant if they work on the left or right side of the line, respectively, and side-ant more generally. Figure 6.4 presents an outline of 2-ANTBAL.

The procedure starts by creating a sub-colony with a pre-determined number of pairs of ants (to work on each side of the line). Each pair of ants collaborate in order to build a feasible balancing solution, i.e., an assignment of tasks to workstations on both sides of the line, in such a way that all constraints of the problem are verified (precedence, zoning, capacity, and synchronism). For each feasible solution obtained a measure of its quality is computed, according to the problem's objective function.

$$\textit{Maximise} \quad \lambda WE^{2s} - B_b^{2s} - B_w^{2s}$$

subject to

$$\sum_{b=L}^{R}\sum_{k=1}^{K} x_{ikb} = 1 \qquad (i = 1,...,N) \qquad (i)$$

$$\sum_{k=1}^{K} x_{ikL} = 1 \qquad (i \in S_L) \qquad (ii\,\text{a})$$

$$\sum_{k=1}^{K} x_{ikR} = 1 \qquad (i \in S_R) \qquad (ii\,\text{b})$$

$$\sum_{b=L}^{R}\sum_{k=1}^{K} k x_{ikb} T_i + \max_m\{t_{im}\} - \sum_{b=L}^{R}\sum_{k=1}^{K} k x_{jkb} T_j \le 0 \qquad (i = 1,...,N; j \in Suc_i) \qquad (iii)$$

$$\sum_{k=1}^{K} k x_{ikL} T_i - \sum_{k=1}^{K} k x_{jkR} T_j = 0 \qquad ((i,j) \in S_C) \qquad (iv)$$

$$\sum_{b=L}^{R}\sum_{k=1}^{K} k x_{ikb} T_i - \sum_{b=L}^{R}\sum_{k=1}^{K} k x_{jkb} T_j \le \mathcal{M} u_i \qquad (i = 1,...,N; j \notin Suc_i) \qquad (v\,\text{a})$$

$$\sum_{b=L}^{R}\sum_{k=1}^{K} k x_{jkb} T_j - \sum_{b=L}^{R}\sum_{k=1}^{K} k x_{ikb} T_i \le \mathcal{M}(1\text{-}u_i) \qquad (i = 1,...,N; j \notin Suc_i) \qquad (v\,\text{b})$$

$$\sum_{i=1}^{N} t_{im} x_{ikb} + s_{kbm} = C \qquad (k = 1,..K; b = L,R; m = 1,...,M) \qquad (vi\,\text{a})$$

$$x_{ikb}\left(T_i + \max_m\{t_{im}\}\right) \le C \qquad (i = 1,...,N; k = 1,...,K; b = L,R) \qquad (vi\,\text{b})$$

$$\sum_{k=1}^{K} k x_{ikb} - \sum_{k=1}^{K} k x_{ikb} = 0 \qquad ((i,j) \in ZP; b = L,R) \qquad (vii)$$

$$\sum_{k=1}^{K} k x_{ikb} - \sum_{k=1}^{K} k x_{ikb} \ne 0 \qquad ((i,j) \in ZN; b = L,R) \qquad (viii)$$

$$\sum_{i=1}^{N} x_{ikb} \le \mathcal{M} y_{kb} \qquad (k = 1,..K; b = L,R) \qquad (ix\,\text{a})$$

$$\sum_{i=1}^{N} x_{ikb} \ge y_{kb} \qquad (k = 1,..K; b = L,R) \qquad (ix\,\text{b})$$

$$S = \sum_{k=1}^{K}\sum_{b=L}^{R} y_{kb} \qquad (ix\,\text{c})$$

$$s_{kbm} \ge 0 \qquad (k = 1,..K; b = L,R; m = 1,...,M) \qquad (x\,\text{a})$$

$$x_{ikb} \in \{0,1\} \qquad (i = 1,...,N; k = 1,...K; b = L,R) \qquad (x\,\text{b})$$

$$u_i \in \{0,1\} \qquad (i = 1,...,N) \qquad (x\,\text{c})$$

$$y_{kb} \in \{0,1\} \qquad (k = 1,..K; b = L,R) \qquad (x\,\text{d})$$

$$S > 0 \qquad S \text{ is integer} \qquad (x\,\text{e})$$

Figure 6.3– Mathematical programming model for the 2-MALBP

After all pairs of ants of the sub-colony have generated a solution, they release a certain amount of pheromone according to the quality of the solution. The sequence in which the tasks are performed at both sides of the line will determine the pheromone trails. If task *j* is performed immediately after task *i*, then a certain amount of pheromone is released between task *i* and *j*. Hence, pheromone trails are built in the paths used by the ants to build the balancing solution.

The procedure is repeated for every sub-colony within the ant colony. The best solution found by the procedure is updated after each sub-colony's iteration.

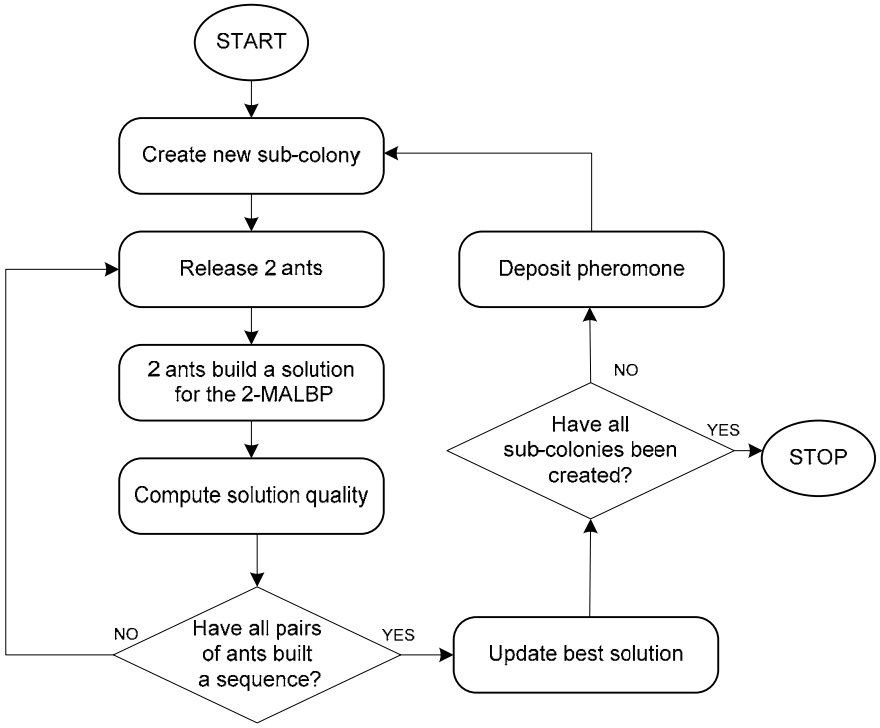

Figure 6.4 – Outline of 2-ANTBAL

6.4.1 Building a balancing solution

An outline of the way the two ants build a balancing solution is presented in Figure 6.5. The procedure starts by initialising the current time of both side-ants (*ct(aS)* is the current time of one side-ant and *ct(aS)* is the current time of the opposite side-ant) and it randomly selects one of the sides of the line to begin the assignment. Then, the corresponding side-ant opens a workstation and determines the set of available tasks, according to the conditions described in the following section.

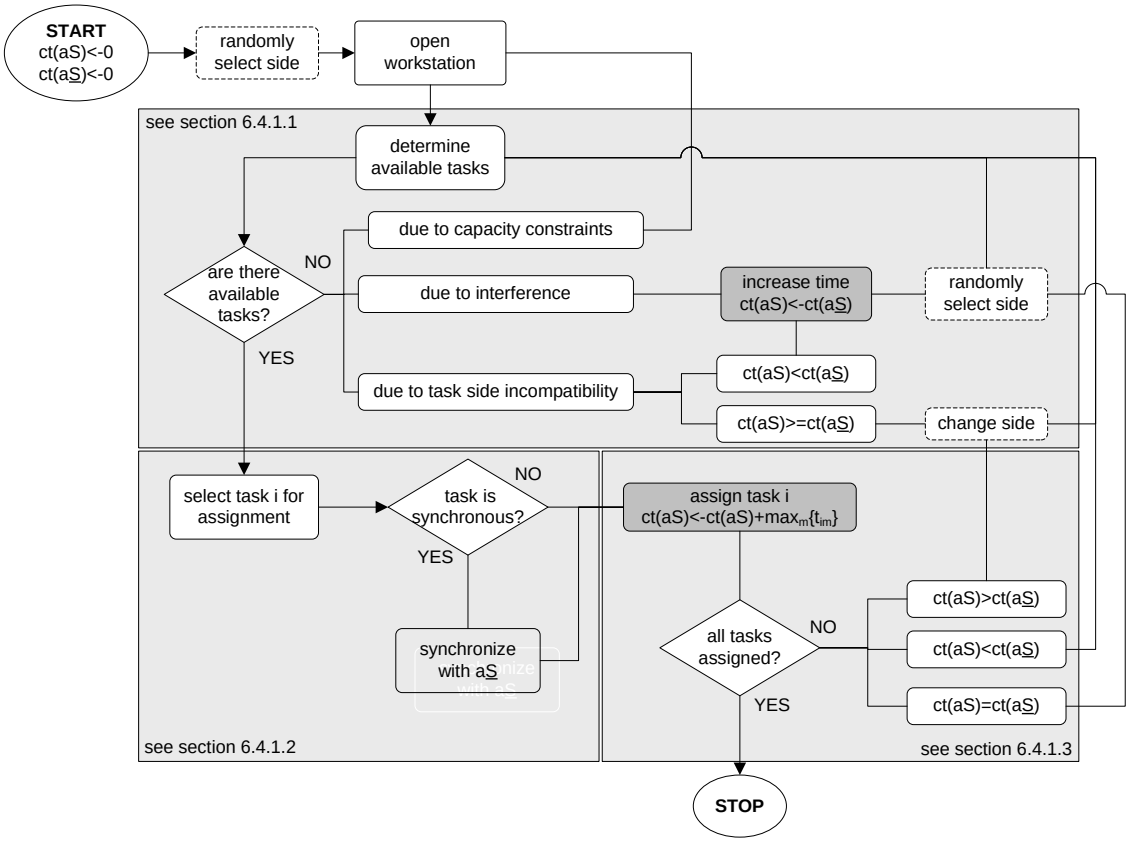

Figure 6.5 – Building a balancing solution for the 2-MALBP

6.4.1.1 Available tasks

The available tasks are the set of tasks that can be assigned to a particular workstation starting at the current time. A task is *available* if it verifies all the following conditions:

(i) the task side is the same as the current side or the task can be performed on either side;

(ii) the task predecessors are assigned to an earlier time (if a predecessor is assigned to the opposite side it must be completed before the current time);

(iii) assigning the task to the current workstation does not violate the capacity (i.e., cycle time) constraints;

(iv) assigning the task to the current workstation does not violate zoning constraints;

(v) if the task has synchronism constraints, it is possible to assign its mated-task to the opposite side of the line, starting at the same time.

If the side-ant does not find any available tasks, it must detect the causes and proceed accordingly, namely:

- Capacity constraints violation occurs when no available task fit the current workstation. In this case the side-ant opens a new workstation.
- Interference problems occur when there are tasks whose predecessors have been assigned to the opposite side but will be finished in a forward time. To deal with the interference issue, side-ants use a timeline when building the balancing solution in order to coordinate the task assignment. When interference occurs, the side-ant must move its current time forward, to the opposite side-ant current time. Then, the procedure randomly selects which side-ant will continue the assignment and therefore determine again the set of available tasks.
- Task side incompatibility occurs when there are no tasks that can be assigned to the current side. This results from one of the following reasons:
  - the current time of the side-ant is inferior to the current time of the opposite side-ant ($ct(aS)<ct(a\underline{S})$). In this case the side-ant must move its current time forward, to the opposite side-ant current time, and then a side is selected randomly to continue;
  - the current time of the side-ant is equal or greater than the current time of the opposite side-ant ($ct(aS)\geq ct(a\underline{S})$). In this case the opposite side-ant takes control of the assignment procedure.

From the set of available tasks, a side-ant must select one to be assigned to the current workstation, starting at the current time. The selection of tasks for assignment is described in the following section.

6.4.1.2 Selecting a task for assignment

Similarly to ANTBAL, two types of rules are used in the proposed procedure: static and dynamic. At the beginning of the procedure, a static priority rule is randomly assigned to each pair of ants. The static priority rules used in the proposed procedure are (i) maximum processing time (for all models), (ii) maximum average processing time, (iii) maximum ranked positional weight, (iv) maximum number of direct successors and (v) maximum total number of successors.

The dynamic priority rules are introduced in the task selection process, while the side-ants are building the solution. The dynamic rules recalculate the parameters after each task is assigned to a workstation, allowing an adaptation of the procedure to the characteristics of the already built part of the balancing solution.

The first dynamic rule is called 'last task becoming available' and it was previously developed for the MALBP. It deals with the work relatedness issue, favouring the assignment of the direct successors of a task immediately after that task has been assigned, by attributing them the highest priority value in the subsequent assignment iteration.

The second dynamic rule was especially developed for the 2-MALBP and it seeks to facilitate the assignment of tasks that must be performed simultaneously at both sides of the line. This rule is called 'predecessor of mated-task'. When a task with synchronism constraints becomes available for assignment at one side of the line, this rule is activated and the predecessors of the mated-task become high priority tasks, being preferably assigned. This allows the assignment of the synchronous tasks as earlier as possible.

The values of the priority rules will vary between 1 for the task with lowest priority and N (number of tasks) for the task with highest priority, and will be the heuristic information used by the ants to select the tasks. From the set of available tasks, the side-ant selects one task for assignment to the current workstation, according to a selection rule that takes into account (i) the pheromone trail intensity between the previously selected task and each available task, and (ii) the heuristic information about each available task. A side-ant s which has selected task i in the previous iteration will select task j by applying the following rule:

$$j = \begin{cases} J_1 = \arg\max_{j \in A_i^s} \left\{ [\tau_{(i,j)}]^{\alpha} [\eta_j]^{\beta} \right\} & \text{if } r \le r_1 \quad (\textit{exploitation}) \\ J_2 : p_{(i,J_2)} = \dfrac{[\tau_{(i,J_2)}]^{\alpha} [\eta_{J_2}]^{\beta}}{\sum_{j \in A_i^s} [\tau_{(i,j)}]^{\alpha} [\eta_j]^{\beta}} & \text{if } r_1 \le r \le r_2 \quad (\textit{biased exploration}) \\ J_3 : \text{random selection of } j \in A_i^s & \text{if } r_2 \le r \le r_3 \quad (\textit{random selection}) \end{cases} \qquad \textbf{(6.20)}$$

where

- r is a random number between 0 and 1 and r_1, r_2 and r_3 three user-defined parameters such that $0 \le r_1, r_2, r_3 \le 1$ and $r_1 + r_2 + r_3 = 1$ (by default r_1=0.6, r_2=0.3, r_3=0.1);

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

Similarly to the procedure developed for one-sided assembly lines, the selection of a task from the set of available tasks is performed by one of three strategies:

- Exploitation: it determines the selection of the best task according to the values of $[\tau_{(i,j)}]^{\alpha}[\eta_j]^{\beta}$.
- Biased exploration: a task is selected with a probability of $p_{(i,j)}$ as given by J_2 in equation (6.20).
- Random selection: from the set of available tasks, the side-ant selects one at random.

6.4.1.3 Assigning tasks to workstations

In the proposed procedure, side-ants use a timeline to build the balancing solution. Every time a side-ant assigns a task to a workstation, its current-time is increased an amount corresponding to the task processing time. Considering the mixed-model nature of the problem, this time will be the maximum processing time of that task for all models, in order to ensure that the cycle time is always met, regardless of the model being assembled. Then the current times of both side-ants are compared, resulting in the following courses of action:

(i) if the current time of the side-ant is inferior to the current time of the opposite side-ant (*ct(aS)<ct(aS)*), the assignment continues on the same side.

(ii) if the current time of the side-ant is superior to the current time of the opposite side-ant (*ct(aS)>ct(aS)*), the side is changed.

(iii) if the current time of the side-ant is equal to the current time of the opposite side-ant (*ct(aS)=ct(aS)*), a side is randomly selected to continue the assignment.

When all tasks have been assigned to workstations, the balancing solution is completed and solution quality is evaluated using the objective function of the mathematical programming model for the 2-MALBP: $Z = \lambda WE^{2s} - B_b^{2s} - B_w^{2s}$.

6.4.2 Pheromone release strategy

The updating of the pheromone trails between tasks is performed at the end of each sub-colony iteration. First, a portion of the existing pheromone value is evaporated in all paths. Then, each side-ant *s* of the *n* pairs of side-ants that constitute the sub-colony releases an amount of pheromone in the paths used to build its task sequence, according to the corresponding balancing solution quality. This amount of pheromone is given by:

$$\Delta\tau^{ns}{}_{(i,j)} = \begin{cases} Z, \text{if task } j \text{ is performed immediatly after task } i, \\ \text{in the solution built by side - ant } s \text{ of pair } n \\ 0, \text{otherwise} \end{cases} \quad \textbf{(6.21)}$$

The overall pheromone update effect of the *n* pairs of side-ants in each path *(i,j)* is then:

$$\tau_{(i,j)} \leftarrow \tau_{(i,j)} + \sum_{n=1}^{N}\sum_{s=L}^{R}\Delta\tau^{ns}{}_{(i,j)} \quad \textbf{(6.22)}$$

6.4.3 Numerical example

In this section, a numerical example, with the following characteristics, is used to illustrate some features of 2-ANTBAL.

- Two models, A and B, are simultaneously assembled in a line over a planning horizon of 480 t.u. (time units). The demand for each model is, respectively 10 and 14 units (the cycle time is then C=20, q_A=42% and q_B=58%).
- The combined precedence diagram is the one depicted in Figure 6.2. Table 6.1 shows the task processing times for the two models.
- Tasks 9 and 10 are synchronous tasks – they must be performed simultaneously at both sides of the line.

Table 6.1 – Task processing times of models A and B

Task	t_A	t_B
1	0	2
2	8	8
3	7	7
4	7	5
5	2	2
6	6	0
7	4	0
8	0	2
9	3	2
10	3	2
11	6	6
12	3	3
13	5	5
14	4	6

Figure 6.6 represents a balancing solution, for the numerical example, built by a pair of side ants. It represents the sequence of tasks performed at the workstations of each side of the line (R-right side, L-left side). Inside the rectangles are the task numbers and shaded areas correspond to idle time.

