UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Organització d'Empreses

Doctoral program:

BUSINESS ADMINISTRATION AND MANAGEMENT

Doctoral thesis:

AWALBP-L2: The Accessibility Windows Assembly Line Balancing Problem Level 2. Formalization and Solution Methods

by

Gema Calleja Sanz

Thesis advisors:

Dr. Albert Corominas Subias
Dr. Alberto García Villoria

Barcelona, Spain. January 2015.

Abstract

Assembly lines are typically found in modern mass production systems. During the design and operation of an assembly line, the so-called *assembly line balancing problem* (ALBP) needs to be solved. It basically consists in assigning a set of tasks to a set of ordered workstations in such a way that specific constraints are fulfilled and an efficiency-based objective is optimized.

A common assumption in the literature on line balancing is the full access of the workstations to the workpieces. However, in various environments of automated assembly, as in the manufacturing of printed circuit boards, the workpieces are larger than the width of the workstations. This implies that, at any given instant, a workstation cannot reach a whole workpiece, but only a restricted portion of one workpiece or two consecutive workpieces. To enable the access of the workstations to different parts of the workpieces, the latter are transported through the line according to a cyclic stepwise pattern called *movement scheme*. Such cycle decomposes into a number of halts, named *stationary stages*, separated by *forward steps*. During a stationary stage each of the workstations can only execute those assigned tasks that are inside its reachable range. After a stationary stage, the workpieces are moved by some common forward step and the next stationary stage begins. Once the cycle ends, a fully assembled workpiece leaves the line, inside which there is the same number of workpieces lying exactly at the same positions as in the start of the cycle.

This doctoral thesis tackles an assembly line balancing problem with restricted access to the workpieces that has been entitled AWALBP: the *Accessibility Windows Assembly Line Balancing Problem*. The problem is described and a general classification for its main optimization levels is proposed. The thesis focuses on a specific case of the optimization level AWALBP-L2. The AWALBP-L2 consists of two subproblems that need to be solved simultaneously: (i) the computation of a feasible movement scheme and (ii) the assignment of each task to one workstation and one stationary stage of the cycle. In the particular case of AWALBP-L2 addressed in this thesis, for each task a single workstation is compatible.

The review of the state of the art reveals that relatively few studies have been published concerning the AWALBP. Regarding the solution of the AWALBP-L2, the only available previous work is a mathematical programming model, but the model is not tested or validated. In order to fill this research gap, the aim of this thesis is three-fold: i) to describe the AWALBP and characterize its main optimization levels, ii) to propose exact methods for the case of AWALBP-L2 considered, and iii) to develop solution procedures for the challenging instances that are out of reach of the former methods.

Consequently, in this doctoral thesis the AWALBP is characterized and the AWALBP-L2 case is addressed through four main approaches. First, the problem is formalized and solved via two mixed integer linear programming (MILP) models. Second, an approach combining a matheuristic and a MILP model is proposed. The third approach considers hybridizing metaheuristics with mathematical programming models. Finally, the fourth approach proposes sequential combinations of the aforementioned hybrid metaheuristics and a MILP model.

The performance of all approaches is evaluated via an extensive computational experiment based on realistic instances, and an optimal solution could be found for a large number of them. Future research work may include additional assumptions on the problem, such as precedence relationships among tasks or several workstations compatible for each task. The methods proposed in this thesis are open in nature and extend perspectives for combining (meta)heuristics and mathematical programming models, either for improving the solution of the AWALBP-L2 or for tackling other combinatorial optimization problems.

Resumen

Las líneas de montaje se encuentran habitualmente en los sistemas modernos de fabricación en serie. Durante el diseño y funcionamiento de una línea de montaje es necesario resolver el problema de equilibrado de líneas de montaje, denominado *assembly line balancing problem* (ALBP). Este consiste básicamente en asignar un conjunto de tareas a un conjunto de estaciones ordenadas, de manera que se cumplan restricciones específicas y se optimice un objetivo de eficiencia dado.