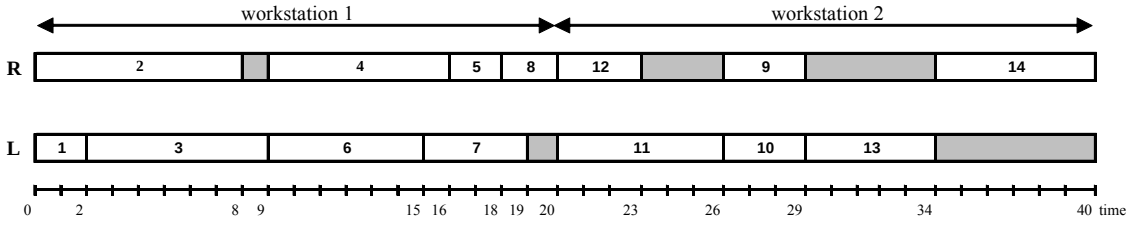

Figure 6.6 – Representation of a balancing solution for the 2-sided line

The actions performed by the pair of side-ants at each moment while building the balancing solution are described in Table 6.2. Both ants work simultaneously, but when they need to perform actions at the same instant, the procedure randomly selects one of them to do it in the first place, because the decisions of one side-ant will have consequences on the decisions of the other.

Table 6.2 – Actions of the side-ants to build a balancing solution

	current time	Left-ant	Right-ant
Workstation 1	0	► selects task 1	► selects task 2
	2	► selects task 3	
	8		► no available tasks due to interference
	9	► selects task 6	► selects task 4
	15	► selects task 7	
	16		► selects task 5
	18		► selects task 8
	19	► no available tasks due to capacity constraints	
Workstation 2	20	► selects task 11	► selects task 12
	23		► no available tasks due to synchronism constraints
	26	► selects task 10	► selects task 9
	29	► selects task 13	► no available tasks due to interference
	34	► no available tasks due to task side	► selects task 14
	40	*Complete solution*	

6.5 Computational experience

The heuristic was coded in C and run on a 2.8 GHz Pentium 4 computer. To test its performance, a series of comparative tests were carried out by applying the heuristic to the benchmark problems A65, B148 and A205, proposed by Lee et al (2001), with 65, 148 and 205 tasks, respectively. For each of the test problems different values of the cycle time were used in order to provide a higher number of problem instances, a total of 22. These consider only the single-model 2-sided assembly line balancing problem of type I with no synchronism or zoning constraints. Table 6.3 shows the computational results. For each problem it presents the given cycle time (C) and the number of workstations obtained by the different tested procedures. Column *LB* presents the lower bound on the number of workstations for each of the test problems, given by:

$$LB = LB_{left} + LB_{right} + LB_{either} \quad \textbf{(6.23)}$$

LB_{left} and LB_{right} are the theoretical minimum number of the left and right side workstations, computed as follows:

$$LB_{left} = \left\lceil \sum_{i \in S_L} \max_m \{t_{im}\} \Big/ C \right\rceil \quad \textbf{(6.24)}$$

$$LB_{right} = \left\lceil \sum_{i \in S_R} \max_m \{t_{im}\} \Big/ C \right\rceil \quad \textbf{(6.25)}$$

The term LB_{either} adds up the number of workstations needed to process tasks of either side. Because these tasks can be included in workstations that perform left or right side tasks, it is necessary to verify if, after filling up these workstations, there are still either side tasks to create new workstations. The minimum number of workstations (LB_{either}) required to perform either side tasks, after filling up the remaining capacity of workstations assigned to left and right side tasks, is given by:

$$LB_{either} = \left\lceil \left\lfloor \sum_{i \in S_E} \max_m \{t_{im}\} - \left(\left(LB_{left} + LB_{right} \right) \cdot C - \sum_{i \notin S_E} \max_m \{t_{im}\} \right) \right\rfloor \Big/ C \right\rceil \quad \textbf{(6.26)}$$

Columns *H* and *G* present the results reported by Lee et al (2001) concerning heuristic rules and group assignment, respectively. Columns *Mean*, *Min* and *Max* present the average, minimum and maximum values of the number of workstation of the best solution found by 2-ANTBAL computed from 10 runs of each instance of the problem. Finally, the last two columns present a comparison of the performance of 2-ANTBAL with (i) procedure *G* (as it is better than procedure *H)*: $Imp_G(\%)$ is the average improvement of 2-ANTBAL compared with *G* and (ii) the lower bound: $Dev_{LB}(\%)$ is the difference between the minimum value obtained by 2-ANTBAL and *LB*.

The computational results show that the proposed procedure 2-ANTBAL clearly outperforms both of the procedures presented by Lee et al (2001) considering the number of workstations of the 2-sided assembly line. This fact is particularly evident in problem A205 where the improvement values of 2-ANTBAL reach 16.7%. Negative improvement values are explained by the fact that the mean value of the number of workstations obtained by 2-ANTBAL was slightly superior to the mean value of procedure *G*. However, in both cases the minimum value of 2-ANTBAL was either equal or inferior to the mean value of *G*.

The values in bold are equal to the lower bound of the problem instance. This means that guaranteed optimal solution was reached for 10 instances. Also, the maximum difference between the best solutions obtained by 2-ANTBAL and the lower bound was only 11.1%, which supports the good performance of the proposed procedure.

Table 6.3 – Results of the computational experience for 2-ANTBAL

			Lee et al (2001)		**2-ANTBAL**			Imp_G	Dev_{LB}
Problem	*C*	*LB*	*H*	*G*	*Mean*	*Min*	*Max*	*(%)*	*(%)*
A65	326	16	17.7	17.4	17.0	17	17	2.3	6.3
	381	14	15.7	15.0	14.8	**14**	15	1.3	**0**
	435	12	14.0	13.4	13.0	13	13	3.0	8.3
	490	11	12.1	12.0	12.0	12	12	0.0	9.1
	544	10	11.5	10.6	10.8	**10**	11	-1.9	**0**
B148	204	26	27.8	27.0	26.0	**26**	26	3.7	**0**
	255	21	22.0	21.0	21.0	**21**	21	0.0	**0**
	306	17	19.3	18.0	18.0	18	18	0.0	5.9
	357	15	16.0	15.0	15.4	**15**	16	-2.7	**0**
	408	13	14.0	14.0	14.0	14	14	0.0	7.7
	459	12	12.1	13.0	12.0	**12**	12	7.7	**0**
	510	11	12.0	11.0	11.0	**11**	11	0.0	**0**
A205	1133	21	24.0	23.0	22.4	22	23	2.6	4.8
	1322	18	21.9	20.7	20.0	20	20	3.4	11.1
	1510	16	18.7	20.0	17.2	17	18	14.0	6.3
	1699	14	16.7	16.0	15.8	15	16	1.3	7.1
	1888	13	15.4	16.0	13.8	**13**	14	13.8	**0**
	2077	12	14.0	14.0	12.0	**12**	12	14.3	**0**
	2266	11	12.5	13.0	12.0	12	12	7.7	9.1
	2454	10	12.0	12.0	10.0	**10**	10	16.7	**0**
	2643	9	11.2	12.0	10.0	10	10	16.7	11.1
	2832	9	10.0	10.0	10.0	10	10	0.0	11.1

6.6 Chapter conclusions

In this chapter, the mixed-model 2-sided assembly line balancing problem was addressed. A mathematical programming model was used to formally describe the problem and an ant colony optimisation algorithm was developed to solve it. In the proposed procedure, two ants work simultaneously one at each side of the line aiming to efficiently coordinate the assignment of tasks to both of the sides of the line.

The procedure uses a non-delay rule for ants while available tasks exist, however this may discard optimal solutions since it may be best to wait a brief time for the opposite ant and then perform a mated-task. Future improvements of the algorithm should only enforce the non-delay rule with some degree of probability at each stage.

The good performance of the algorithm was proved by computational experiments with a set of benchmark problems from the literature. These problems were for single-model assembly lines, so additional computational tests should be made using mixed-model instances in order to evaluate the features of the procedure specifically developed to address 2-sided mixed-model assembly line balancing problems.

7

Real world applications

Contents

7.1 Chapter introduction

This set of short case studies is the outcome of a business internship program sponsored by the University of Aveiro for the students in the last year of the Industrial Management and Engineering program. Four students working as trainees in four companies had to analyse the operation of assembly lines and propose changes in order to improve the line's performance.

With the information gathered by the trainees, some of the developed procedures were adapted and applied to the real assembly lines, aiming to improve the existing assignment of tasks to workstations. The success of these applications depended on the level of interaction with the trainee, as a deep knowledge of the assembly process is required to adapt the algorithms to the real conditions of the assembly line.

7.2 Case 1 – Combining heuristic procedures and simulation models for balancing a PC camera assembly line*

The company in which the study took place is a major manufacturer of consumer electronic goods and goal of the project was to analyse and improve the performance of a PC camera assembly line.

The assembly line under analysis is used to assemble three different versions of a PC camera with some dissimilar technical specifications, thus, a mixed-model assembly line. Most of the tasks required to complete the assembly of the models are manual and only the final tasks (testing operations) are performed automatically by a computer. The line employs low skilled labour which is cross-trained to perform all the operations and, as a result, it is relatively easy to rebalance the line and change its configuration. However, there is a high level of absenteeism among the workforce and consequently the line managers' need to rebalance the line on a daily basis.

On the other hand, the great variability and uncertainty associated with the product demand levels is a major problem that the company has to deal with, requiring frequent

* The work presented in this section is published in Mendes et al (2005)

changes in the line configuration. The ability to quickly manage the assembly line to compensate for changes in both the labour workforce and market demand is becoming an important competitive factor. Consequently, the company must be agile to implement changes in a quick and effective way.

To support the operational decisions of the line managers, in light of the issues stated above, a combination of analytical and simulation models was developed in this project. The main goal of the study was to produce a set of assembly line configurations for different levels of demand, making it easier to rebalance the line according to the circumstances of a specific planning horizon.

In the first phase of the study, a heuristic procedure was used to solve the mixed-model assembly line balancing problem and derive line configurations, with a minimum number of workstations and a smooth workload balance between and within the workstations, for the relevant levels of demand. The heuristic procedure was the simulated annealing based approach described in section 4.3. In the second phase, the solutions provided by the heuristic procedure were used as an input to discrete event simulation models in order to test the robustness of these solutions when variability was introduced in some of the design parameters (e.g., stochastic task times). Different performance measures, like flow times and resources utilisation, were derived from the simulation models helping the decision maker to fine-tune the suggested line configurations.

The following sections describe the characteristics of the assembly line under analysis and the procedures developed to achieve the goals of the study. The preliminary results of this study were presented in an international conference (Ramos et al, 2001) and the final results enabled the publication of a paper in a scientific journal (Mendes et al, 2005).

7.2.1 The PC camera assembly line

The assembly line under analysis is used to assemble three different versions of a PC camera (model A, model B and model C), with some dissimilar technical specifications. Figure 7.1 shows an exploded view of the PC camera. The PCB (printed circuit board) is the only camera part that is manufactured in the facility, while all the other components are outsourced.

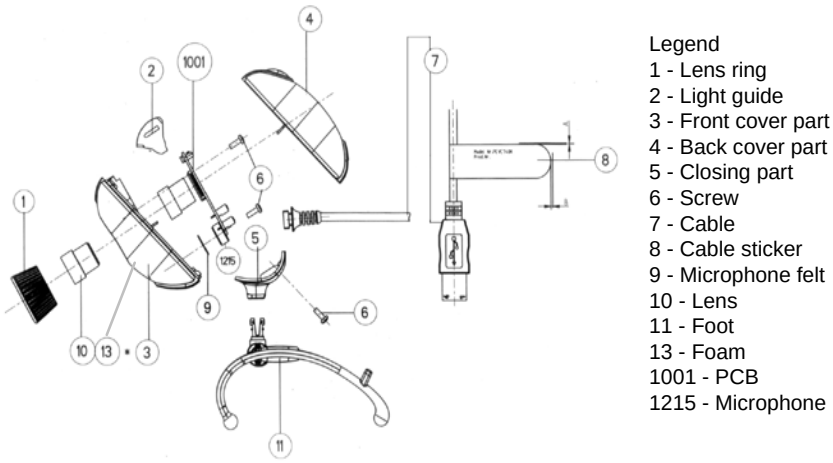

Figure 7.1 – Exploded view of the PC camera

The assembly line is composed by a sequence of workstations performing manual operations and an automated conveyor that transports the sub-assemblies along the process. The assembly of a PC camera requires the following steps:

(i) cutting of collective PCBs;
(ii) soldering of microphone pins;
(iii) cleaning, inspecting and soldering of electronic components;
(iv) functional testing;
(v) soldering of cable connector;
(vi) attaching lens into front cover part;
(vii) placing and screwing individual PCB at front cover;
(viii) attaching back cover part and screwing closing part;
(ix) final testing;
(x) encasing lens ring, putting foot into camera and placing sticker on the cable.

When a camera is complete it proceeds to the packaging table where some other tasks are performed, namely, cleaning and packaging the camera into its individual box with software and documentation, placing the closing sticker and the bar code and packing several individual boxes into a collective one. These collective boxes are then grouped in pallets and transported to the finished products warehouse.

The assembly process for each model defines: (i) the task processing times and (ii) a set of precedence relationships, which determine the sequence in which the tasks can be

performed. As a large subset of tasks is common to all models, the precedence diagrams of the three models were combined and the resulting one accounts for all the tasks required to assemble all the models. The combined precedence diagram for the three PC camera versions is depicted in Figure 7.2 and the standard task processing times for each model in time units (t.u.) are shown in Table 7.1.

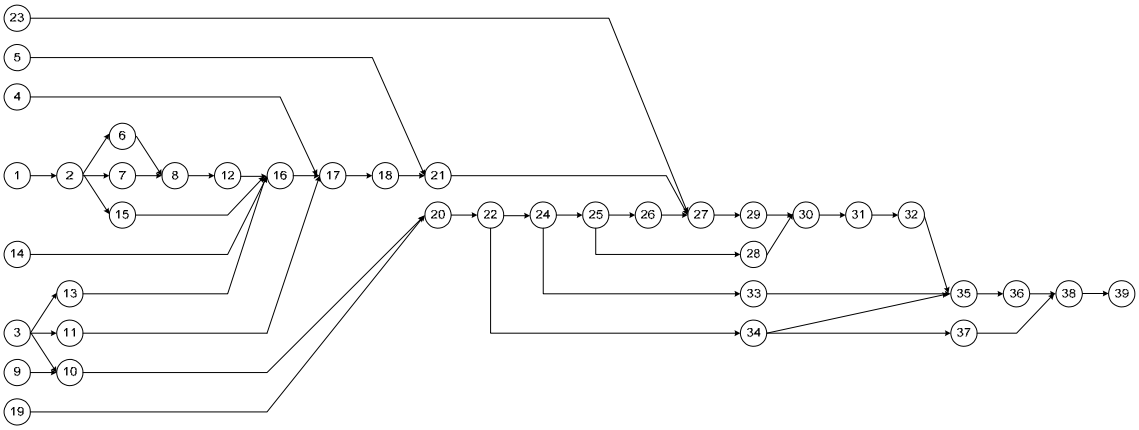

Figure 7.2 – Combined precedence diagram for the three PC camera models

Table 7.1 – Task processing times

Task	t_A	t_B	t_C	Task	t_A	t_B	t_C	Task	t_A	t_B	t_C
1	2	2	2	**14**	0	4	4	**27**	5	5	5
2	2	2	2	**15**	9	9	0	**28**	0	0	2
3	2	2	2	**16**	13	13	12	**29**	1	1	1
4	2	2	2	**17**	6	6	6	**30**	3	3	3
5	2	2	2	**18**	7	7	7	**31**	3	3	3
6	0	11	11	**19**	3	3	3	**32**	0	0	3
7	0	0	16	**20**	28	37	33	**33**	4	4	4
8	21	39	37	**21**	3	3	3	**34**	2	2	2
9	2	2	2	**22**	8	8	8	**35**	2	2	2
10	10	10	10	**23**	5	5	5	**36**	1	1	1
11	3	0	0	**24**	7	7	9	**37**	1	1	1
12	11	11	11	**25**	4	4	4	**38**	1	1	1
13	4	4	4	**26**	6	6	6	**39**	1	1	1

Tasks 8 and 20 are inspection operations performed automatically (i.e., without the operator intervention) after being set-up by the operator. The set-up time for these operations is 5 t.u.. So, if these operations are carried out simultaneously with other tasks

in the same workstation, only the set-up time needs to be accounted for, as the operator will be available to perform those tasks.

Some tasks cannot be performed in the same workstation (incompatible tasks) due to physical or process related constraints. For example, soldering and packaging tasks are incompatible due to ergonomic issues: soldering requires the operator to be seated, while packaging requires the operator to be standing. The set of pairs of incompatible tasks is presented in Table 7.2.

Table 7.2 – Set of pairs of incompatible tasks

Pair	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Task 1	6	7	7	8	8	10	18	19	20	20
Task 2	8	8	20	12	20	20	20	20	24	34

7.2.2 Balancing the mixed-model assembly line

The simulated annealing procedure developed to solve the mixed-model assembly line balancing problem, described in section 4.3, was adapted to address the PC camera mixed-model assembly line. The goal of this stage of the study was to derive line configurations (i.e., balancing solutions), for the relevant levels of demand, with a minimum number of operators and a smooth balance of workloads between and within workstations.

The typical lot sizes of each model for different demand levels (low, medium and high) are presented in Table 7.3. For each model, it presents the number of units to be produced according to the demand level, over a planning horizon of P=132 900 t.u..

Table 7.3 – Number of units to be produced for each demand level

	Product demand		
Demand level	**D_A**	**D_B**	**D_C**
Low (L)	1610	390	1670
Medium (M)	3150	1330	3130
High (H)	5230	2720	6580

In order to derive line configurations able to cope with the most frequent demand situations, five different demand scenarios were provided by the line manager and were used as input for the heuristic procedure. Table 7.4 shows the five scenarios and the value of the cycle time for each of them, computed by $\left(C = P / \sum_{m=1}^{M} D_m \right)$.

Table 7.4 – Demand values and cycle times for the different production scenarios

	Product demand			
Scenario	**D_A**	**D_B**	**D_C**	**C**
1 (LLL)	1610	390	1670	36.2
2 (MMM)	3150	1330	3130	17.5
3 (HHH)	5230	2720	6580	9.1
4 (MLM)	3150	390	3130	19.9
5 (MMH)	3150	1330	6580	12.0

As the processing time of some tasks is higher than the required cycle time, it is necessary to allow the use of parallel workstations. The minimum replication time (*MRT*), i.e., the processing time that triggers the replication process, was set to *MRT=C*. This means that only workstations performing tasks with processing time higher than the cycle time for, at least, one of the models, are allowed to work in parallel.

After setting all the data, the simulated annealing based heuristic was then used to provide balancing solutions for the different demand scenarios. For scenario 1, the initial solution, built by the modified ranked positional weight technique, is depicted in Table 7.5, where the first column represents the workstation index, the second column shows the set of tasks assigned to each workstation and the third column shows the number of replicas of each workstation (parallel workstations have more than one replica of the workstation).

In this case study, there were special task related conditions that had to be taken into account when applying the heuristic: tasks 8 and 20 are related to testing operations and while the test program is running the assigned operator can execute other tasks simultaneously. The heuristic was therefore modified to in order to reproduce, as closely as possible, this issue.

After the initial solution is obtained, the procedure tries to improve its number of workstations through a simulated annealing approach, in which the neighbouring solutions are generated by (i) swapping two tasks in different workstations or (ii) transferring a task to another workstation. At the end of this stage, the best solution has the lowest number of workstations. For the demand scenario 1 the heuristic could not improve the number of workstations of the initial solution.

Table 7.5 – Initial solution for scenario 1

Workstation	Tasks	Replicas
1	1,2,3,4,6,7,11,15	1
2	5,8,9,10,13,14,19,23	2
3	12,16,17	1
4	18,21	1
5	20,22	2
6	24,25,26,27,28,29,30,31,34,37	1
7	32,33,35,36,38,39	1

The second stage of the simulated annealing based procedure aims to balance the workloads between and within the workstations and it starts with the best solution found at the end of the first stage. For demand scenario 1 this solution has 9 workstations (including parallel workstations), a workload balance between workstations of B_b=0.15 and a workload balance within workstations of B_w=0.38. At the second stage the initial number of workstations cannot be exceeded and, if possible, may be improved. Swap and transfer movements are also performed, but the tasks and workstations involved in these movements are selected to foster improving solutions considering workload smoothing (see section 4.3.2.1). At the end of this stage, the best solution has the minimum number of workstations and the best workload balance between and within workstations. At each iteration, the procedure verifies precedence, incompatibility and capacity constraints, in order to always generate feasible solutions. The best solution found for the demand scenario 1 has workload balances of B_b=0.06 and B_w=0.13, which shows an improvement of 50% in the objective function B_b+B_w.

The heuristic was used to derive line configurations for the five demand scenarios and the final balancing solutions are depicted in Table 7.6.

Table 7.6 – Final line configurations for the different demand scenarios

Scenario 1 - LLL

W	Tasks	R
1	1,2,6,7	1
2	3,4,8,9,10,14,15,19	2
3	12,13	1
4	11,16,17,18	1
5	5,20,21,22,23	2
6	24,25,33	1
7	26,27,28,29,30,31,32,34, 35,36,37,38,39	1

Scenario 2 - MMM

W	Tasks	R
1	1,2,6	1
2	7,15	1
3	3,4,8,9,10,11,13,14,19,23	3
4	12	1
5	5,16	1
6	17,18	1
7	20,21,22	3
8	24,25,34,37	1
9	26,27,28,29,30	1
10	31,32,33,35,36,38,39	1

Scenario 3 - HHH

W	Tasks	R
1	1,2,6	2
2	7,15	2
3	3,4,8,9,10,13,14,19,23	5
4	12	2
5	11,16	2
6	17	1
7	5,18	1
8	20,22	5
9	24	1
10	25,28,34	1
11	21,26	1
12	27,29,30	1
13	31,33	1
14	32,35,36,37,38,39	1

Scenario 4 - MLM

W	Tasks	R
1	1,2,6	1
2	7,15	1
3	3,8,9,10,13,14,19	2
4	11,12	1
5	4,16	1
6	17,18	1
7	5,20,22,23	2
8	21,24,25,28	1
9	26,27,29,33,34	1
10	30,31,32,35,36,37,38,39	1

Scenario 5 - MMH

W	Tasks	R
1	1,2,7	2
2	6	1
3	3,8,9,10,13,14,15,19,23	4
4	12	1
5	11,16	2
6	4,17	1
7	18	1
8	5,20,21,22	4
9	24,34,37	1
10	25,26,28	1
11	27,29,30,31	1
12	32,33,35,36,38,39	1

The cycle times for each demand scenario were recomputed taking into account the task assignments provided by the heuristic. The effective cycle time of a balancing solution is the sum of the processing times of the workstation with the maximum workload. Table 7.7 presents the theoretical and the real values of the cycle time for each scenario. The reduction of the value of the cycle time means an increase of the production rate, i.e., with the same assembly system it is possible to produce more than the number of units defined by the demand level.

Table 7.7 – Comparison of theoretical and real cycle times

	Cycle time (t.u.)	
Scenario	**Theoretical**	**Real**
1	36.2	31.0
2	17.5	17.0
3	9.1	9.0
4	19.9	18.0
5	12.0	12.0

In order to evaluate the balancing solutions considering the number of operators, the lower bound for the mixed-model assembly line balancing problem with parallel workstations (LB_{pmix}), proposed by Vilarinho and Simaria (2002) and presented in Appendix 3, was adapted to take into account a maximum of five replicas of one workstation, as the original version of the LB_{pmix} considered a maximum of only two replicas. The details of the computation of this lower bound are given in Appendix 5. As one can observe from Table 7.8, the solutions obtained by the heuristic are optimal for four of the scenarios.

Table 7.8 – Comparison of solutions with the lower bounds (LB_{pmix})

	Number of operators	
Scenario	**Solution**	**LB_{pmix}**
1	9	8
2	14	14
3	26	26
4	12	12
5	20	20

The line configurations provided by heuristic for the different demand scenarios were used as an input for simulation models, which are able to include randomness and uncertainty in the analysis of the PC camera mixed-model assembly line. As the development of these models is not within the scope of this dissertation (as it was developed by another person), only a very brief description of the work is presented in the following sections. A more detailed explanation is provided in the paper of Mendes et al (2005).

7.2.3 Development of the simulation models

The set of line configurations produced by the heuristic procedure was based on several operational parameters that do not mimic exactly the real system, mainly because the solutions obtained do not reflect the operational variability and randomness induced to the system by the manual operations and by other factors, like rework, which affect the regular system operation.

So, simulation models for the different line configurations were developed, in order to check their dynamic behaviour in the presence of modelling parameters that better describe the system dynamics. The complete assembly system was modelled: the assembly line, the packaging table and the material handling equipments and the following measures were used to evaluate the performance of the different simulation models:

(i) throughput (number of cameras assembled in the planning horizon);
(ii) flow time (for each product);
(iii) utilisation of resources (labour).

Previously to the development of the simulation models for the line configurations suggested by the heuristic, a simulation model of the actual assembly system was built. This model allowed (i) the better understanding of the actual assembly system operation, (ii) the validation of the assumptions used to build it and later included in the different models and (iii) the gaining of confidence of the decision makers regarding the used methodology.

In this particular case study, one of the members involved in the project worked fulltime on the facility, thus the process of input data collection and analysis was easily accomplished. Her presence on site was also crucial to obtain a clear definition of control and decision rules used in the daily operation of the assembly line. In addition, there were large amounts of historical data related to processing times of all the assembly and packaging tasks for each version of the product, enabling the fit of proper distributions to this data.

Some important data of the assembly process, which were not addressed by the heuristic, were included in the simulation models. Conveyor details (e.g., length, speed) were set as specified in technical documentation and accordingly to the cycle time. The number of units rejected at the inspecting and testing operations was modelled as a

percentage of the cameras processed. There was also a wide availability of historical data to determine the reject rates related to those operations. It was assumed that some of the rejected products could be repaired, re-entering the assembly process at fixed points.

The workforce consists of one shift working 8.5 hours a day, five days a week. The shift has fixed daily breaks for meals and work meetings. The production scheduling is made on a weekly basis and it is conveyed to the assembly line supervisor. The printed circuit boards are manufactured on the facility and, usually, they are ready to enter the assembly line when required. The outsourced materials are located near the corresponding assembly/packaging station and can be picked from stock when needed.

The simulation models were implemented using the Arena® simulation software. This software has a high capability to model manufacturing systems and embeds key technology for desktop application integration, enabling the use of existing enterprise models. It also includes tools to analyse input and output data.

Verification and validation are two important phases of the development of simulation models. Model verification deals with building the model right and ensures that the computer program of the computerised model and its implementation are correct. Model validation deals with building the right model and confirms that the simulation model behaves with satisfactory accuracy, i.e., it is consistent with the modelling objectives.

Different techniques were used to verify and validate the models as described in Mendes et al (2005). One of these was animation, which played an important role on the results presentation phase. With this technique, the operational behaviour of the assembly line is displayed graphically as the model evolves through time. Figure 7.3 shows two snap shots (assembly line and packaging table) of the three-dimensional animation model developed for the actual assembly line.

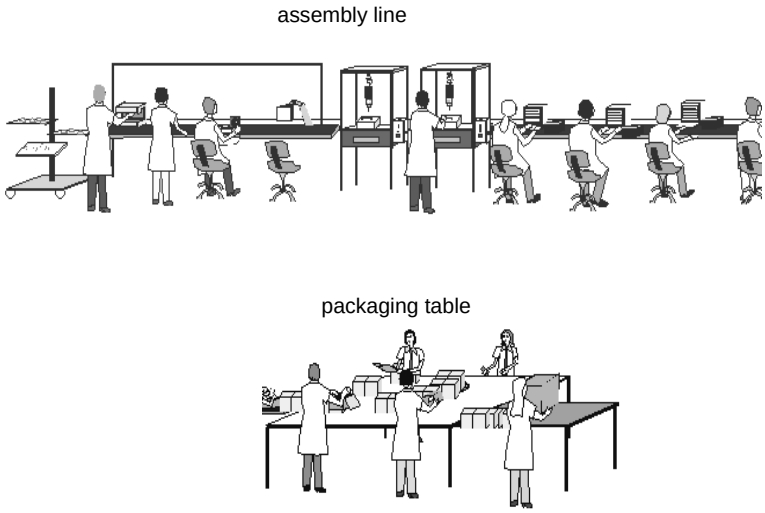

Figure 7.3 – Animation of the actual PC camera assembly system

Once again, the team member who took part in the project on site was crucial on the verification and validation process, as she combined the knowledge of using the simulation tool with the perception of the assembly process details.

7.2.4 Simulation experiment and results

The outcome of the simulation of the actual assembly line study showed that the estimates obtained for the selected performance measures were very similar to the real system measures and no major deviances between the simulation results and reality were found. The results also emphasised that the actual line was clearly unbalanced. Given these results, it was decided to go on to the second phase of the study, which aimed to build simulation models for the set of configurations, for the different levels of demand, generated by the heuristic procedure.

Regarding the throughput performance measures, the simulation results for these models showed that:

(i) the demand levels of scenarios 1 and 5 could be easily satisfied with the line configurations proposed by the heuristic procedure;

(ii) the forecasted demand for scenarios 2, 3 and 4 was not satisfied with the configurations proposed by the heuristic procedure.

The line bottlenecks were identified as being workstation 8 for scenario 2 and workstation 7 for scenarios 3 and 4. Several experimental tests were carried out in order to eliminate the bottlenecks. These tests included adjustments of the demand levels for the different models and parallelisation of the bottleneck workstations. The demand levels for the different scenarios were provided by the line manager as a guideline for typical production runs. Small adjustment (up to 5%) to these demand levels are allowed when leading to a reduction in the number of workers in the line. The demand levels were then adjusted in order to determine the number units of each model that could be produced with the configuration suggested by the heuristic procedure. As a significant reduction in the number of units to be assembled, for some of the models, was required, this course of action was abandoned.

On the other hand, the parallelisation of the bottleneck stations led to the desired production levels and, in some of the scenarios, some slack capacity was left available in the workstation for an eventual increase in the production rate. The values of the performance measures for the actual assembly system and for the line configurations for each scenario (with replicated bottleneck workstations in scenarios 2, 3 and 4) are shown in Table 7.9 (average flow time) and Figure 7.4 (average usage rate).

Table 7.9 – Simulation results for the average flow time

	Average flow time (t.u.)		
	Model A	**Model B**	**Model C**
Actual system	352.4	453.2	416.4
Scenario 1	207.6	437.4	401.3
Scenario 2	237.5	456.3	418.9
Scenario 3	227.9	483.9	407.6
Scenario 4	217.6	442.6	422.9
Scenario 5	220.3	487.9	438.9

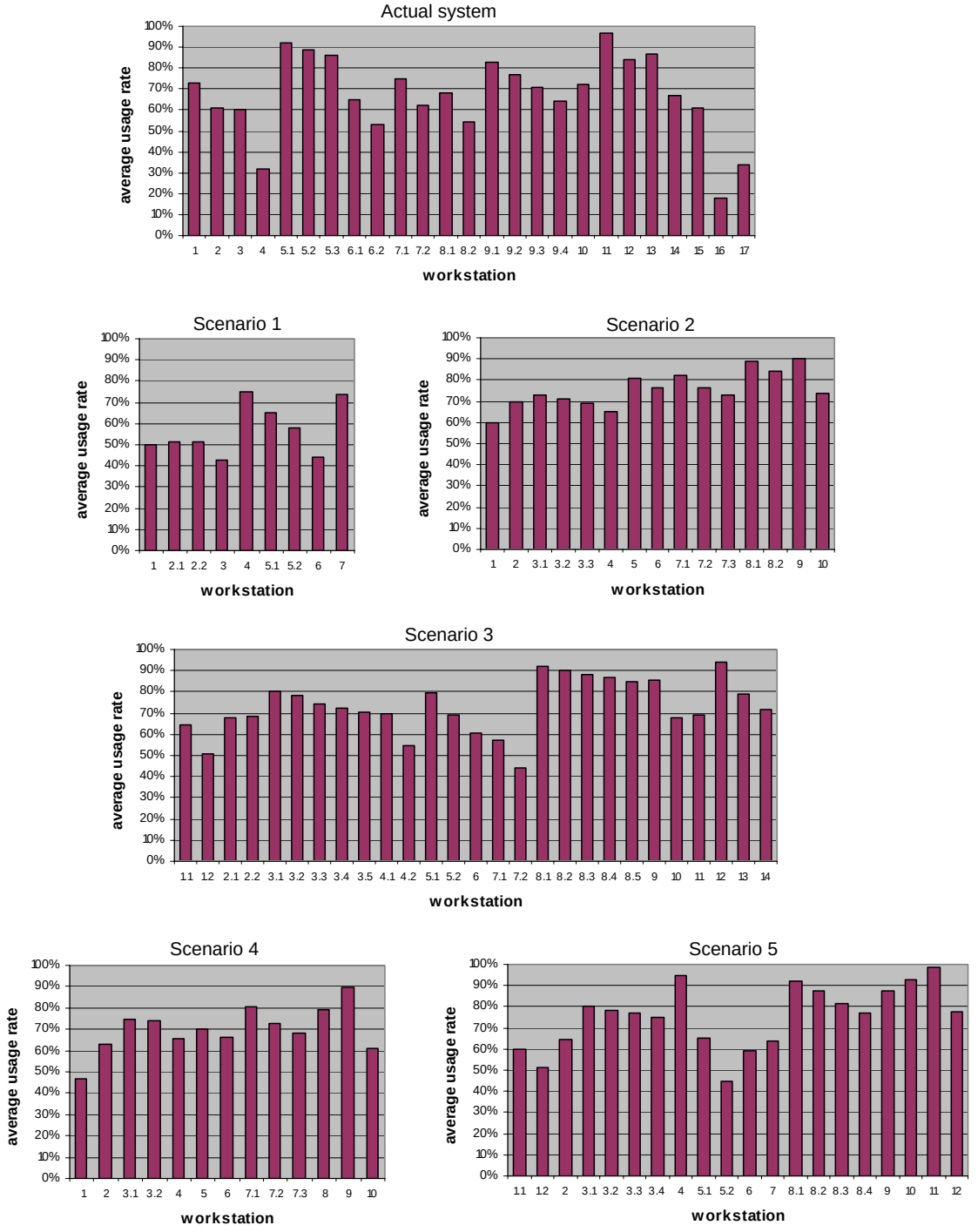

Figure 7.4 – Simulation results for the average usage rate

When the average flow time (i.e., the average time required to completely assemble one unit) of the actual system is compared to the different scenarios, one can notice that for model A the average flow time is reduced by 30% to 40%. As can be observed in Figure

7.4 for scenario 1, for example, when the demand is met there is still a slack capacity of around 20% in the most loaded workstation (workstation 4), so it is possible to increase production if required.

Table 7.10 shows the average usage rate and the correspondent standard deviation of the workstations for the actual system and for each of the scenarios. The average usage rate shows a reduction in the overall idle time for all of the scenarios, except scenario 1, in comparison with the actual system performance. The poorer performance of scenario 1 is justified by the fact that the line was simulated to produce 3.670 cameras (theoretical cycle time of C=36.2 t.u.) when the configuration provided by the heuristic could increase the output to 4.287 cameras (real cycle time of C=31 t.u.). The standard deviation explains how evenly split the workload is distributed across the workstations and all of the scenarios show an improvement over the actual system.

Table 7.10 – Average usage rate and standard deviation

	Usage rate	
	Average	**Std. Deviation**
Actual system	67%	0.19
Scenario 1	57%	0.12
Scenario 2	76%	0.08
Scenario 3	73%	0.13
Scenario 4	70%	0.11
Scenario 5	75%	0.15

These results showed that the configurations proposed by the heuristic procedure were suitable when the stochastic behaviour of the assembly system was addressed in the simulation models. In fact, with some adjustments to the solutions obtained for scenarios 2, 3 and 4:

(i) the desired levels of demand were satisfied;

(ii) the flow times for the three PC camera models and for the different scenarios were at acceptable levels and, for a particular case (Model A) were significantly reduced when compared to the actual system;

(iii) the workload of the proposed configurations is more evenly distributed.

It can be stated that the heuristic procedure provided good results that were easily fine-tuned using simulation. On the other hand, simulation allowed to gain the confidence of the

decision makers and to test different options to fine-tune the line configurations suggested by the heuristic, when more realistic parameters and operational details were introduced.

7.2.5 Conclusions

The results of this study were very useful for the line manager to define the line configurations for different demand scenarios by providing production run figures that maximise the use of the assembly line for these different scenarios. It can also be stated that the integrated approach used to design and analyse assembly line configurations is promising for this type of line.

From an academic perspective, this methodology makes a contribution to the literature because (i) it focuses attention on the joint use of analytical and simulation models to provide operational decision support for assembly line balancing and (ii) it demonstrates that when dealing with real-world problems, effective communication channels and company involvement are critical factors on the attainment of meaningful and in-depth results. In fact the team member who worked fulltime within the company throughout the duration of the project has established privileged communication channels between the university and the company and has directed management and staff attention to the project.

7.3 Case 2 – Improving the performance of an assembly line by sequentially solving type I and type II problems*

The goal of this study was to analyse and improve the assembly line's performance of an industrial manufacturer of plastic parts for household goods. Prior to this study, a simulation of the actual assembly line was conducted and the results showed a high unbalance of workloads between workstations. Some workstations had usage rates of 100%, which was causing long queues of sub-assemblies, while others had high values of idle time. It was clear that the assembly line needed to be re-balanced and, at this point, it was decided to try to apply some of the developed heuristics to this particular line. The main details of the project are described in the following sections.