Un supuesto habitual en la literatura de equilibrado de líneas es el acceso total de las estaciones a las piezas. Sin embargo, en varios entornos de montaje automatizado, tales como en la fabricación de placas de circuitos impresos, las piezas son de mayor tamaño que el ancho de las estaciones. Esto implica que, en un instante dado, una estación no puede acceder a una pieza entera, sino únicamente a una porción de una pieza o de dos piezas consecutivas. Para permitir el acceso de las estaciones a las diferentes partes de las piezas, estas son transportadas a lo largo de la línea según un patrón cíclico de pasos denominado *esquema de movimiento*. Dicho ciclo se descompone en un número de paradas, o *etapas estacionarias*, separadas entre ellas por *pasos de avance*. Durante una etapa estacionaria cada estación solamente puede ejecutar aquellas tareas asignadas que están dentro de su región accesible. Tras una etapa estacionaria, las piezas recorren un paso de avance común y la siguiente etapa estacionaria comienza. Una vez finalizado un ciclo una pieza completamente montada abandona la línea, en la cual hay el mismo número de piezas y exactamente en las mismas posiciones que al inicio del ciclo.

Esta tesis doctoral aborda un problema de equilibrado de líneas con acceso limitado a las piezas que ha sido titulado AWALBP: *Accessibility Windows Assembly Line Balancing Problem*. Se describe el problema y se propone una clasificación general de sus principales niveles de optimización. La tesis se centra en un caso específico del nivel AWALBP-L2. El AWALBP-L2 consta de dos subproblemas que deben ser resueltos simultáneamente: (i) cálculo de un esquema de movimiento factible y (ii) asignación de cada tarea a una estación y a una de las etapas estacionarias del ciclo. En el caso particular de AWALBP-L2 tratado en esta tesis, para cada tarea existe una única estación compatible.

La revisión del estado del arte revela que relativamente pocos estudios han sido publicados sobre el AWALBP. Respecto a la resolución del AWALBP-L2, el único trabajo anterior disponible es un modelo de programación matemática, el cual no está probado o validado. Con tal de cubrir este hueco de investigación, el objetivo de la presente tesis es triple: i) describir el AWALBP y caracterizar sus principales niveles de optimización, ii) proponer métodos exactos para el caso considerado de AWALBP-L2, y iii) desarrollar métodos de resolución para los ejemplares más difíciles que quedaron fuera del alcance de los métodos anteriores.

Por consiguiente, en esta tesis doctoral se caracteriza el AWALBP y se aborda el caso de AWALBP-L2 mediante cuatro enfoques principales. En primer lugar, el problema se formaliza y se resuelve mediante dos modelos de programación lineal entera mixta (PLEM). En segundo lugar se propone una mateheurística combinada con un modelo de PLEM. El tercer enfoque consiste en hibridizar metaheurísticas con modelos de programación matemática. Finalmente, el cuarto enfoque propone combinaciones secuenciales de las mencionadas metaheurísticas híbridas con un modelo de PLEM.

Los enfoques propuestos se evalúan mediante una extensa experiencia computacional con ejemplares realistas, y se obtuvo una solución óptima para un gran número de ellos. Las líneas propuestas de investigación futura incluyen supuestos adicionales tales como relaciones de precedencia entre tareas o varias estaciones compatibles para una misma tarea. Los métodos propuestos en esta tesis son de naturaleza abierta y ofrecen perspectivas para la combinación de (meta)heurísticas con modelos de programación matemática, tanto para mejorar la solución del AWALBP-L2 como para abordar otros problemas de optimización combinatoria.

Acknowledgements

First and foremost, my gratitude goes to my two supervisors, Dr. Albert Corominas and Dr. Alberto García. Their patience, continuous support and knowledge were my motivation during my doctorate. I am also grateful to Dr. Rafael Pastor, for reviewing my doctoral research and offering insightful feedback on this thesis. It is a privilege working with all of you.

I would like to thank Ernest Benedito, Amaia Lusa and Jordi Olivella, members of the Enginyeria d'Organització i Logística Industrial (EOLI) research group for their valuable comments and advice.

I want to acknowledge as well my fellows from the Department of Management, Rocío de la Torre and Mariona Vilà, for the professional and personal great moments shared.

I extend my gratitude to the members of the Institute of Industrial and Control Engineering (IOC). Particular thanks to Vicenç, Leo and Enric for all the help with the computers in the lab. Thank you to Carme, Noemi and Marta for both the administrative assistance and good humor. I also wish to thank my colleagues and fellow PhD students in the Institute for their hearty laughs and great conversations. Thank you (in no particular order) to Carlos A., Leo, Orestes, Carlos R., Diana, Fernando, Andrés, Abiud, Isiah, Sergi, Josep, Niliana, Marcos, Ali and Henry. I apologize if I forgot to mention someone.