* Part of the work of this section was presented in Simaria and Vilarinho (2003)

7.3.1 Characteristics of the assembly line

The assembly line produces five models (M1, M2, M3, M4 and M5) with some technical similarities. The first step of the study was to build the precedence diagrams for each model giving to common tasks the same identification, as shown in Figure 7.5.

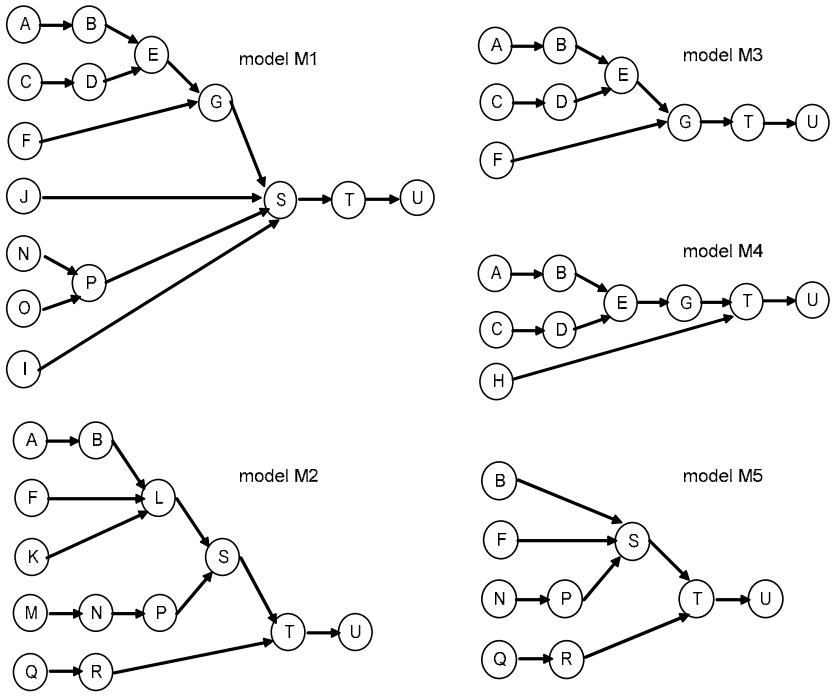

Figure 7.5 – Precedence diagrams of the five models

The actual line assembles the models in batches, so, in order to test the feasibility of having a mixed-model assembly line, able to produce the models in any intermixed sequence, the precedence diagrams of the models were combined into one diagram. The combined precedence diagram is depicted in Figure 7.6 and the task processing times, in time units, for each model are presented in Table 7.11.

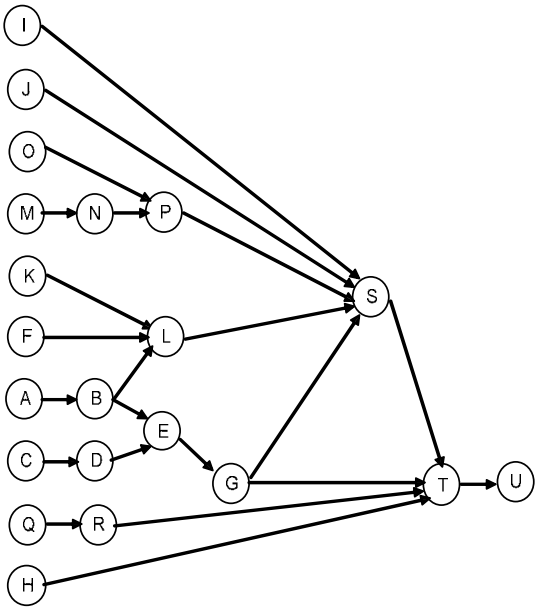

Figure 7.6 – Combined precedence diagram for the five models

Table 7.11 – Task processing times for the five models (t.u.)

	Model				
Task	**M1**	**M2**	**M3**	**M4**	**M5**
A	8.95	8.95	8.95	8.95	-
B	10.73	10.73	10.73	10.73	10.73
C	2.56	2.56	2.56	-	-
D	5.59	5.59	5.59	-	-
E	9.63	9.63	9.63	-	-
F	6.85	6.85	-	7.44	7.55
G	17.64	12.56	12.56	-	-
H	-	-	11.77	-	-
I	8.16	-	-	-	-
J	2.5	-	-	-	-
K	-	-	-	6.23	-
L	-	-	-	5.45	-
M	-	-	-	15.14	-
N	5.83	-	-	12.55	5.83
O	4.95	-	-	-	-
P	8.9	-	-	9.03	8.9
Q	-	-	-	12.17	11.58
R	-	-	-	5.68	5.68
S	25.74	-	-	12.77	12.77
T	17.1	17.1	17.1	17.1	17.1
U	5.02	5.02	5.02	5.02	5.02

From this set of tasks, task E and task G have negative zoning constraints, i.e., they cannot be performed in the same workstation. From an ergonomic point of view, it is not desirable that these two tasks are performed by the same operator, because each of them demands a substantial physical effort, so it would be too demanding for one person having to perform two hard tasks.

7.3.2 Two-step procedure for balancing the assembly line

Usually, when designing a new assembly line, companies do not know exactly how many product units the line should assemble in a given planning period. There is a demand forecast (more or less accurate) and a required cycle time (planning period/number of units) is estimated in order to balance the line. So, a line balance can be obtained using a technique to solve the assembly line balancing problem of type I (ALBP-I), in which the goal is to find a task assignment that minimises the number of operators for a given cycle time. However, this procedure does not guarantee a maximum production rate of the line. If a task reassignment is made, it might be possible to decrease the cycle time (thus increasing the production rate), for the same number of operators. This can be achieved by using a technique to solve the ALBP-II, in which the goal is to find a task assignment that minimises the cycle time for a given number of operators.

To improve the performance of the assembly line under study, a two-step procedure was developed. It sequentially solves ALBP-I and ALBP-II in order to (i) find a task assignment that minimises the number of operators for a given cycle time and (ii) maximises the production rate for that number of operators. In the first step, the cycle time of the actual assembly line is used as input to solve a balancing problem of type I. In the second step, the number of operators provided by the first step is used as input to solve a balancing problem of type II. In this particular study, simulated annealing and genetic algorithms were used in the first and second steps, respectively. The two steps are called the balancing step and the fine-tuning step, as shown in Figure 7.7.

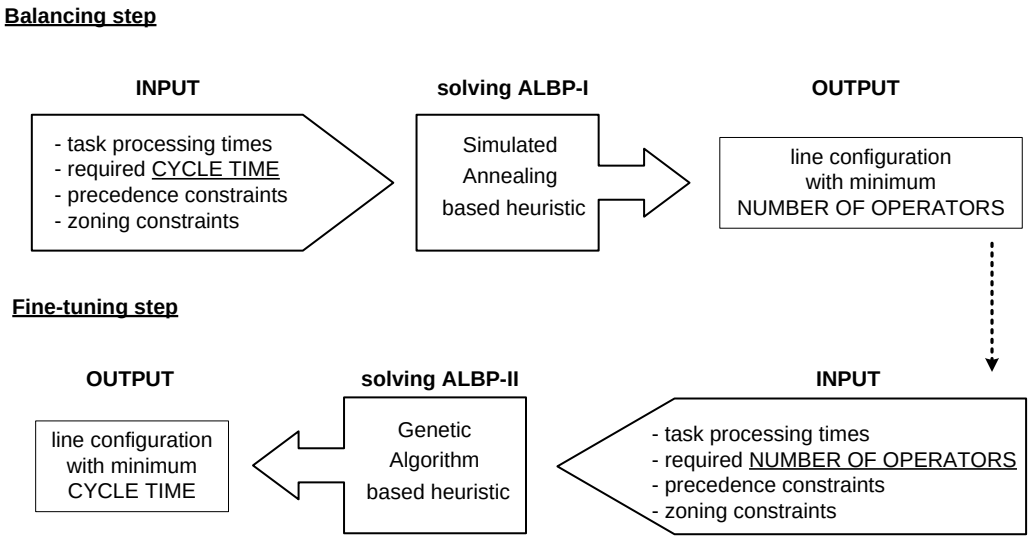

Figure 7.7 – Two-step procedure for balancing the assembly line

The combined precedence diagram was used to find a mixed-model line configuration, however the resulting solution was not attractive in terms of practical implementation. When processing some models, there were workstations with no assigned tasks. This is due to the fact that the similarities between the models are not very strong (e.g., the combined diagram has 21 tasks, while model C only requires 9 tasks). Therefore, it was decided to abandon the mixed-model idea and to move on to balance five single-model assembly lines, as this is also the way in which the actual line works.

The two-step procedure was applied to each of the models, and the results are presented in Table 7.12, which shows the number of operators and the cycle time of the actual assembly line and the results after each of the step of the procedure. After the balancing step, a reduction of one to three operators was verified and after the fine-tuning step cycle time was improved up to 19% (for model B).

Table 7.12 . Results of the two-step procedure

	Actual assembly line		Balancing step	Fine-tuning step
Model	**Operators**	**Cycle time (t.u.)**	**Operators**	**Cycle time (t.u.)**
M1	13	9.4	11	8.4
M2	13	9.8	11	7.9
M3	10	7.0	8	6.6
M4	10	7.3	9	7.1
M5	12	7.6	9	6.6

7.3.3 Implementation of the proposed solutions

The line configurations proposed by the two-step procedure were analysed by the line manager and a few task reassignments were made in order to allow a better flow of materials along the line. The new balancing solutions were then tested in the assembly line and the following aspects were observed:

- The number of units produced daily slightly increased in the first day and in the following days it increased significantly. This was due to the learning effect, as the operators needed time to adapt themselves to the new tasks and work flow.
- The distribution of workload between operators was much more levelled than what it used to be, leading to a more continuous flow of sub-assemblies and a reduction of the length of queues.
- The operators' motivation increased due to the smoother workload balance and to the awareness of the better performance of the line, for which they had a major responsibility.

7.3.4 Conclusions

The results of this project showed that, without any capital investment in more automatic equipment, it is possible to improve the performance of an assembly line only by studying the assignment of tasks to workstations.

On one hand, the simulation tool (not described in this document) was useful to detect the main problems of the line, namely, the unbalance of workloads among operators and the large number of sub-assemblies in queues. On the other hand, the two-step heuristic procedure was essential to re-balance the assembly line, optimising the utilisation of the company's resources.

The implementation of the proposed solutions showed a great improvement in the assembly line's performance, making the company able to cope with the increase of the products' demand.

7.4 Case 3 – Increasing flexibility by turning a straight line into a U-shaped line

The recent acquisition of the company in which the study took place by a large group led to the implementation of the group's business philosophy concerning management and production issues. One of these issues is the flexibility of production systems to cope with the uncertainty and variability of demand in the current market environment. Trying to achieve this flexibility, the philosophy supports the use of U-shaped assembly lines. Within this scope, the goal of this study was to analyse the performance of one of the assembly lines of the company (a major manufacturer of electronic security systems) and propose changes in its configuration in order to improve its flexibility.

7.4.1 Problems with the actual assembly line

The assembly line produces three models of a product in a straight line configuration. When the production volume is low, the assignment of operators to workstations (with specific equipment required to perform the tasks) increases the distances between workstations which harms the flow of the line. Also, it is difficult to have multi-skilled workers, able to perform tasks in several workstations when these are physically distant from each other. Another problem with the actual assembly line is the unbalance of workloads between workstations. While some operators have high workloads others have long idle times and because workstations are distant, they cannot help each other and smooth the workload. The original line was designed to assemble a different product that no longer exists, so the facilities were adapted in order to assemble other types of products. This somehow explains the poor performance of the line.

The demand of the product, and consequently the production volume of the line, is highly variable which forces the line to be frequently re-balanced. Whenever the line is re-balanced, the workstations have to be modified, as the equipment for the new tasks has to be installed, and operators have to be trained to perform the new set of tasks. This represents increased costs for the company that could be avoided if the line was easily adaptable to changes in production volumes.

7.4.2 Using U-ANTBAL to build a U-shaped layout

The approach to change the line configuration was to move to a U-shape, not only because it was a common practice among the other companies of the group but also because of its advantages, when compared with straight lines, considering the flexibility of re-balancing the line when production volume changes (see section 5.2).

The idea of applying the algorithm U-ANTBAL to this assembly line was challenging, as it was an opportunity to validate its assumptions and, at the same time, an opportunity to learn more about real industrial problems. The first step was to collect data about task processing times and to build the precedence diagrams of the models. Figure 7.8 shows the precedence diagram of one of the models. Then, the algorithm was run for three different production volume scenarios and U-line balancing solutions were obtained. An immediate conclusion of the analysis of these solutions was that they could not be the final U-line configurations. This was due to the following:

(i) Some workstations had to handle a high number of parts of the product – a difficult aspect to be implemented in the real line.

(ii) Some workstations required too many different pieces of equipment to perform its tasks.

(iii) The assignment of tasks to workstations for the different production volume scenarios was completely different, meaning that a radical change in the line (considering the equipment required for each workstation) had to be performed every time the production volume would change.

The first problem could be easily solved by adding negative zoning constraints in U-ANTBAL, forbidding the assignment of determined tasks to the same workstation. However the other two points are not included in the assumptions of the procedure. They are related with the physical equipment required in a workstation to perform its tasks. This could be implemented by assigning each task to a type of equipment and changing the algorithm accordingly. It would require a deep interaction with the trainee but unfortunately it was not possible.

Nevertheless, the balancing solutions provided by U-ANTBAL were a useful tool to understand the potential and limitations of a U-shaped line configuration.

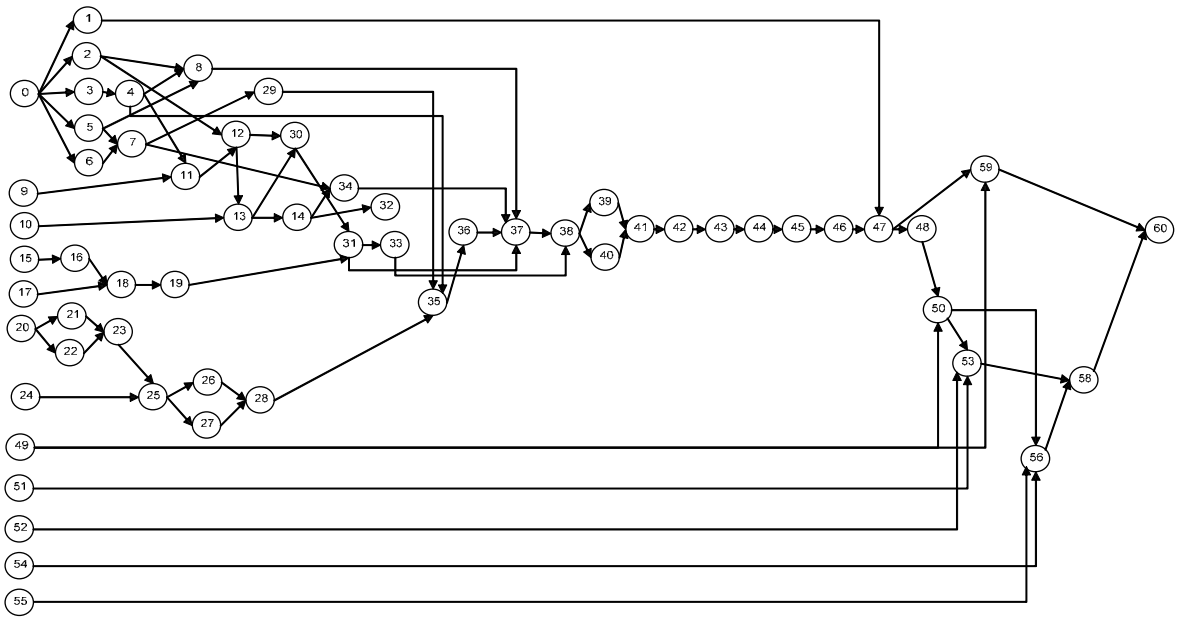

Figure 7.8 – Precedence diagram of one of the models

7.4.3 Adding flexibility to the line

The line had to be flexible enough so that whenever the production volume changed, the only change in the line would have to be the number of operators working on it – the physical workstations (and the correspondent equipment) had to remain the same. As it was not possible to improve U-ANTBAL to deal with this new set of constraints, the trainee did an empirical study and, by trial and error, was able to achieve line configurations with the desired flexibility. Figure 7.9 shows the assignments of operators to workstations in the U-shaped assembly line for three different production volumes. The workstations remain unchanged (concerning tools and equipment) and the number of operators, and the tasks they perform, vary for each scenario. This way, the required flexibility was attained.

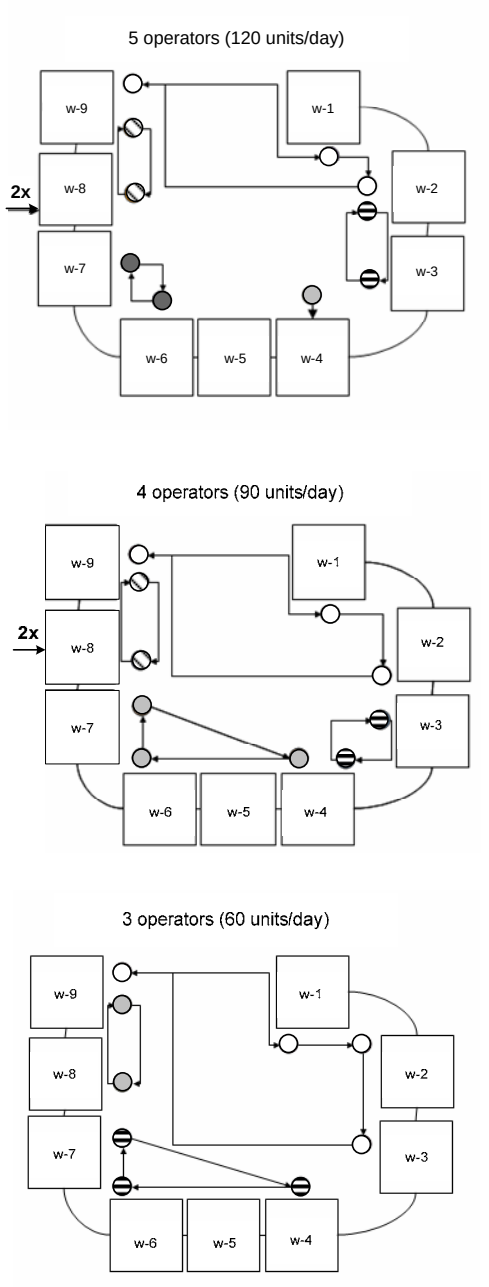

Figure 7.9 – Assignment of operators to workstations for different production volumes

Besides the flexibility to cope with the demand's variability, the U-shaped configuration allows an improvement of the performance of the assembly line. Table 7.13 provides a comparison between the number of operators and the percentage of idle time of the straight and U-shaped configurations for the three scenarios. These results prove that re-designing this particular assembly line will bring considerable gains to the company.