To my long-life friends, for their support during these four years. Thank you for making my life colorful and rich.

To Josué, for his understanding, support and motivation all the way.

Finally, yet importantly, I owe my sincere appreciation to my family, for always believing in me and encouraging me to go further. I dedicate this work to them.

Table of contents

Chapter 1

Introduction

1.1 Motivation and scope

Balancing assembly lines is a prime focus for the manufacturing industry. In today's world of fierce competition, the balancing of the operations among the workstations is required in order to cope with a market that demands increasingly complex products and ever-shorter innovation cycles. Hence, manufacturers are gearing towards balancing their assembly lines with the aim to strengthen their competitiveness.

In the literature on line balancing the following scenario is usually considered: at any instant of the cycle, there is exactly one workpiece inside each workstation, and all the workpiece is visible from any workstation.

However, the scenario that motivates this thesis is different: the workpieces are longer than the width of the workstations. This implies that each workstation cannot access one entire workpiece, but only a portion of one or two consecutive workpieces. As a result, each workstation may access portions of two consecutive workpieces at any given time, and each workpiece may be accessed by several workstations at any given time.

The optimization of this type of assembly line with restricted access to the workpieces has recently emerged from various kinds of advanced manufacturing, such as in the automated assembly of large printed circuit boards (see Fig. 1). Essentially, such assembly lines use robotic workstations to place electronic components at predefined locations of the workpieces. A number of identical workpieces are to be processed in a cyclic fashion. The workstations are linked together by a transport system, which moves the workpieces forward in steps, according to a cyclic pattern called *movement scheme*. At every halt between two forward steps, there is a *stationary stage*. In a stationary stage, the workstations perform tasks on the workpieces. A task can only be performed if it is visible inside the *accessibility window* of the workstation where it will be executed. After a stationary stage is finished, the workpieces are moved forward by some same distance, and the next stationary stage is entered. This scenario, where the accessibility windows of the workstations do not allow reaching the whole workpiece, gives rise to the problem which we entitle *Accessibility Windows Assembly Line Balancing Problem* (AWALBP).

The scope of this thesis is twofold. First, it presents the AWALBP and proposes a classification for its main optimization levels. Second, it focuses on the solution of a particular case of the optimization level AWALBP-L2. More specifically, the addressed case corresponds to the real-world industrial problem described in Müller-Hannemann and Weihe (2006), where for each

task one single compatible workstation is available. In order to solve this case of AWALBP-L2, two problems are to be solved simultaneously: i) the (cyclic) movement scheme of the workpieces through the workstations, and ii) the assignment of each task to one stationary stage of the cycle, in such a way that the cycle time is minimized.