Table 7.13 – Comparison of performance measures between straight and U-shaped configurations

Scenario	Straight line	U-shaped line
120 units/day		
Number of operators	6	5
% idle time	15.8	8.0
90 units/day		
Number of operators	5	4
% idle time	24.2	14.0
60 units/day		
Number of operators	4	3
% idle time	36.8	13.0

7.4.4 Conclusions

This project showed that the use of U-lines is a common practice in large industrial groups, whose business philosophies are based in *just-in-time* principles, and that it is an effective way to address the uncertainty of demand volumes.

The results of the application of the algorithm U-ANTBAL provided more insight into the assembly process's characteristics and constraints. However, further developments of the procedure would be necessary in order to directly solve the problem on hand.

7.5 Case 4 – Balancing a 'n-sided' assembly line

The goal of this study was to analyse and improve the assembly line's performance of an industrial manufacturer of vehicle electrical wiring systems. The assembly process of this type of product has particular characteristics that make it very different from the assembly process in traditional assembly lines. However, with adequate modifications, it is possible to use algorithms for assembly line balancing to address this problem.

7.5.1 Characteristics of the assembly process

Electrical distribution systems are networks of wiring and associated control devices that route electrical power and signals throughout the vehicle. Wire harness assemblies consist of raw, coiled wire, which is automatically cut to length and terminated. Individual

circuits are assembled together on a jig or table, inserted into connectors and wrapped or taped to form wire harness assemblies, as shown in Figure 7.10.

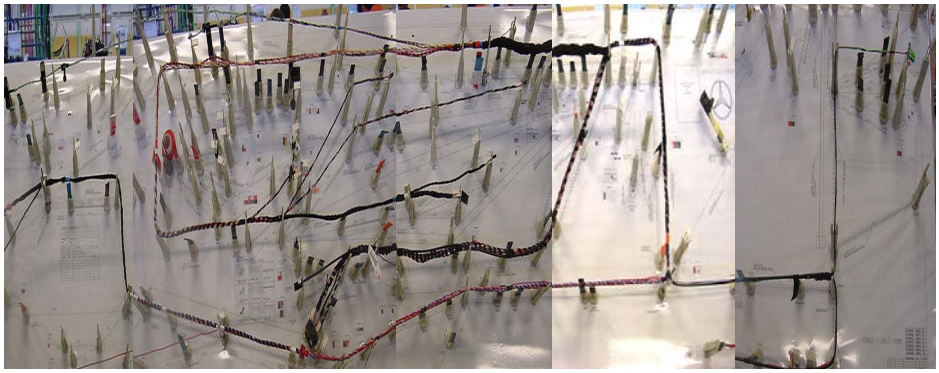

Figure 7.10 – A wire harness assembly jig

The jigs are linked together and move like a ‘carrousel’ along the workstations. Each operator in a workstation performs a set of tasks, either material preparation tasks or assembly tasks directly on the jig. Figure 7.11 illustrates such an assembly system. Several problems were identified when analysing the performance of the assembly line, and most of them were due to a misadjusted assignment of tasks to operators. In fact, it was often observed too many operators working simultaneously on the same jig (causing movement interferences among them) and some operators with too long idle times performing tasks that were not assigned to them in order to help more busy colleagues. It was clear the need of a deep study of the assembly line balancing.

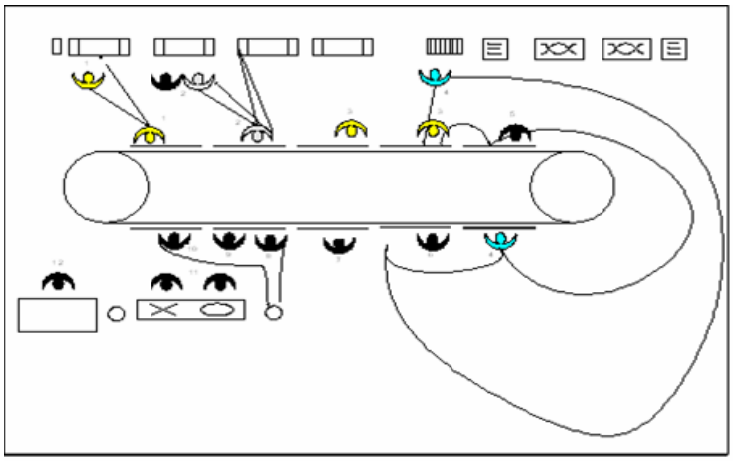

Figure 7.11 – Illustration of the wire harness assembly line

7.5.2 Adaptation of 2-ANTBAL to balance the assembly line

To balance the wire harness line it is necessary to define the set of tasks that each operator will perform in the assembly jig. The jigs are large enough to have more than one operator (three at maximum), however, in order to minimise the interference between them it is desirable that each operator work only in one area of the jig (left, right or centre). Tasks performed by each operator on the same jig can have precedence relationships, so, the sequence in which operators perform tasks must take them into account. Given these conditions, it was decided to adapt the algorithm 2-ANTBAL, especially developed for 2-sided assembly lines, to balance the wire harness assembly line. Instead of balancing a 2-sided line, the algorithm will have to balance the tasks of a 'n-sided' assembly process, meaning that different areas of the jig can be defined, according to the tasks involved in the assembly of a specific product.

The number of ants that simultaneously build the solution will depend on the number of sides defined for the problem: two if only the left and right sides of the jig are considered or three for left, right and centre. Obviously this can be generalised to any number of sides, according to the problem's characteristics.

The first step to use the balancing procedure was to build the precedence diagrams for the models being assembled and to specify task processing times, task sides and other assignment constraints. Figure 7.12 shows the precedence diagram of one of the models. Then, the algorithm was run and the resulting balancing solutions were analysed by the trainee. Like in the study presented in the previous section, the solutions provided by the algorithm were not adequate to be implemented in the line, due to the following:

(i) A large number of operators had to perform tasks on different branches of the wire – although it does not violate the assembly process constraints it makes the sequence of tasks more complicated to perform by the operators than if they only had to work on one branch.

(ii) The types of movement that the operators have to make to perform both their preparation of material and assembly tasks (from the jig to the preparation shelves and backwards) is not taken into account by the algorithm, which makes the assignment of some tasks to the same operator very difficult to implement.

The first problem can be solved by creating groups of tasks that are preferably, but not compulsory, performed by the same operator. This issue is addressed in the following section. The second problem could be addressed by adding to the algorithm more information and constraints about the movements involved in the performance of tasks and different levels of feasibility for the combination of movements. However, this would require a deep interaction with the element working in the company and, similarly to what happened in the previous study, this further interaction was not possible.

7.5.3 Addressing the assembly line planner's preferences

To address the assembly line planner's preferences for grouping tasks in the same workstation, although it is not compulsory according to the assembly process, like it was verified in the particular assembly line, the algorithm was modified to solve the problem. It was included the possibility of defining groups of tasks that will be preferably, but not compulsory, grouped in the same workstation. R_g is the set of pairs of tasks that the assembly planner prefers to group in the same workstation and $\#R_g$ is the number of pairs of tasks belonging to R_g. To address this issue, a new term of the objective function, used to guide the search of 2-ANTBAL, is included. It computes the distance (measured in number of workstations) between the tasks in the assembly planner preferences groups, as follows:

$$D = 1 - \frac{\sum_{g} \sum_{(i,j) \in R_g} y_{ij}}{\sum_{g} \# R_g} \quad \textbf{(7.1)}$$

where y_{ij} equals 1 if task i is assigned to the same workstation of task j and zero otherwise. When all the tasks belonging to the same group, g, are assigned to the same workstation (most favourable case), function D takes the value of zero. If none of the related tasks is assigned to the same workstation D takes the value of 1. The new objective function of 2-ANTBAL is then:

$$\lambda WE^{2s} - B_b^{2s} - B_w^{2s} - D \quad \textbf{(7.2)}$$

The lack of deeper interaction with the element in the company (to accurately define the groups of tasks) made it impossible to implement this new approach in the wire harness

assembly line. However, it seems a promising way to tackle this issue of relevant practical importance.

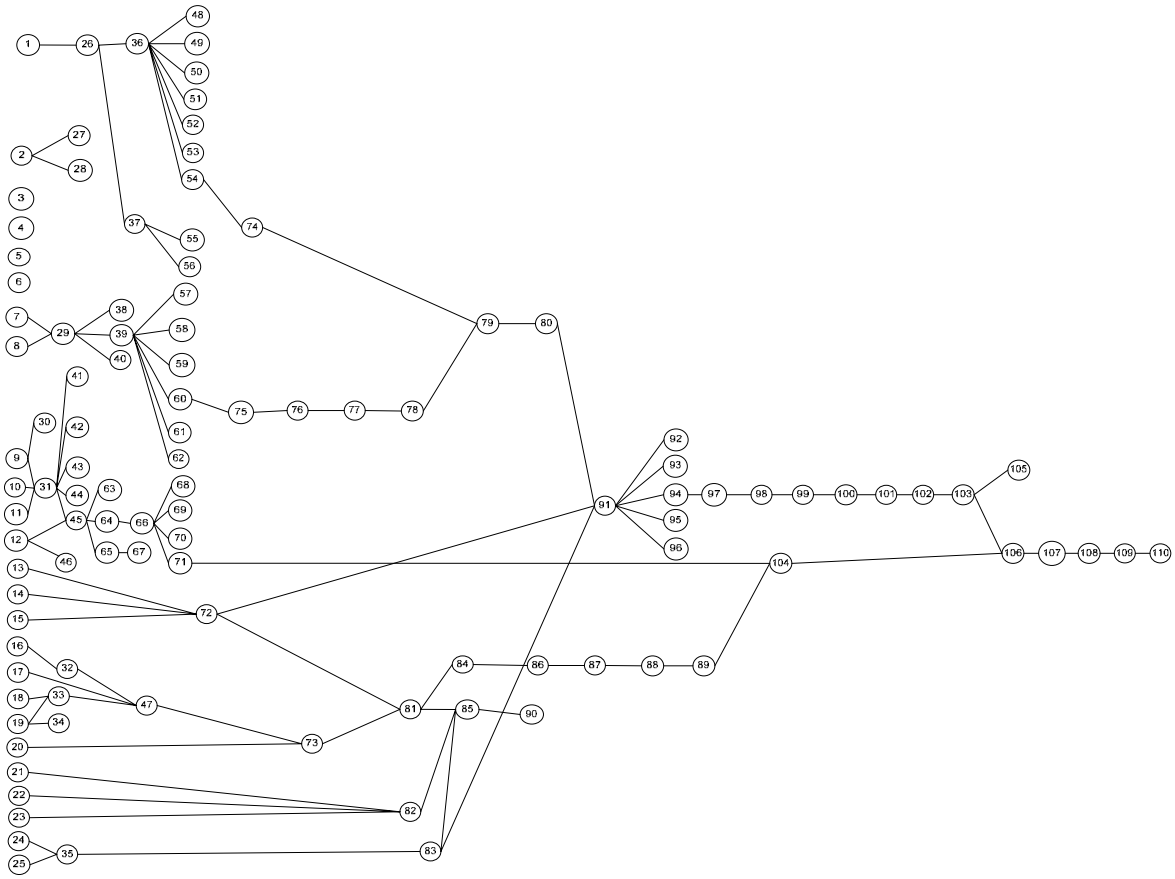

Figure 7.12 – Precedence diagram for one model

7.5.4 Conclusions

The study of line's supply and operator's movement features to improve the assignment of tasks to operators in the 'carrousel' was done empirically by the trainee, as there was no opportunity to include these aspects on the algorithm. Still, some balancing ideas were taken from the solutions previously provided by 2-ANTBAL. Significant improvements of the line's performance were achieved, namely, the increase of production volume, the smoothing of workloads between operators and the improvement of the operators' working conditions.

7.6 Chapter conclusions

The results of these four projects showed that the developed procedures for balancing assembly lines, described throughout this document, are suitable to address real world industrial problems. However, a very strong exchange of information is needed between researchers and practitioners in order to adequately adapt the algorithms.

The most surprising aspect of these experiences was related with the precedence diagrams. For a researcher the precedence diagrams are just input data as it is the cycle time or the task processing times. However, it was verified in all the four cases that companies do not have precedence diagrams of their assembly processes. In fact companies do not know this tool at all. What they have is an idea of the sequence in which the tasks are performed in the assembly line and they are reluctant about changing this sequence.

So, the biggest challenge was to help the trainees in building the precedence diagrams. It was a team work exercise because, on one hand, it required accurate information about technological constraints and, on the other hand, it required a constant questioning of the actual task sequence, which could only be made by someone who did not know the process. The resulting precedence diagrams were the most useful tool provided to the company, as they presented a wide range of alternatives of assembling the products. Just because of this, it was worthwhile to carry out these projects.

The outcomes of these studies prove that a more deep interaction between the scientific and industrial community is needed in order to improve the quality of both research work and production systems' performance.

8

Conclusion

Contents

- **Final remarks**
- **Future developments**

8.1 Final remarks

Recent market trends show that there is a growing demand for customised products, increasing the pressure for manufacturing flexibility. Mixed-model assembly lines, in which a set of similar models of a product can be assembled simultaneously, are an adequate production system to respond to the shifting of manufacturing assembly operations from high-volume/low-mix to high-mix/low-volume production systems. Assembly lines have the ideal structure to perform final product customisation tasks under a mass customisation concept. Also, as they are labour intensive, assembly lines can be easily located geographically closer to the final customer marketplace. The efficient design and operation of mixed-model assembly lines is, therefore, a crucial factor for the success of the implementation of the new manufacturing paradigms, namely postponement strategies.

The aim of this thesis was to address the mixed-model assembly line balancing problem by providing a set of procedures to efficiently tackle it for different types of assembly lines, thus contributing to the research in a field that involves a key factor of competitiveness in the actual market environment: assembly operations.

For balancing mixed-model assembly lines with a straight line configuration, three procedures, based on the meta-heuristics *simulated annealing*, *genetic algorithms* and *ant colony optimisation* (ACO) were developed and 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). Also, the proposed approach is different from the ones reported in the literature taking into account the fact that it provides good balancing solutions regardless of the sequence in which the models are launched into the line, making the mixed-model assembly line sequencing problem irrelevant.

The ACO based approach, called ANTBAL, was selected to be applied to other assembly line types, due to the results of the computational experience carried out in the straight line scenario, as well as to enlarge the range of ACO applications. This way, the mixed-model U-shaped and the mixed-model 2-sided line balancing problems were

addressed using adaptations of the original version of ANTBAL. Computational experiments proved a good performance of these procedures.

For all of the addressed problems, mathematical programming models were built in order to formally describe the problems as well as to help the description of the underlying principles of the proposed approaches. The goal was not to solve the models, as its high complexity made its resolution impossible, even for medium sized problems.

Some of the procedures were applied to real assembly lines in order to test their flexibility to cope with real industrial settings, as they may differ significantly from theoretical problems. The results showed that the developed procedures are suitable to address real world industrial problems. However, a very strong exchange of information is needed between researchers and practitioners in order to adequately adapt the algorithms.

The major of contribution of the work presented in this thesis derives from the following:

(i) The proposed procedures are able to address some particular features of the assembly process very common in real world assembly lines that most of the techniques existing in the literature do not consider. The aim was to obtain good solutions for complex problems instead of trying to find optimal solutions for simpler versions of the problem, the most frequent approach found in the literature.

(ii) The proposed models include novel criteria to access the quality of the solutions generated, namely workload smoothing within and among workstations.

(iii) The approach developed to handle the mixed-model nature of the problem is unique. All existing approaches use the average task processing times and solve the mixed-model problem like a single-model one. The proposed method uses the particular task times for each model and the balancing solution is built ensuring that the capacity constraints are verified for every model being assembled. This way, the feasibility and efficiency of the line configuration is valid for every model sequence.

(iv) The use of meta-heuristics follows the recent developments in combinatorial optimisation problem solving of using procedures inspired by nature to address complex problems. More particularly, the application of ACO algorithms to

production problems is still a research area with a lot to explore and the results of this study confirm its potential.

(v) The projects developed in industrial companies showed that the proposed procedures can be adequately adapted to deal with the conditions of real production systems.

8.2 Future developments

The developed algorithms will be included in a module of a decision support system (DSS) to design manufacturing systems geared to the make-to-order production stage, essential to implement postponement strategies. The research project in which the DSS will be developed has already started and it will last three years.

The development of new procedures to address different types of problems is also within the future research perspectives. Among these problems are:

(i) Designing flexible line configurations to efficiently handle uncertainty in product demand (similarly to what was verified in the project 'Case 3').

(ii) Balancing hybrid lines, i.e., lines in which some tasks are performed by manual operators and other tasks are automatically performed by machines.

(iii) Balancing multiple assembly line facilities, i.e., facilities that have more than one line in which sub-assemblies are manufactured that feed into a central line where the final product his assembled.

(iv) Providing support to the design of supply chains that operate under a postponement strategy, namely by developing models to define the customer order decoupling point for the relevant product range.

Also, further developments should be made in matching theoretical procedures and practical applications, shortening the gap between scientific research and industrial needs.

References

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

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

Ajenblit, D.A. and Wainwright, R.L. (1998) Applying genetic algorithms to the U-shaped assembly line balancing problem. *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*.

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

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

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

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

Bautista, J. and Pereira, J. (2002) Ant algorithms for assembly line balancing. *Proceedings of the Third International Workshop, ANTS 2002, Brussels, Belgium*, 65-76.

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

Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999) *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford University Press.

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

Bukchin, Y. and 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*, in press.

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

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

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

Cicirello, V.A. and Smith, S.F. (2001) Insect Societies and Manufacturing. *IJCAI-01 Workshop on Artificial Intelligence and Manufacturing, Working Notes*, 33-38.

Coronado, A.E., Lyons, A.C., Kehoe, D.F. and Coleman, J. (2004) Enabling mass customization: extending build-to-order concepts to supply chains. *Production Planning and Contro*l, 15, 398-411.

CPLEX Optimization, Inc (1999). MPL Modelling System.

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

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

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

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

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

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

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

Eglese, R.W. (1990) Simulated annealing: a tool for operational research. *European Journal of Operational Research*, 46, 271-281.

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

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

Falkenauer, E. (1998) *Genetic Algorithms and Grouping Problems*. Chichester: John Wiley & Sons.

Feyzbakhsh, S.A. and Matsui, M. (1999) Adam-Eve genetic algorithm: a methodology for optimal design of a simple flexible assembly system. *Computers and Industrial Engineering*, 36, 233-258.

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

Glover, F. (1989) Tabu search – part I. *ORSA Journal on Computing*, 1, 190-206.

Glover, F. (1990) Tabu search – part II, *ORSA Journal on Computing*, 2, 4-32.