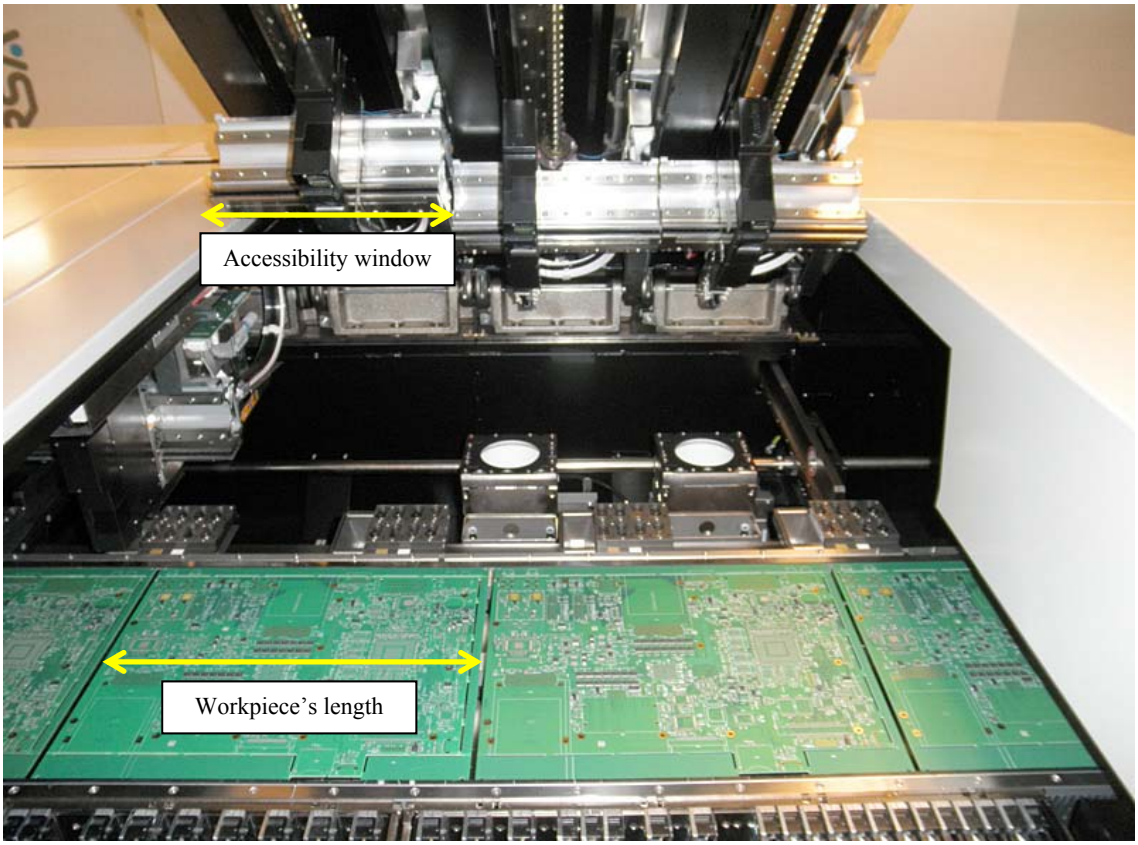

Figure 1: *Inside view of an automated assembly line with accessibility windows*[1].

Just as most assembly line balancing problems, the AWALBP and optimization levels thereof are *NP-hard* due to their combinatorial nature. Even the simplest version of the problem, AWALBP-L1, is already *NP-hard*, as is proven in the article ***Combining matheuristics and MILP to solve the Accessibility Windows Assembly Line Balancing Problem (AWALBP-L2)*** (see Annex A1), and so it is the case of AWALBP-L2 addressed in this thesis (Müller-Hannemann and Weihe, 2006). With this in mind, mathematical programming models are proposed first in order to try to solve the problem optimally. Subsequently, in order to tackle large and difficult instances intractable by mathematical programming models, hybrid procedures are developed.

The thesis is presented in the format of a collection of published articles (jointly with an article in review process and conference papers) in accordance with the regulations of the doctoral program in Business Administration and Management of the Universitat Politècnica de Catalunya (UPC).

[1] *Retrieved November 6, 2014, from http://www.lewis-clark.com/product/assembleon-ax5-camera/.*

1.2 Objectives

The main objective of this thesis is three-fold:

(a) To formally describe the AWALBP and provide a classification of its main optimization levels.

(b) To propose exact methods for solving a specific case belonging to the optimization level AWALBP-L2 and determine the size limit of the instances solved optimally.

(c) To develop solution approaches for solving the challenging large instances.

In order to achieve the main objective, the following sub-objectives are developed:

1. **Definition and classification of the AWALBP.** Define the main characteristics of the problem and provide a classification of its optimization levels. Subsequently, give a detailed description focusing on the specific case of optimization level AWALBP-L2 adressed in this thesis.

2. **State of the art review.** Analyze the previous works on assembly line balancing problems involving cyclic movement schemes with accessibility windows.

3. **Benchmark generation.** Generate a set of benchmark instances for the AWALBP-L2 case and publish it online, such that it can be available to the research community.

4. **Mathematical formulation of the AWALBP-L2 case.** Develop and test mathematical programming models in order to identify the best option regarding performance results and the size limit of the instances that can be solved to optimality.

5. **Design and implementation of solution approaches for challenging instances.** The *NP-hard* nature of the AWALBP-L2 case renders the use of exact methods computationally intractable for many medium to large instances of the problem. Thus it is required to develop solution approaches in order to deal with such challenging instances.

6. **Performance evaluation and comparison of the developed methods.** Conduct computational experiments on the generated set of benchmark instances to test the performance of the proposed solution methods. Based on the analysis of these experimental results, draw conclusions and propose directions for future lines of research.

1.3 Structure of the thesis

This thesis is divided into six chapters, including the present one, as follows.

Chapter 1 presents an overview of the *Accessibility Windows Assembly Line Balancing Problem* (AWALBP) and outlines the aims of this work.

Chapter 2 introduces and characterizes the AWALBP. It gives a classification of its main optimization levels and defines the case of AWALBP-L2 addressed in this thesis.

Chapter 3 gives a state of the art review of the optimization levels of AWALBP that have been tackled in the literature and the solution methods that have been proposed.

Chapter 4 presents the solution methods developed. Firstly, two mathematical programming formulations are proposed. Secondly, an approach combining a matheuristic and a mixed integer linear programming (MILP) model is proposed. Furthermore, three hybrid approaches of metaheuristics and mathematical programming are presented. Finally, sequential combinations of metaheuristics with a MILP model are proposed.

Chapter 5 describes the computational experiments conducted to validate the proposed solution methods and discusses the obtained results.

Chapter 6 gives the conclusions, recommendations for further research, and a list of the publications derived from this thesis.

Finally, in the Annexes, a collection of articles and conference papers is presented, as follows.

Annex A1 contains three articles that have been published in journals included in the JCR.

Annex A2 contains one article submitted to a journal included in the JCR, together with eight communications presented at international conferences.

Throughout this document, references to works derived from this thesis are highlighted with bold italic letters. These references are included in chapter 6.2.

Chapter 2

AWALBP: The Accessibility Windows Assembly Line Balancing Problem

This chapter presents the problem studied in this doctoral thesis, namely the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2). After an introduction to assembly line balancing in Section 2.1, the problem of balancing assembly lines with accessibility windows (AWALBP) is described and a classification for its optimization levels is proposed in Section 2.2. Finally, in section 2.3 the specific case of AWALBP-L2 considered in this thesis is detailed.

2.1 Introduction to assembly line balancing problems

In its most basic form, an assembly line consists of a sequence of workstations placed along a transport mechanism. The workpieces are consecutively launched down the line and moved from one workstation to the next. At each workstation, a specified set of tasks necessary to assemble the product is repeatedly performed. Tasks require a certain time to be processed and are related to each other according to existing technological constraints. Each workstation must complete the tasks with a time limit called the cycle time.

Generally, an important decision problem arising in the management of the assembly line is to determine the assignment of tasks to workstations in such a way that some constraints are satisfied, the workload of each workstation does not exceed the cycle time and an objective function is optimized. Such problem is called assembly line balancing problem (ALBP).

According to Baybars (1986a), both Tonge (1961) and Prenting and Thomopoulos (1974) credit the first analytical statement of the ALBP to Bryton (1954). However, the first published mathematical programming formulation is due to Salveson (1955). Since those pioneer works extensive research has been done in the field of ALBPs, as can be seen in the reviews of Baybars (1986a); Ghosh and Gagnon (1989), Erel and Sarin (1998); Kumar and Mahto (2013); Battaïa and Dolgui (2013); and Pachghare and Dalu (2014).

A well-known early classification of ALBP is the one proposed by Baybars (1986a), which differentiates between the simple ALBP (SALBP), and the general ALBP (GALBP). The SALBP considers a single straight assembly line for only one type of product, and its complexity is significantly reduced by several simplifying assumptions with respect to practice.

Despite these simplifying assumptions, SALBP is known to be *NP-hard* (Wee and Magazine, 1986). As a result, SALBP has been studied intensively in the literature, and numerous operations research techniques have been developed to solve this problem to optimality or approximately. Several heuristics (e.g. Helgeson and Birnie, 1961; Pinto, 1978; Baybars, 1986b; Talbot *et al*., 1986; Boctor, 1995; Scholl, 1997) and exact methods (e.g. Bowman, 1960; Johnson, 1981; Baybars, 1986a; Hoffmann, 1992; Erel and Sarin, 1998; Scholl, 1999; Scholl and Becker, 2006) have been proposed. However, recent publications show that this topic remains challenging (e.g. Pastor and Ferrer, 2009; Sewell and Jacobson, 2012; Morrison, 2014; Pape, 2015). GALBP, on the other hand, includes those problems incorporating further characteristics and constraints in order to address more realistic line configurations and manufacturing contexts. This class of problems is very large and contains all extensions that are relevant in practice including parallel workstations (Inman and Leon, 1994), multiple assembly lines (Lusa, 2008), multi-product lines (Pastor *et al*., 2002), mixed models (Akpinar, 2014), U-shaped lines (Miltenburg, 2002), setup times (Andrés *et al*., 2008), resource constraints (Corominas *et al*. (2011), processing task times that depend on the sequence (Kalayci, 2014), or on the worker (Corominas *et al*., 2008), or are stochastic (Dong, 2014). An overview of GALBP variations can be found in Becker and Scholl (2007).