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

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

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

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

Gonçalves, J.F. and Almeida, J.R. (2002) A hybrid genetic algorithm for assembly line balancing. *Journal of Heuristics*, 8, 629-642.

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

Hackman, S.T., Magazine, M.J. and Wee, T. S. (1989) Fast, effective algorithms for simple assembly line balancing problems. *Operations Research*, 37, 916-924.

He, D. and Babayan, A. (2002) Scheduling manufacturing systems for delayed product differentiation in agile manufacturing. *International Journal of Production Research*, 40, 2461-2481.

Heinrici, A. (1994) A comparison between simulated annealing and tabu Search with an example from the production planning In: Dyckoff, H, U. Denigs, M. Salomon, H.C. Tijns(eds): *Operations Research Proceedings* (pp 498-503). Berlin: Springer.

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

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

Ji, P. Sze, M.T. and Lee, W.B. (2001) A genetic algorithm of determining cycle time for printed circuit board assembly lines. *European Journal of Operational Research*, 128, 175-184.

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

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

Kim, Yeo Keun, Kim, Yong Ju and Kim, Yeongho (1996) Genetic algorithms for assembly line balancing with various objectives. *Computers and Industrial Engineering*, 30, 397-409.

Kim, Yong Ju, Kim, Yeo Keun and Cho, Yongkyun (1998) A heuristic-based genetic algorithm for workload smoothing in assembly lines. *Computers and Operations Research*, 25, 99-111.

Kim, Yeo Keun, Kim, Yeongho and Kim, Yong Ju (2000a) Two-sided assembly line balancing: a genetic algorithm approach. *Production Planning and Control*, 11, 44-53.

Kim, Yeo Keun, Kim, Sun Jin, Kim, JaeYun (2000b) Balancing and sequencing mixed-model U-lines with a co-evolutionary algorithm. *Production Planning and Control*, 11, 754-764.

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

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

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

Lapierre, S.D. and Ruiz, A.B. (2004) Balancing assembly lines: an industrial case study. *Journal of the Operational Research Society*, 55, 589-597.

Lee, S.G., Khoo, L.P. and Yin, X.F. (2000) Optimising an assembly line through simulation augmented by genetic algorithms. *International Journal of Advanced Manufacturing Technology*, 16, 220-228.

Lee, T.O., Kim, Y. and Kim, Y.K. (2001) Two-sided assembly line balancing to maximize work relatedness and slackness. *Computers and Industrial Engineering*, 40, 273-292.

Leu, Y., Matheson, L.A. and Rees, L.P. (1994) Assembly line balancing using genetic algorithms with heuristic-generated initial populations and multiple evaluation criteria. *Decision Sciences*, 25, 581-606.

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

Liu, S.B., Ong, H.L. and Huang, H.C. (2003) Two bi-directional heuristics for the assembly line type II problem. *International Journal of Advanced Manufacturing Technology*, 22, 656-661.

Malakooti, B. (1991) A multiple criteria decision making approach for the assembly line balancing problem. *International Journal of Production Research*, 29, 1979-2001.

Malakooti, B. (1994) Assembly line balancing with buffers by multiple criteria optimization. *International Journal of Production Research*, 32, 2159-2178.

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

McMullen, P.R. and 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. and Frazier, G.V. (1998) Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations. *Computers and Industrial Engineering*, 36, 2717-2741.

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

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

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

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

Mikkola, J.H. and Skjott-Larsen, T. (2004) Supply-chain integration: implications for mass customization, modularization and postponement strategies. *Production Planning and Control*, 15, 352-361.

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

Miltenburg, J. (2000) The effect of breakdowns on U-shaped production lines. *International Journal of Production Research*, 38, 353-364.

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

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

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

Monden, Y. (1993) *Toyota Production System*. Norcross: Institute of Industrial Engineers.

Nair, A. (2005) Linking manufacturing postponement, centralized distribution and value chain flexibility with performance. *International Journal of Production Research*, 43, 447-463.

Nakade, K. and Ohno, K. (1999) An optimal worker allocation problem for a U-shaped production line. *International Journal of Production Economics*, 60, 353-358.

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

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

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

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

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

Rachamadugu, R. and Talbot, B. (1991) Improving the equality of workload assignments in assembly lines. *International Journal of Production Research*, 29, 619-633.

Ramos, A.L., Mendes, A.R., Simaria, A.S. and Vilarinho, P.M. (2001) Using simulation models for balancing a PC camera assembly line. *Proceedings of the 6th Annual International Conference on Industrial Engineering – Theory, Applications and Practice, San Francisco, CA, U.S.A, November 2001*.

Rekiek, B., De Lit, P. and Delchambre, A (2000) Designing mixed-product assembly lines. *IEEE Transactions on Robotics and Automation*, 16, 268-280.

Rekiek, B., De Lit, P., Pellichero, F., l'Eglise, T., Fouda, P., Falkenauer, E. and Delchambre, A. (2001) A multiple objective grouping genetic algorithm for assembly line design. *International Journal of Intelligent Manufacturing*, 12, 467-485.

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

Rudberg, M. and Wikner, J. (2004) Mass customization in terms of the customer order decoupling point. *Production Planning and Control*, 15, 445-458.

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

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

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

Schofield, N.A. (1979) Assembly line balancing and the application of computer techniques. *Computers and Industrial Engineering*, 3, 53-69.

Scholl, A. (1993) Data of assembly line balancing problems. <http://www.wiwi.uni-jena.de/Entscheidung/>

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

Scholl, A. and Klein, R. (1999) ULINO: Optimally balancing U-shaped JIT assembly lines. *International Journal of Production Research*, 37, 721-736.

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

Scholl, A. and Voβ, S. (1996) Simple assembly line balancing - heuristic approaches. *Journal of Heuristics*, 2, 217-244.

Shtub, A. and Dar-El, E.M. (1990) An assembly chart oriented assembly line balancing approach. *International Journal of Production Research*, 28, 1137-1151.

Simaria, A.S. (2001) *Uma Metodologia para o Balanceamento de Linhas de Montagem*. Dissertação de Mestrado. Escola de Gestão do Porto, Universidade do Porto.

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

Simaria, A.S. and Vilarinho, P.M. (2003) Fine-tuning assembly line balancing – a decision support framework. *Proceedings of the 8th Annual International Conference on Industrial Engineering – Theory, Applications and Practice, Las Vegas, Nevada, U.S.A, November 2003*.

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

Skipworth, H. and Harrison, A. (2004) Implications of form postponement to manufacturing: a case study. *International Journal of Production Research*, 42, 2063-2081.

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

Stützle, T. and Dorigo, M. (1999) ACO algorithms for the traveling salesman problem. *Tech.Rep. IRIDIA/99-3 Université Libre de Bruxelles, Belgium*.

Su, J.C.P., Chang, Y.-L. and Ferguson, M. (2005) Evaluation of postponement structures to accommodate mass customization. Journal of Operations Management, 23, 305-318.

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

Tsujimura, Y, Gen, M. and Kubota, E. (1995) Solving fuzzy assembly-line balancing problem with genetic algorithms. *Computers and Industrial Engineering*, 29, 543-547.

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

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

Vilarinho, P.M. and 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.

Yang, B. and Burns, N. (2003) Implications of postponement for the supply chain. *International Journal of Production Research*, 41, 2075-2090.

Yang, B., Burns, N.D. and Backhouse, C.J. (2004) Management of uncertainty through postponement. *International Journal of Production Research*, 42, 1049-1064.

Yang, B., Burns, N.D. and Backhouse, C.J. (2005) An empirical investigation into barriers to postponement. *International Journal of Production Research*, 43, 991-1005.

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

Zhao, Z.Y. and de Souza, R. (2000) Genetic production line-balancing for the hard disk drive industry. *The International Journal of Advanced Manufacturing Technology*, 16, 297-302.

Appendix 1

Demonstration of the maximum and minimum values of functions B_b and B_w

A1.1 Function B_b (balance between workstations)

$$Minimise \quad B_b = \frac{LL}{LL-1}\sum_{k=1}^{LL}\left(\frac{\sum_{m=1}^{M} q_m s_{km}}{WIT} - \frac{1}{LL}\right)^2$$

- **Minimum value** – B_b reaches the minimum value of zero when *WIT* is evenly distributed between workstations (best case).
  - for all (*LL*) workstations: $\sum_{m=1}^{M} q_m s_{km} = \frac{WIT}{LL}$

$$B_b = \frac{LL}{LL-1} LL\left(\frac{\frac{WIT}{LL}}{WIT} - \frac{1}{LL}\right)^2 = \frac{LL}{LL-1} LL\left(\frac{1}{LL} - \frac{1}{LL}\right)^2 = 0$$

- **Maximum value** – B_b reaches the maximum value of 1 when *WIT* is only accountable to one workstation (worst case).
  - for one workstation: $\sum_{m=1}^{M} q_m s_{km} = WIT$
  - for *LL*-1 workstations: $\sum_{m=1}^{M} q_m s_{km} = 0$

$$B_b = \frac{LL}{LL-1}\left(\left(\frac{WIT}{WIT} - \frac{1}{LL}\right)^2 + (LL-1)\left(\frac{0}{WIT} - \frac{1}{LL}\right)^2\right) =$$

$$= \frac{LL}{LL-1}\left(\left(1 - \frac{1}{LL}\right)^2 + (LL-1)\left(\frac{1}{LL}\right)^2\right) = \frac{LL}{LL-1} \cdot \frac{LL-1}{LL^2} \cdot \left((LL-1)+1\right) = 1$$

A1.2 Function B_w (balance within workstations)

$$Minimise \quad B_w = \frac{M}{LL(M-1)}\sum_{k=1}^{LL}\sum_{m=1}^{M}\left(\frac{q_m s_{km}}{S_k} - \frac{1}{M}\right)^2$$

- **Minimum value** – B_w reaches the minimum value of zero when, for all workstations, S_k is evenly distributed among the models (best case).

- for all (LL) workstations and all (M) models: $q_m s_{km} = \frac{S_k}{M}$

$$B_w = \frac{M}{LL(M-1)} LL \cdot M \cdot \left(\frac{\frac{S_k}{M}}{S_k} - \frac{1}{M} \right)^2 = \frac{M}{LL(M-1)} LL \cdot M \cdot \left(\frac{1}{M} - \frac{1}{M} \right)^2 = 0$$

- **Maximum value** – B_w reaches the maximum value of 1 when, for all workstations, S_k is only accountable to one model (worst case).
  - for one model: $q_m s_{km} = S_k$
  - for M-1 models: $q_m s_{km} = 0$

$$B_w = \frac{M}{LL(M-1)} LL \left(\left(\frac{S_k}{S_k} - \frac{1}{M} \right)^2 + (M-1) \left(\frac{0}{S_k} - \frac{1}{M} \right)^2 \right) =$$

$$= \frac{M}{LL(M-1)} LL \left(\left(1 - \frac{1}{M} \right)^2 + (M-1) \left(\frac{1}{M} \right)^2 \right) = \frac{M}{LL(M-1)} LL \cdot \frac{M-1}{M^2} \cdot \left((M-1) + 1 \right) = 1$$

Appendix 2

Characteristics of the MALBP data sets

A2.1 Precedence diagrams of the problems of the MALBP data sets

Problem	Number of tasks	Reference of the precedence diagram
1, 2	8	Bowman in Scholl (1993)
3, 4	11	Gokçen and Erel (1998)
5, 6	21	Mitchell in Scholl (1993)
7, 8	25	Numerical example of Figure XX
9, 10	28	Heskiaoff in Scholl (1993)
11, 12	30	Sawyer in Scholl (1993)
13, 14	32	Lutz1 in Scholl (1993)
15, 16	35	Gunther in Scholl (1993)
17, 18	45	Kilbridge in Scholl (1993)
19, 20	70	Tonge in Scholl (1993)

A2.2 Task processing times (in time units)

A2.2.1 MALBP data set with typical task times

	Problem 1		Problem 2			Problem 3		Problem 4			Problem 5		Problem 6			Problem 7	
task	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B
1	3.8	3.8	4.4	4.4	0	3.0	0	8.3	8.6	8.3	12.8	12.8	11.8	11.8	11.8	0	2
2	1.9	1.9	0	6.0	0	3.1	3.1	0	2.0	2.0	7.4	8.5	12.1	12.1	12.1	7.7	7.7
3	1.8	1.8	0	7.0	7.0	1.9	1.9	9.6	9.6	9.6	11.7	11.7	14.0	14.0	14.0	7.3	7.3
4	2.1	2.1	10	11.3	10	8.4	8.4	1.8	1.8	1.8	3.8	3.8	5.4	5.0	5.5	15	15
5	7.8	7.9	0	6.0	6.0	3.1	3.1	2.4	2.4	2.5	4.8	4.8	6.5	7.2	6.5	8.8	8.8
6	4.5	4.5	12.3	12.3	12.3	11.2	9.9	2.3	2.3	2.3	5.8	5.8	3.2	0	3.2	6.2	0
7	12.0	9.1	7.8	0	7.8	8.8	0	2.3	2.3	2.5	4.9	4.9	5.0	5.8	5.2	3.6	0
8	1.9	2.0	0	0	10	8.7	8.7	4.7	4.7	4.7	4.7	0	6.3	6.7	7.2	0	2.0
9						2.5	2.5	0	9.0	9.0	4.5	4.5	5.0	5.0	5.0	6.6	6.6
10						5.2	0	13.6	13.6	13.6	9.4	8.6	0	0	10	2.5	2.5
11						4.4	4.4	1.0	1.0	1.0	3.5	3.5	9.0	10.3	0	5.5	5.5
12											4.2	4.2	0	1.0	1.0	7.1	7.1
13											9.6	9.6	0	5.0	5.0	5.9	5.9
14											2.1	0	2.7	2.7	2.4	1.3	0
15											0	0	15.0	15.0	15.0	5.5	5.5
16											13.7	0	2.4	0	2.4	1.9	2.0
17											8.5	8.5	2.1	2.1	1.8	3.7	0
18											6.6	6.6	2.8	2.8	2.8	9.4	9.4
19											2.1	2.1	7.3	7.3	7.3	1.3	1.3
20											6.1	6.1	5.6	4.8	5.6	0	9.0
21											3.9	0	0	1.0	1.0	2.0	2.0
22																4.7	4.7
23																9.6	8.2
24																4.1	3.7
25																12.5	0

	Problem 8			Problem 9		Problem 10			Problem 11		Problem 12			Problem 13		Problem 14		
task	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C
1	4.1	4.1	4.1	1.0	1.0	3.4	3.4	0	9.5	9.5	2.1	2.1	2.1	2.4	2.4	0	4.0	0
2	2.7	2.7	2.7	3.7	3.7	10.0	10.0	10.0	1.3	1.3	3.2	3.2	3.6	5.3	5.3	6.6	5.7	6.6
3	4.6	4.6	4.6	7.2	7.2	8.0	8.0	8.0	4.8	4.8	4.3	0	4.3	2.0	2.0	2.0	2.0	2.0
4	4.1	4.1	4.1	12.6	12.6	12.7	11.8	12.7	3.3	3.3	0	5.0	5.0	4.0	4.3	4.3	4.0	4.3
5	2.0	2.0	2.0	4.2	4.2	9.3	9.3	9.3	1.5	1.7	3.7	3.7	3.7	4.0	4.0	4.0	4.0	4.0
6	0	2.0	2.0	3.4	3.4	3.3	3.3	3.3	4.5	4.1	13.4	13.4	13.4	4.4	4.4	4.4	4.4	4.4
7	11.3	11.3	11.3	3.4	3.4	4.7	4.7	4.7	3.6	3.6	2.1	0	2.1	1.3	1.5	1.3	1.3	1.5
8	7.8	7.8	7.8	10.7	10.7	8.6	8.6	8.6	0	2.0	7.8	0	7.8	1.4	1.4	1.4	1.4	1.4
9	0	10.0	10.0	9.9	0	8.2	0	8.2	12.3	12.3	4.2	4.2	4.2	4.9	4.5	4.9	4.9	4.5
10	3.5	3.5	3.3	7.6	7.6	8.4	8.4	8.4	0	8.0	9.9	9.9	9.9	6.6	6.6	4.5	4.4	4.5
11	3.9	4.2	3.9	9.8	9.8	0	0	9.0	2.5	2.5	5.8	5.8	6.0	6.9	0	3.8	3.8	3.8
12	1.0	1.0	1.0	2.4	2.4	8.9	8.3	8.9	4.3	4.3	3.0	2.9	3.0	2.4	2.4	2.4	2.4	2.4
13	2.5	2.3	2.5	9.8	8.7	4.6	4.6	0	6.5	0	0	2.0	2.0	5.3	5.3	5.3	5.3	5.3
14	5.1	5.1	5.1	10	10	0	0	5.0	1.7	1.7	2.4	2.4	2.4	12.2	12.2	12.2	12.2	12.2
15	3.5	3.5	3.5	0	9.0	9.2	9.3	9.2	7.0	7.0	7.0	6.5	7.0	2.7	2.7	2.7	2.7	2.7
16	3.5	3.5	3.4	0	9.0	1.9	2.1	2.0	1.4	1.4	3.8	0	3.8	5.5	5.5	5.5	5.5	5.5
17	6.8	6.8	6.8	2.7	2.7	4.8	4.8	4.8	7.8	7.8	0	2.0	2.0	6.6	6.6	6.6	6.6	6.6
18	8.5	8.5	9.6	3.6	3.7	10.2	11.5	10.2	2.9	2.9	5.7	5.7	5.7	6.8	6.5	6.8	6.8	6.5
19	9.9	9.9	9.9	7.7	7.7	4.4	4.6	4.4	1.6	1.6	1.4	1.4	1.4	9.5	9.5	9.5	9.5	9.5
20	7.2	7.2	7.2	12.4	12.4	0	3.0	0	7.0	7.0	13.7	13.7	13.7	0	14.9	14.9	0	14.9
21	4.8	4.8	4.8	6.2	6.2	1.4	1.4	1.4	8.7	8.7	6.4	6.4	6.9	2.2	2.2	2.8	2.8	2.8
22	3.8	3.8	3.9	6.0	6.6	4.1	4.1	0	3.9	4.1	5.0	5.0	5.3	4.8	4.8	6.4	6.4	6.4
23	2.9	2.8	2.6	5.5	5.5	4.1	4.1	4.1	6.4	6.4	12.8	12.8	12.8	5.1	5.8	8.6	8.6	8.6
24	3.5	3.5	3.5	1.9	1.9	0	9.0	9.0	2.8	2.7	0	2.0	2.0	0	10	9.7	8.2	9.7
25	7.8	7.8	7.8	4.4	4.4	9.5	9.5	10	8.5	8.5	8.2	8.2	0	1.1	1.0	5.8	5.5	5.8
26				12.1	12.1	9.2	9.2	9.2	6.7	6.7	9.1	9.1	9.1	2.4	2.4	2.4	2.4	2.4
27				9.2	9.2	9.9	9.9	9.9	1.9	1.9	9.7	9.7	9.7	1.7	1.7	0	2.0	2.0
28				9.5	9.9	3.7	3.7	3.1	9.9	9.9	7.2	7.2	7.2	12.3	13.5	3.9	3.9	4.5
29									4.6	0	10.5	10.5	10.5	2.5	2.5	7.1	7.1	7.1
30									4.0	4.2	2.4	2.4	2.4	0	0	2.0	0	0
31														5.1	5.1	5.1	5.1	5.1
32														4.1	4.0	4.0	4.1	4.0