2.1.1 Classification

Despite the classification of Baybars (1986a) is frequently used in the literature, it is insufficient to reflect the ever-growing heterogeneity of GALB problems. Consequently, more detailed classifications have been proposed in order to structure the research field of ALBP and provide a common taxonomy for researchers and practitioners. Such classifications use a compact notation incorporating a significant number of features to describe real assembly systems. Some classification schemes based on condensed notation include the ones proposed by Hao (2005), Boysen *et al*. (2007) and Battaïa and Dolgui (2013).

Based on the aforementioned proposed schemes, the following classification summarizes some of the most relevant attributes of assembly lines according to: the number of products or models produced, shape or layout of the line, task attributes, the workpieces flow, and the level of automation of the line. Whitout the aim to be exhaustive some references are included below to illustrate each attribute.

According to the number of models

- *Single-model line*. It is the standard configuration where only one model of a unique product is produced (Kara *et al*., 2009; Dolgui and Proth, 2010; Dou *et al*., 2011).

- *Mixed-model line*. Several variants from a basic product, referred to as models, are manufactured simultaneously. The production process does not involve setup times since all models require very similar manufacturing tasks (Erel and Gökçen, 1999; Yang *et al*., 2011; Tonelli *et al*., 2013).

- *Multi-model line*. Several different models are produced in separate batches with setup times between them (Van Zante-de Fokkert and de Kok, 1997; Hao and Wei, 2013).

According to line layout

- *Basic straight lines*. Each workpiece visits a sequence of workstations in their order of installation (Gökçen *et al*., 2010, Mohd-Hafizuddin *et al*., 2012).

- *Straight lines with multiple workplaces*. Workstations are aligned in a serial manner. However, at each workstation, a number of parallel workplaces (Scholl and Boysen, 2009;

Delorme *et al*., 2012), or serial workplaces (Guschinskaya *et al*., 2008) are installed in such a way that the workers or the resources associated with each workplace can operate simultaneously or sequentially on each workpiece, respectively.

- *U-shaped lines*. The workstations are arranged in a U-shaped line and have both the entrance at the exit in the same place. Being commonly manual lines, workers may walk from one leg to another of the line. Therefore they can work during the same cycle on two or more workpieces at different positions of the line (Miltenburg and Wijngaard, 1994; Jayaswal and Argawal, 2014).

- *Two-sided lines*. This type of line consists of two serial lines in parallel (Bartholdi, 1993; Kim *et al*., 2000; Özcan and Toklu, 2009), in which pairs of opposite workstations (left-hand side and right-hand side) process simultaneously the same workpiece.

- *Circular transfer lines*. The workstations are installed around a rotating table (o similar mechanism). Before being completed, a workpiece can stay for a single (Dolgui *et al*., 2008; Battaïa *et al*., 2012) or multi-turn circular transfer (Battini *et al*., 2007).

- *Multiple lines*. Using multiple lines can be considered when the production system involves multiple products, in which each line can be designed for one family of similar products. The survey presented in Lusa (2008) lists the possible configurations of multiple lines.

According to task attributes

Besides the processing time of a task, a number of other attributes may be relevant when assigning them to workstations, such as process and ergonomic aspects, probability of failure, cost, etc. Such attributes may have constant/uncertain/dynamic/dependent values, as described next.

- *Constant*: all task attributes are fixed and known (Amen, 2006).

- *Uncertain*. Uncertain tasks attributes are not known exactly at the point when line balancing decisions have to be made. For example, in manual lines, the effectiveness of workers may vary with the work rate, skill level and motivation, which may affect the processing times of tasks (Xu and Xiao, 2011).

- *Dynamic*. Task attributes, such as processing time or required resources, may vary over time and can be reduced in successive cycles due to improvements observed in the assembly line or learning effects observed for the operators (Digiesi *et al*., 2009).

- *Dependent*. Task attributes are not fixed but dependent, for example, on the skill of the operator (Corominas *et al*., 2008; Blum and Miralles, 2011), on the processing sequence (Capacho and Pastor, 2006), or on the type of workstation to which the task is assigned (Gao *et al*., 2009).

According to the workpieces flow

- *Paced lines*. In a paced line, also referred as synchronous line, all operators or workstations have a common limited span time to work on a workpiece. Therefore, the workpiece stops at every workstation, and is automatically transferred as soon as a given time span is elapsed (Lapierre and Ruiz, 2004; Salehi *et al*., 2013).

- *Unpaced lines*. In an unpaced line there is no maximum limit imposed on the processing time available to the operator or the workstation. In *unpaced asynchronous lines*, the

movement of the workpieces is not coordinated, and the workpieces are transferred whenever the required tasks are completed, as long as the successive workstation is available (Sabuncuoglu *et al*., 2006). Buffers are installed in-between workstations to store the workpieces that cannot be advanced when the successive workstation is blocked by another workpiece. In *unpaced synchronous lines* all workstations wait to the lowest workstation before the workpieces are transferred (Karabati and Sayin, 2003). In contrast to unpaced asynchronous lines, buffers between workstations are not required.

- *Bucket brigades*. A bucket brigade is essentially a human chain used to transport and complete a product through an assembly line. The product is processed and passed from one worker to the next. When the last worker finishes his product he walks back upstream to take over the work of the next-to-last worker, who in his turn also walks back and so on, until the first worker is reached, who then walks back to the start of the line and begins a new product (Bartholdi and Eisenstein, 1996; Bratcu and Dolgui, 2005).

According to the level of automation

- *Manual lines*. In manual lines the tasks mainly or completely rely on manual labour. They are especially common where tasks are difficult and complex to automate (Finnsgård and Wänström, 2013).

- *Robotic or automated lines*. Robotic lines are fully automated lines and are mainly implemented whenever the work environment is somehow hostile for human beings, as for instance in the paint shops in the automotive industry, or when robots are able to perform the tasks more economically and with a higher precision (Aghajani *et al*., 2014).

2.1.2 Problem constraints

As has been mentioned before, many different features of the line balancing problem are vital in real-life environments other than the precedence and cycle time constraints. Among these features, the following ones are particularly relevant for the problem studied in this thesis:

- *Accessibility constraints and task assignment contraints*. In practice, there are usually constraints related to the positioning of the workpiece on the workstation which restrict the access of the workstations over the workpieces. For every position there corresponds a set of tasks which can be executed (Essafi *et al*., 2010). An example of accessibility constraints is the case where the workpieces are larger than the workstation width (Müller-Hannemann and Weihe, 2006). Another case arises if workpieces need to undergo position changes when they are processed, and a task can only be processed if it is situated in the required position for performing the task (Lapierre and Ruiz, 2004; Essafi et al., 2010). If the workpieces are weighty, large and fixed at the conveyor belt and cannot be turned in any position, may also need to be processed at a certain workstation (Wang and Wilson, 1986). Another case for workstations restrictions might be if a task that need heavy machinery have to be processed there (Scholl *et al*., 2010).

- *Movement constraints*. Movement constraints refer to the transportation pattern of workpieces through the line. In contrast to common variants of line balancing problems, the forward steps may be variable and smaller than the distance between two workstations. Therefore, the movement of the workpieces through the line should be such that each task is reachable from the workstation where it will be executed at least in one stage of the cycle (Müller-Hannemann and Weihe, 2006). On the other hand, another type of assignment constraint is related to distance restrictions such as the minimum and maximum distances measured in time, space or workstation positions between the tasks (Buxey, 1974; Pastor and Corominas, 2000).

2.1.3 Solution procedures

Solution procedures for solving assembly line balancing problems are often divided into two categories: exact or approximate. Since even the simplest case of line balancing problems, SALBP, is *NP-hard*, the computational time for obtaining an optimal solution with exact methods may increase exponentially for most of line balancing problems as the size of the instance increases. Consequently, approximate methods are needed in order to cope with large challenging instances, and aiming at obtaining good feasible solutions in an acceptable computation time. Additionally, simulation can be helpful to analyze the dynamic behavior of the line (Adham *et al*., 2013; Junsong, 2014).

Exact methods

Generally, line balancing problems can be solved optimally via one of the two following approaches: using a *standard solver* (like IBM ILOG CPLEX, Gurobi, COIN-OR, etc), or an *original dedicated solution method*. In the former case, the goal is to define an appropriate mathematical programming model and to adjust the solver parameters in order to solve it as quickly as possible. The mathematical models presented in the literature to describe line balancing problems mostly include mixed integer linear programming models (Miralles *et al*., 2007; Corominas *et al*., 2008; Pastor, 2011; Delorme *et al*., 2012). Other mathematical models used include integer linear programming (Bowman, 1960), nonlinear integer programming (Hamta *et al*., 2011), goal and fuzzy goal programs (Özcan and Toklu, 2009), and constraint satisfaction programs (Topalogu *et al*., 2012).