	Problem 15		Problem 16			Problem 17		Problem 18			Problem 19		Problem 20		
task	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C
1	4.9	4.9	7.1	7.1	7.1	1.0	1.0	7.5	8.0	7.5	5.8	5.8	3.2	3.5	3.2
2	7.2	7.2	3.2	3.2	3.2	4.4	5.1	6.0	0	6.0	5.7	5.7	11.7	11.7	11.7
3	3.3	3.3	4.1	0	4.1	14.3	0	5.6	5.6	5.6	0	10	6.0	5.6	6.0
4	13.6	13.6	8.0	8.0	8.0	2.2	2.2	3.8	0	3.8	5.0	5.1	1.9	0	1.9
5	4.4	4.4	5.3	5.3	6.0	4.8	4.8	3.1	2.9	3.1	5.1	5.1	0	15.0	0
6	6.5	6.5	11.8	11.8	11.9	5.1	5.8	4.2	4.2	4.2	0	8.0	7.4	7.4	7.4
7	4.1	4.1	11.7	11.7	11.7	0	10.0	10.6	10.6	10.6	6.8	6.8	7.9	8.4	7.9
8	4.2	4.2	3.4	3.4	3.4	5.1	5.1	0	8.0	8.0	4.5	4.5	5.2	5.4	5.2
9	4.1	4.1	8.7	8.7	8.7	9.4	9.4	3.6	3.6	3.6	5.1	5.1	2.3	2.3	2.3
10	11.7	12.2	2.9	3.0	2.9	5.0	5.0	2.4	2.4	2.4	9.6	9.6	5.8	4.9	5.8
11	2.2	2.2	4.4	4.4	0	3.5	3.5	9.3	9.3	0	1.5	1.5	2.6	2.6	0
12	0	3.0	5.6	5.6	5.6	0	4.0	1.1	1.2	1.1	3.9	3.9	4.3	4.6	4.3
13	0	1.0	4.2	4.3	4.2	7.0	0	2.7	0	0	9.6	9.6	2.2	2.2	2.2
14	8.0	6.9	13.8	13.8	13.8	2.7	0	5.6	5.6	5.0	2.7	2.7	3.5	3.5	3.7
15	6.6	7.4	2.5	2.5	2.5	5.3	5.3	9.3	9.3	9.3	1.5	1.5	8.7	0	8.7
16	4.2	4.2	5.9	6.1	5.9	0	3.0	0	8.0	7.2	4.2	0	8.4	8.4	8.4
17	7.6	7.6	7.7	7.7	7.7	2.2	2.2	10.4	10.4	10.4	5.3	5.3	2.1	2.1	2.1
18	6.5	6.5	0	4.0	4.0	0	3.0	9.0	9.0	9.0	3.7	3.7	8.4	8.4	8.4
19	5.3	5.3	1.0	1.0	1.0	8.3	8.3	12.0	12.0	12.1	8.1	8.1	8.1	8.1	8.1
20	7.4	7.4	7.4	7.4	7.4	2.6	2.6	6.6	6.6	0	13.4	13.4	5.8	5.8	5.8

(cont.)

task	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C
21	4.6	4.6	5.9	5.9	5.9	2.5	2.5	8.7	0	8.7	13.4	13.4	0	6.0	6.0
22	6.9	6.9	4.0	4.0	4.0	5.7	5.7	2.9	2.9	2.8	1.1	0	6.6	7.3	5.7
23	5.6	0	12.4	12.4	12.4	9.7	8.8	3.3	0	3.3	1.3	1.3	5.8	5.8	5.8
24	2.8	2.8	2.4	2.4	2.0	3.7	3.7	4.0	4.0	4.0	9.3	9.3	9.2	9.2	9.2
25	0	9.0	8.3	8.3	8.8	9.6	9.6	1.2	1.2	1.2	6.7	6.4	3.1	3.1	3.1
26	8.2	8.2	8.2	8.2	8.2	8.8	8.8	9.0	9.0	0	5.3	0	4.7	4.7	4.7
27	3.2	0	2.9	2.9	3.2	4.8	4.8	5.6	5.6	5.6	9.9	9.9	9.7	9.7	0
28	3.4	3.4	2.1	0	1.9	8.0	0	2.0	2.0	2.0	0	3.0	6.7	6.7	6.7
29	2.9	2.6	2.3	2.3	2.3	5.6	5.6	7.5	7.5	7.5	3.7	3.7	1.8	1.8	1.8
30	13.9	13.9	12.6	12.6	12.6	4.0	4.0	2.5	2.5	2.5	3.2	3.2	4.3	4.3	4.3
31	6.2	6.2	0	8.0	8.0	4.8	4.4	8.8	8.8	8.8	1.1	1.1	8.4	8.4	8.3
32	6.6	6.6	11.6	11.6	11.6	8.6	8.6	9.7	10.1	9.7	8.8	8.8	3.1	3.1	3.1
33	6.9	0	1.5	1.5	1.5	10	8.9	3.6	3.6	3.6	6.4	6.4	8.0	8.0	8.0
34	2.4	2.4	1.1	1.1	1.1	5.4	5.4	6.3	5.9	6.3	7.4	7.4	3.4	3.4	3.4
35	8.2	8.2	6.0	6.0	6.0	4.7	5.4	0	13.0	0	7.1	7.0	3.9	3.7	3.9
36						9.4	9.4	4.7	4.7	4.7	6.2	6.9	9.7	0	0
37						1.0	1.0	7.9	7.9	7.9	0	14.0	4.9	4.9	4.9
38						7.3	6.9	5.6	6.1	5.6	6.5	6.5	4.4	4.4	4.4
39						4.1	4.1	2.2	2.2	0	9.2	8.4	2.1	2.1	2.1
40						1.2	1.4	8.7	8.7	8.7	4.3	4.3	14.0	14.6	14.0
41						1.1	1.0	3.6	3.6	3.6	0	6.0	9.4	0	9.4
42						2.4	2.4	1.6	1.5	0	6.4	6.4	5.0	5.0	5.0
43						1.7	1.7	1.2	1.2	0	7.9	7.9	5.2	5.2	5.2
44						12.3	13.5	1.7	1.7	1.7	3.8	4.0	0	9.0	9.0
45						2.5	2.5	4.9	4.9	4.9	4.8	0	9.6	9.6	9.6
46											2.3	0	1.3	1.3	1.3
47											8.5	8.5	3.0	3.0	3.0
48											4.6	4.8	7.6	7.6	7.6
49											13.6	13.6	0	2.0	2.0
50											3.6	3.6	8.4	8.4	8.4
51											9.2	9.2	3.8	3.8	3.8
52											1.5	1.5	3.3	3.3	3.3
53											5.1	0	7.9	7.9	7.9
54											4.4	4.4	4.6	4.2	5.1
55											3.8	3.8	13.6	13.6	13.6
56											6.7	6.7	5.9	5.9	6.0
57											11.3	11.3	9.5	9.5	9.5
58											7.0	0	7.7	0	7.7
59											2.2	2.2	4.4	4.4	4.7
60											15.0	15.0	3.9	3.9	3.9
61											8.4	8.4	10.0	11.2	10.0
62											1.3	1.3	8.8	8.8	8.8
63											1.7	1.7	0	6.0	6.0
64											7.5	7.5	0	7.0	0
65											5.1	5.1	4.5	4.5	4.5
66											0	5.0	3.7	3.7	3.7
67											1.4	1.6	1.1	0	1.0
68											6.5	7.1	2.9	2.9	2.9
69											3.9	3.9	8.7	8.4	8.5
70											2.9	2.9	2.7	3.1	2.7

A2.2.2 MALBP data set with random task times (in time units)

	Problem 1		Problem 2			Problem 3		Problem 4			Problem 5		Problem 6			Problem 7	
task	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B
1	4.1	6.4	13.9	10.4	0	3.6	2.7	1.2	9.7	13.7	0.6	6.5	3.0	16.1	1.8	18.5	1.7
2	4.9	12.6	2.5	12.5	13.8	9.9	7.9	13.8	1.2	14.8	13.4	5.4	15.7	0.6	4.1	16.7	9.9
3	15.1	13.3	6.7	4.6	2.4	1.3	9.0	9.3	14.2	14.5	9.2	14.2	13.9	1.5	5.4	14.8	10.0
4	8.5	6.4	10.8	1.5	5.3	2.9	7.5	7.2	11.6	14.9	12.3	12.1	15.9	6.0	2.0	11.9	14.8
5	7.0	8.1	7.8	3.1	17.9	2.1	2.5	10.1	0.5	1.4	7.2	3.1	11.0	5.3	2.3	15.0	7.4
6	1.8	19.4	19.9	7.1	4.5	0.5	18.5	3.6	15.2	12.8	1.4	10.0	19.1	19.0	12.4	6.2	13.1
7	8.2	13.5	5.7	12.7	3.6	1.2	3.1	1.2	12.6	19.3	3.8	1.3	17.0	1.4	12.8	9.9	6.3
8	17.3	16.8	1.6	6.9	8.5	6.0	19.9	13.8	6.3	7.5	9.2	9.3	9.6	7.2	3.6	2.4	5.4
9						10.9	6.9	9.7	5.9	2.4	11.9	8.0	7.9	7.9	8.3	11.1	1.6
10						16.5	7.5	6.0	8.3	10.7	8.1	11.8	5.6	3.5	15.6	6.8	13.7
11						4.0	0.4	7.4	0.3	8.4	11.8	1.8	10.8	7.6	9.1	15.8	11.5
12											13.7	20.0	13.3	5.3	17.3	14.7	10.6
13											17.1	4.0	12.7	14.4	4.9	11.0	7.8
14											12.4	0.9	9.7	2.5	0.4	7.7	18.0
15											11.1	12.2	16.8	12.8	15.6	10.2	18.9
16											12.0	3.0	6.9	12.9	0	14.5	6.8
17											14.0	10.9	12.9	14.3	2.1	17.5	19.9
18											14.2	6.4	7.2	15.0	4.1	16.2	2.7
19											3.8	7.2	14.6	2.2	4.3	9.0	17.5
20											15.6	10.9	11.9	10.3	18.2	11.5	10.4
21											17.8	8.9	14.9	10.6	2.8	15.9	16.8
22																18.1	4.8
23																1.8	13.3
24																14.1	3.7
25																4.7	10.3

	Problem 8			Problem 9		Problem 10			Problem 11		Problem 12			Problem 13		Problem 14		
task	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C
1	10.8	17.9	6.5	9.7	1.6	11.8	3.0	0.5	1.3	4.5	10.5	19.3	1.5	8.2	14.7	13.2	18.5	12.0
2	15.6	8.5	0.0	4.0	0.8	8.6	3.1	7.7	10.0	10.2	5.6	3.0	8.3	11.0	19.8	1.6	15.6	15.7
3	2.1	4.9	18.2	6.4	11.7	16.6	13.2	15.9	8.8	8.2	18.4	12.7	13.1	0.2	18.2	8.4	5.6	19.1
4	10.0	16.9	8.3	10.7	8.8	6.2	14.9	2.2	10.0	6.5	5.5	1.4	19.1	13.5	8.2	11.8	12.6	8.7
5	13.8	16.6	4.9	3.2	6.3	3.7	0.1	14.6	10.9	15.0	16.9	7.3	12.8	1.5	19.6	16.1	8.5	3.7
6	11.7	16.3	5.9	16.8	12.1	13.4	7.7	4.1	13.9	2.7	3.7	15.7	18.1	9.6	4.9	11.4	3.1	9.8
7	0.2	4.3	11.7	5.6	7.7	4.7	5.8	14.6	12.2	4.7	1.5	3.7	12.5	18.7	5.7	5.8	13.4	3.8
8	16.3	14.4	11.9	12.1	9.8	18.1	17.8	4.8	2.2	4.0	19.3	9.7	6.9	0.8	18.5	1.8	8.2	10.5
9	16.1	17.9	4.5	0.1	3.4	0.2	9.1	10.4	13.8	17.9	10.7	10.5	15.2	19.7	8.0	10.9	11.6	14.0
10	19.9	2.8	2.6	8.1	17.6	4.1	10.1	10.9	6.2	7.8	11.1	19.9	19.0	2.1	3.7	16.7	6.3	12.1
11	10.5	11.1	9.8	12.1	19.8	11.9	19.3	8.7	0.8	3.2	6.6	9.3	1.6	15.9	18.2	13.3	14.6	14.9
12	7.6	15.9	10.9	19.5	0.6	11.2	0.5	7.9	8.3	13.2	15.7	19.3	18.1	1.7	12.3	3.2	7.3	15.5
13	18.3	6.9	13.5	15.0	15.6	3.5	19.4	17.7	5.6	4.8	3.9	12.3	11.6	7.8	3.2	4.3	15.5	19.0
14	3.9	8.3	18.8	11.7	0.9	16.6	6.4	14.6	0.2	10.2	12.6	10.0	2.6	5.3	9.8	3.6	4.7	19.7
15	11.8	17.2	4.4	2.2	7.4	16.0	15.4	8.3	12.1	13.5	17.9	8.8	3.2	17.6	6.3	1.0	8.2	9.8
16	15.3	2.7	14.7	2.5	15.7	6.1	3.7	5.4	15.5	4.4	12.7	13.8	11.5	16.9	2.7	0.4	12.9	5.5
17	19.2	2.0	0.1	15.0	16.2	0.5	1.2	10.3	7.6	0.4	15.6	6.3	10.2	18.5	8.0	9.5	16.9	3.4
18	3.9	14.1	1.3	16.0	4.7	7.8	11.4	0.6	15.3	2.9	13.0	1.2	15.7	4.8	1.7	3.3	5.7	5.2
19	11.5	3.9	5.0	0.5	16.0	10.0	16.4	9.4	16.0	11.7	18.4	11.8	19.1	10.8	14.0	18.3	9.3	11.6
20	10.8	14.0	0.4	11.6	0.4	7.8	5.7	5.6	16.7	19.3	1.8	18.7	12.8	3.9	5.1	19.1	7.2	16.4
21	12.8	15.1	5.6	9.3	10.2	4.7	10.3	14.0	15.0	11.4	17.0	15.8	14.4	1.7	19.2	8.1	14.3	13.0
22	19.5	15.9	4.4	8.0	11.0	1.2	16.9	2.6	14.6	14.4	1.9	15.5	14.4	16.9	14.6	2.8	2.2	18.8
23	3.6	13.6	0.5	17.8	13.4	18.4	7.8	15.0	16.6	2.8	15.1	4.1	10.2	12.4	13.0	11.1	2.4	3.7
24	1.7	4.4	19.1	17.8	15.1	1.6	9.0	2.5	1.4	8.0	0.4	6.0	8.1	4.1	3.5	16.9	1.3	3.9

(cont.)

task	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C
25	17.2	12.5	9.0	0.2	2.8	14.2	17.7	14.6	12.8	4.8	4.5	13.8	11.0	15.4	16.0	12.1	18.3	3.3
26				19.4	12.2	13.5	3.7	13.3	7.1	15.2	3.2	8.3	5.4	3.9	12.7	16.6	13.9	2.8
27				12.4	16.5	13.8	11.5	7.5	10.9	9.8	9.5	14.3	17.5	19.2	5.9	8.1	9.6	7.1
28				16.7	8.7	2.1	1.6	13.7	14.3	18.2	4.3	0.9	17.7	19.4	3.9	17.3	15.9	15.7
29									6.5	16.8	10.7	14.7	2.6	8.7	18.1	9.5	16.8	8.9
30									16.5	3.4	14.0	4.3	8.1	0.5	15.4	12.3	18.0	17.8
31														6.7	2.8	4.4	12.2	5.8
32														2.6	13.2	3.6	11.8	12.5

	Problem 15		**Problem 16**			**Problem 17**		**Problem 18**			**Problem 19**		**Problem 20**		
task	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C
1	5.2	16.8	9.3	9.3	16.6	17.8	2.7	1.7	12.3	8.8	3.9	17.8	17.9	18.7	10.6
2	9.7	6.9	9.9	12.7	4.7	19.3	18.0	2.3	16.2	17.7	14.8	12.0	17.6	4.4	13.9
3	6.1	6.0	2.7	5.1	17.8	17.3	7.2	3.1	19.3	8.7	11.4	5.9	1.5	5.3	16.5
4	9.2	7.7	6.6	12.6	6.3	16.4	7.4	16.4	19.6	10.3	4.7	10.0	15.8	4.2	16.1
5	7.6	3.2	13.7	8.3	15.0	0.4	18.5	16.0	12.0	2.9	2.5	2.4	9.4	10.8	13.7
6	2.5	9.0	13.2	16.6	19.3	12.0	14.1	9.0	18.7	1.4	13.3	1.1	5.7	10.4	18.4
7	18.9	17.7	14.8	19.9	10.6	0.1	8.6	5.7	3.0	10.3	8.1	17.9	12.7	16.1	14.8
8	3.8	15.7	2.7	5.5	7.7	14.4	5.9	1.0	14.6	9.6	15.5	19.5	16.0	13.5	15.5
9	4.0	13.8	4.8	2.1	7.0	8.3	16.1	10.0	9.9	12.4	7.7	5.6	19.9	8.1	2.0
10	4.1	1.0	9.7	12.5	17.4	10.0	0.1	14.7	4.6	13.9	3.5	9.0	2.6	2.9	11.9
11	17.4	5.4	0.0	3.5	6.6	4.0	17.3	5.1	5.7	17.1	18.6	1.7	12.3	19.6	18.2
12	14.6	10.0	2.6	19.5	13.8	15.7	4.0	17.0	18.6	19.2	10.1	6.9	4.2	2.5	11.0
13	9.6	16.4	18.3	14.4	9.9	18.5	16.8	18.2	13.6	9.4	14.1	9.7	13.2	17.4	13.5
14	11.8	10.6	19.1	4.5	18.7	19.1	18.9	17.7	16.3	14.2	16.3	13.2	4.3	19.4	16.1
15	10.5	14.4	14.4	9.4	19.1	8.7	7.4	5.9	11.6	15.6	5.2	19.5	11.8	1.8	5.0
16	2.9	2.5	5.0	17.1	3.3	18.7	5.1	13.1	14.3	10.4	8.0	6.5	9.9	13.8	0.5
17	4.0	17.4	1.1	13.0	14.7	7.4	13.3	19.4	17.5	15.1	13.9	16.2	4.5	19.5	3.6
18	12.2	11.3	3.9	12.4	7.2	17.0	15.9	12.4	4.4	3.7	5.5	10.4	9.0	13.2	19.5
19	2.3	14.6	19.4	5.4	12.1	18.7	19.8	11.5	10.7	15.1	16.1	9.8	17.1	7.7	10.5
20	12.3	3.5	7.5	8.6	15.2	10.2	2.2	0.4	2.9	14.8	0.5	19.9	13.5	3.1	16.3
21	9.1	9.5	0.6	11.5	1.5	0.9	0.4	14.8	3.5	15.3	6.6	0.5	15.5	8.1	4.9
22	3.1	0.3	15.5	3.7	10.3	13.8	1.1	15.5	19.9	17.2	14.5	4.2	18.0	0.1	5.3
23	1.1	15.9	1.4	4.0	6.4	4.3	9.1	6.7	3.7	18.8	1.3	6.8	3.1	7.1	14.4
24	13.0	12.2	10.7	16.1	15.8	4.7	13.6	1.6	5.7	14.0	18.5	7.9	6.2	7.4	11.3
25	4.7	4.5	4.9	13.7	2.5	13.6	12.3	19.9	16.8	13.3	6.3	8.1	14.2	19.9	15.7
26	0.8	19.3	9.9	10.2	10.5	13.9	1.8	19.0	16.7	2.0	19.9	12.8	8.7	13.7	12.2
27	17.4	9.8	0.8	18.3	5.6	10.8	11.3	10.6	9.3	10.8	3.1	8.7	16.9	7.1	2.6
28	17.8	13.2	14.9	17.3	16.5	4.8	15.5	7.1	8.7	19.5	4.0	2.4	15.8	5.9	3.5
29	13.0	17.6	9.3	3.3	15.9	2.6	7.7	12.6	19.2	16.6	3.9	18.2	17.9	10.5	5.4
30	19.7	6.2	0.2	5.9	6.0	7.3	8.5	11.6	11.8	8.4	17.4	11.8	9.3	13.7	1.1
31	19.2	14.0	3.5	5.7	7.4	6.0	15.3	9.6	11.0	2.8	1.1	4.3	7.2	3.1	6.5
32	17.5	14.1	14.9	4.1	16.1	5.0	19.9	12.4	19.4	8.7	9.1	3.3	10.0	3.5	19.1
33	19.8	5.4	8.3	0.3	13.3	4.1	17.1	1.6	2.0	10.9	4.0	0.0	7.5	12.3	2.4
34	18.9	1.9	9.0	7.8	14.6	11.6	15.1	10.8	13.6	8.0	15.5	14.4	7.2	16.2	7.7
35	0.3	11.0	9.2	16.7	9.9	1.1	19.9	8.8	13.1	8.4	10.7	16.8	13.4	11.9	18.0
36						12.1	11.3	11.1	10.7	4.3	9.8	4.8	2.3	4.0	14.0
37						2.5	11.6	5.3	7.6	14.2	0.6	12.9	11.6	2.2	1.7
38						2.7	8.3	5.3	19.2	4.7	2.8	2.1	0.1	2.3	18.0
39						0.3	12.2	10.7	6.4	15.5	6.9	17.2	19.2	2.7	8.0
40						4.2	12.1	19.5	7.7	3.7	9.2	0.7	17.9	4.1	18.5
41						8.5	3.0	17.2	9.8	6.3	12.9	5.2	15.0	11.5	9.5
42						5.7	6.2	15.4	2.9	0.8	8.9	0.4	5.6	5.2	16.0
43						16.1	17.0	12.9	6.4	8.1	18.1	12.5	17.4	9.0	9.9
44						3.0	6.1	18.7	12.4	7.2	12.0	0.7	13.2	17.7	1.7

(cont.)

task	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C	t_A	t_B	t_A	t_B	t_C
45						10.1	10.4	10.5	4.7	0.9	18.1	4.9	5.1	0.8	18.8
46											6.0	14.3	9.9	11.9	15.0
47											1.3	16.3	17.8	4.6	14.5
48											8.6	1.7	3.4	2.9	4.7
49											18.5	11.6	1.2	5.2	13.1
50											2.9	18.4	11.6	15.0	6.5
51											8.6	2.9	3.3	7.8	4.6
52											6.6	6.7	19.6	15.8	12.1
53											0	17.6	0.0	3.9	2.3
54											10.5	3.6	6.6	12.5	6.1
55											16.9	17.0	2.6	11.5	19.4
56											17.7	3.9	14.2	2.5	2.0
57											15.5	8.0	3.0	6.8	18.8
58											12.1	19.4	1.5	17.4	6.9
59											18.6	0.8	6.9	1.2	14.5
60											0.9	8.9	3.1	7.9	7.9
61											9.3	6.6	14.6	3.0	15.4
62											18.0	12.1	13.8	11.9	12.1
63											19.8	1.8	1.2	9.4	15.9
64											13.0	13.7	11.4	1.0	3.9
65											3.5	15.5	15.6	9.6	15.0
66											5.5	14.5	5.2	8.9	3.9
67											17.1	9.0	12.6	10.6	12.3
68											6.7	19.9	10.5	12.0	15.3
69											4.7	0.8	8.2	15.0	5.9
70											12.4	1.9	9.9	17.6	8.5

Appendix 3

Computation of LB_{pmix} for problems with maximum task processing time less or equal to 2C

A lower bound for the mixed-model assembly line balancing problem with parallel workstations, LB_{pmix}, was derived using the following set of assumptions:

(i) the maximum number of replicas per workstation is given by $MAXT_k=\lceil t_{max}/C \rceil$, where t_{max} is the processing time of the longest task assigned to workstation *k*,

(ii) a workstation can be duplicated only if the task time of one of the tasks assigned to it exceeds the cycle time ($MRT=C$), and

(iii) the task time of the longest task does not exceed twice the cycle time ($t_{max} \leq 2C$).

The steps required to compute LB_{pmix} are described as follows and illustrated for the numerical example introduced in section 4.3.3:

Step 1: For each model, classify the tasks according to the corresponding task time, as shown in Table A3.1.

Table A3.1 – Classification of tasks to compute LB_{pmix}

		Tasks	
Task type	**Task time**	**Model A**	**Model B**
A	$\frac{5}{3}C < t_A \leq 2C$	-	-
B	$\frac{4}{3}C < t_B \leq \frac{5}{3}C$	4	4
C	$C < t_C \leq \frac{4}{3}C$	25	-
D	$\frac{2}{3}C < t_D \leq C$	2,3,5,12,18,23	2,3,5,12,18,20,23
E	$\frac{1}{3}C < t_E \leq \frac{2}{3}C$	6,7,9,11,13,15,17,22,24	9,11,13,15,22,24
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	$t_J < \frac{1}{3}C$	1,8,10,14,16,19,20,21	1,6,7,8,10,14,16,17,19,21,25

Step 2: For each model, compute *LB'(m)*.

$$LB'(m) = \left\lceil 2(n_A + n_B + n_C) + y(n_D - n_C) + \frac{1}{2}w(n_E - n_B) + \frac{5}{3}n_F + \frac{4}{3}n_G + \frac{2}{3}n_H + \frac{1}{3}n_I \right\rceil \quad \textbf{(A3.1)}$$

LB'(m) is the main term of $LB_{pmix}(m)$. It is derived from one of the lower bounds, LB_3, defined by Scholl (1999) for the SALPB and adapted to the parallel workstations problem. *LB'(m)* is given by expression A3.1, where n_X is the number of tasks of type X ($X=A,\ldots,J$), y equals 1 if $n_D-n_C>0$ or zero otherwise and w equals 1 if $n_E-n_B>0$ or zero otherwise. The reasoning for this computation is as follows.

The workstations performing tasks of types *A*, *B* or *C* (whose task time is longer than the cycle time) need to be duplicated. As two tasks of any of these types cannot share the same workstation, because the maximum number of replicas allowed would be exceeded, a lower bound for the overall number of workstations (including replicas) is twice the number of tasks of types *A*, *B* and *C*. Each task of type *D* can be combined with a task of type *C* in a duplicated workstation, however if there are not enough duplicated workstations of type *C* to accommodate the tasks of type *D*, each of these remaining tasks will require a workstation. The same reasoning applies to tasks of type *E*, that is, two tasks of type *E* require a single workstation, but can also be combined with a duplicated workstation performing tasks of type *B*. Finally, the tasks of types *F*, *G*, *H* and *I* have a fixed task time and so they occupy a fraction of a workstation corresponding to the ratio between their task time and the cycle time.

For the numerical example, the values of *LB'(A)* and *LB'(B)* are computed as follows:

$$LB'(A) = \left\lceil \left(2(0+1+1) + (6-1) + \frac{1}{2}(9-1) \right) \right\rceil = 13$$

$$LB'(B) = \left\lceil \left(2(0+1+0) + (7-0) + \frac{1}{2}(6-1) \right) \right\rceil = 12$$

Step 3: For each model, compute *Z(m)*.

$$Z(m) = \left\lceil \left\lfloor \sum_{i=J} t_{im} - \left(LB' \cdot C - \sum_{i \neq J} t_{im} \right) \right\rfloor \Big/ C \right\rceil \qquad \textbf{(A3.2)}$$

A second term is added to *LB'(m)* to compute the value of LB_{pmix}. This term adds up the number of workstations needed to process tasks of type *J*, which in most real world problems account for a large proportion of the workstations. Because these tasks can easily be included in workstations that perform tasks of the other types, it is necessary to verify if, after filling up these workstations, there are tasks of type *J* remaining to create new

workstations. The minimum number of workstations (*Z(m)*) required to perform tasks of type *J*, after filling up the remaining capacity of the workstation assigned to other task types is then given by expression A3.2.

For the numerical example, the values of *Z(A)* and *Z(B)* are computed as follows:

$$Z(A) = \lceil [9 - (13 \times 10 - 123.2)]/10 \rceil = 1$$

$$Z(B) = \lceil [11.8 - (12 \times 10 - 104.4)]/10 \rceil = 0$$

Step 4: For each model, compute $LB_{pmix}(m) = LB' + Z(m)$. For the numerical example, $LB_{pmix}(A) = 14$ and $LB_{pmix}(B) = 12$.

Step 5: Select LB_{pmix} for the problem. $LB_{pmix} = \max_m[LB_{pmix}(m)]$. For the numerical example, $LB_{pmix} = LB_{pmix}(A) = 14$.

Appendix 4

Demonstration of the maximum and minimum values of functions B_b^U and B_w^U

A4.1 Function B_b^U (balance between workstations of a U-line)

$$Minimise \quad B_b^U = \frac{S}{S-1}\sum_{k=1}^{S}\left(\frac{S_k}{WIT^U}-\frac{1}{S}\right)^2$$

- **Minimum value** – B_b^U reaches the minimum value of zero when WIT^U is evenly distributed between workstations (best case).
  - for all (S) workstations: $S_k = \frac{WIT^U}{S}$

$$B_b^U = \frac{S}{S-1}S\left(\frac{\frac{WIT^U}{S}}{WIT}-\frac{1}{S}\right)^2 = \frac{S}{S-1}S\left(\frac{1}{S}-\frac{1}{S}\right)^2 = 0$$

- **Maximum value** – B_b^U reaches the maximum value of 1 when WIT^U is only accountable to one workstation (worst case).
  - for one workstation: $S_k = WIT^U$
  - for S-1 workstations: $S_k = 0$

$$B_b^U = \frac{S}{S-1}\left(\left(\frac{WIT^U}{WIT^U}-\frac{1}{S}\right)^2 + (S-1)\left(\frac{0}{WIT^U}-\frac{1}{S}\right)^2\right) =$$

$$= \frac{S}{S-1}\left(\left(1-\frac{1}{S}\right)^2 + (S-1)\left(\frac{1}{S}\right)^2\right) = \frac{S}{S-1}\cdot\frac{S-1}{S^2}\cdot\left((S-1)+1\right) = 1$$

A4.2 Function B_w^U (balance within workstations of a U-line)

$$Minimise \quad B_w^U = \frac{M^2}{S(M^2-1)}\sum_{k=1}^{S}\sum_{m=1}^{M}\sum_{n=1}^{M}\left(\frac{q_{mn}s_{kmn}}{S_k}-\frac{1}{M^2}\right)^2$$

- **Minimum value** – B_w^U reaches the minimum value of zero when, for all workstations, S_k is evenly distributed among all model combinations (best case).
  - for all (S) workstations and all (M^2) model combinations: $q_{mn}s_{kmn} = \frac{S_k}{M^2}$

$$B_w^U = \frac{M^2}{S(M^2-1)} S \cdot M^2 \cdot \left(\frac{\frac{S_k}{M^2}}{S_k} - \frac{1}{M^2} \right)^2 = \frac{M^2}{S(M^2-1)} S \cdot M^2 \cdot \left(\frac{1}{M^2} - \frac{1}{M^2} \right)^2 = 0$$

- **Maximum value** – B_w^U reaches the maximum value of 1 when, for all workstations, S_k is only accountable to one model combination (worst case).
  - for one model combination: $q_{mn} s_{kmn} = S_k$
  - for M^2-1 models: $q_{mn} s_{kmn} = 0$

$$B_w^U = \frac{M^2}{S(M^2-1)} S \left(\left(\frac{S_k}{S_k} - \frac{1}{M^2} \right)^2 + (M^2-1) \left(\frac{0}{S_k} - \frac{1}{M^2} \right)^2 \right) =$$

$$= \frac{M^2}{S(M^2-1)} S \left(\left(1 - \frac{1}{M^2} \right)^2 + (M^2-1) \left(\frac{1}{M^2} \right)^2 \right) = \frac{M^2}{S(M^2-1)} S \cdot \frac{M^2-1}{M^2} \cdot \left(\left(M^2-1 \right) + 1 \right) = 1$$

Appendix 5

Computation of LB_{pmix} for problems with maximum task processing time less or equal to 5C

The lower bound for the mixed-model assembly line balancing problem with parallel workstations, LB_{pmix}, proposed described in Appendix 3 was adapted to take into account the following set of assumptions:

(i) the maximum number of replicas of each workstation is given by $MAXP_k=\lceil t_{max}/C \rceil$, where t_{max} is the processing time of the longest task assigned to workstation k,

(ii) a workstation can be replicated only if the task time of one of the tasks assigned to it exceeds the cycle time ($MRT=C$), and

(iii) the task time of the longest task does not exceed five times the cycle time ($t_{max} \leq 5C$).

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

Step 1: For each model, classify tasks according to the corresponding task time, as shown in table A5.1.

Table A.1 – Classification of tasks to compute LB_{pmix}

Task type	Task time	Task type	Task time	Task type	Task time
A	$\frac{14}{3}C < t_A \leq 5C$	I	$2C < t_I \leq \frac{7}{3}C$	R	$t_R = \frac{10}{3}C$
B	$\frac{13}{3}C < t_B \leq \frac{14}{3}C$	J	$\frac{5}{3}C < t_J \leq 2C$	S	$t_{S=}\frac{8}{3}C$
C	$4C < t_C \leq \frac{13}{3}C$	K	$\frac{4}{3}C < t_K \leq \frac{5}{3}C$	T	$t_T = \frac{7}{3}C$
D	$\frac{11}{3}C < t_D \leq 4C$	L	$C < t_L \leq \frac{4}{3}C$	U	$t_U = \frac{5}{3}C$
E	$\frac{10}{3}C < t_E \leq \frac{11}{3}C$	M	$\frac{2}{3}C < t_M \leq C$	V	$t_V = \frac{4}{3}C$
F	$3C < t_F \leq \frac{10}{3}C$	N	$\frac{1}{3}C < t_N \leq \frac{2}{3}C$	W	$t_W = \frac{2}{3}C$
G	$\frac{8}{3}C < t_G \leq 3C$	O	$t_O = \frac{14}{3}C$	X	$t_X = \frac{1}{3}C$
H	$\frac{7}{3}C < t_H \leq \frac{8}{3}C$	P	$t_P < \frac{13}{3}C$	Y	$t_Y < \frac{1}{3}C$
I	$2C < t_I \leq \frac{7}{3}C$	Q	$t_Q = \frac{11}{3}C$		

Step 2: For each model, compute $LB'(m) = \lceil LB''(m) \rceil$, where

$$LB''(m)=5(n_A+n_B+n_C)+4(n_D+n_E+n_F)+3(n_G+n_H+n_I)+2(n_J+n_K+n_L)+y(n_M-n_C-n_F-n_I-n_L)$$
$$+(1/2)w(n_N-n_B-n_E-n_H-n_K)+(14/3)n_O+(13/3)n_P+(11/3)n_Q+(10/3)n_R+(8/3)n_S$$
$$+(7/3)n_T+(5/3)n_U+(4/3)n_V+(2/3)n_W+(1/3)n_X \quad \textbf{(A5.1)}$$

where n_X is the number of tasks of type i (i=A,…,X), y equals 1 if n_M-n_C-n_F-n_I-n_L >0 or zero otherwise and w equals 1 if n_N-n_B-n_E-n_H-n_K >0 or zero otherwise. The reasoning for this computation is as follows.

The workstations performing tasks whose processing time is longer than the cycle time (tasks of types A to L) need to be replicated. As two tasks of any of these types cannot share the same workstation, because the value of $MAXP_k$ would be exceeded, a lower bound for the overall number of workstations (including replicas) is the number of tasks of each type multiplied by the number of replicas created in a workstation by the assignment of each task (for instance, each task of type A, B, and C will create a workstation with 5 replicas, because they have processing times between 4C and 5C). Each task of type M can be combined with a task of type C, F, J or L in a replicated workstation, however if there are not enough replicated workstations to accommodate the tasks of type M, each of these remaining tasks will require a workstation. The same reasoning applies to tasks of type N, that is, two tasks of type N require a single workstation, but they can also be combined with a replicated workstation performing tasks of type B, E, H or K. Finally, the tasks of types O to X have a fixed task time and so they occupy a fraction of a workstation corresponding to the ratio between their task time and the cycle time.

Step 3: For each model, compute *Z(m)*.

Z(m) adds up the number of workstations needed to process tasks of type Y. Because these tasks can easily be included in workstations that perform tasks of the other types, it is necessary to verify if, after filling up these workstations, there are tasks of type Y remaining to create new workstations. The minimum number of workstations required to perform tasks of type Y, after filling up the remaining capacity of the workstation assigned to other task types is then given by:

$$Z(m) = \left\lceil \left[\sum_{i=Y} t_{im} - \left(LB'(m) \cdot C - \sum_{i \neq Y} t_{im} \right) \Big/ C \right] \right\rceil \qquad \textbf{(A5.2)}$$

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

Step 5: Select LB_{pmix} for the problem: $LB_{pmix} = \max_m[LB_{pmix}(m)]$.