Since solvers are designed to deal with a large class of optimization problems, they might not be efficient enough for certain types of line balancing problems or even for a particular structure of input data. In this case, original dedicated methods can be developed, such as dynamic programming (Gungor and Gupta, 2001; Dolgui *et al*., 2008) or branch and bound (Miralles *et al*., 2008; Hu *et al*. 2010; Borisovsky *et al*., 2012; Sewell and Jacobson, 2012).

Approximate methods

A great variety of approximate methods have been proposed in the literature to solve assembly line balancing problems (e.g. Talbot *et al*. 1986; Scholl and Voβ, 1996; Amen, 2001). These approximate methods can be roughly divided into three categories: *bounded exact methods*, *simple heuristics*, and *metaheuristics*.

- *Bounded exact methods* perform an incomplete enumeration of the solution space. They can be obtained by bounding existing exact methods either by restricting the explored solution space or by limiting the available computational time (Blum and Miralles, 2011; Bautista and Pereira, 2011).

- *Simple heuristics* are usually very specific and problem-dependent techniques. In most of the cases, priority rules are used to assign tasks. These rules are typically based on task attributes such as task time or number of followers (Capacho and Pastor 2006; Scholl and Becker, 2006; Pastor *et al*., 2012). The category of simple heuristic methods can be divided into two classes:

  - *Single-pass heuristics*. The tasks are assigned in a single iteration using a greedy function or a priority rule (Toksari *et al*., 2008). The solution of this assignment is the final output, which generally is obtained in a very short time even for large-scale problems.

- *Multi-pass heuristics*. Due to the randomness nature of these algorithms, different results can be obtained and the output can be defined as the best solution found after a number of iterations (Andrés *et al*., 2008). Randomness may also be used to select a task to be assigned: task can be selected from a list of candidates (Toksari *et al*., 2010), or among tasks having the greatest value of a greedy function (Guschinskaya *et al*., 2011), or according to a random priority rule (Gamberini *et al*., 2009). The stop criterion may be expressed with a specified number of iterations or a number of iterations without improving the best obtained solution and/or a resolution time limit.

Simple heuristics can be used to provide an upper bound for an exact method (Baldacci *et al*., 2004) or be integrated into metaheuristics for local improvements of intermediate solutions (Essafi *et al*. 2012).

❖ *Metaheuristics, hybrid metaheuristics and matheuristics*. Metaheuristics are general methodologies designed to solve a wide range of hard optimization problems without having to deeply adapt them to each problem. A survey on metaheuristics can be found in Boussaïd *et al*. (2013). While such methods are numerous and varied, they can be roughly divided into the following classes:

- *Neighborhood methods* such as tabu search (Glover and Laguna, 1997; Özcan and Toklu, 2009), GRASP (Chica *et al*., 2010), simulated annealing (Kirkpatrick *et al*., 1983; Jayaswal and Argawal, 2014), variable neighborhood search (Hansen and Mladenović, 1999), etc.

- *Evolutionary approaches,* such as differential evolution methods (Mozdgir *et al*., 2013), genetic algorithms (Kazemi *et al*., 2011), imperialist competitive algorithms (Bagher *et al*., 2011) or memetic algorithms (Gamberini *et al*., 2009).

- *Swarm intelligence based metaheuristics* such as particle swarm optimization algorithms (Nearchou, 2011), bees algorithm (Tapkan *et al*., 2011), or ant colony optimization (Bautista and Pereira, 2007).

In the recent years, the focus of research has experienced a noteworthy shift towards the *hybridization of metaheuristics* with other techniques of optimization. The main motivation behind the hybridization of metaheuristics is to exploit the complementary character of different optimization strategies. The works of Raidl (2006) and Blum *et al*. (2011) provide a literature review on hybrid metaheuristics in combinatorial optimization.

A new and promising trend is *matheuristics* (Maniezzo *et al*., 2009). These are based on the interaction of metaheuristics and exact methods. An essential feature is the exploitation in some part of the algorithms of features derived from the mathematical programming (MP) model of the problem considered (Boschetti *et al*., 2009), or the use of an approximate procedure to deal with the non-feasible or non-optimal solutions yielded by a MP algorithm. However, because of their novelty, there is not a consolidated classification of this field, and therefore it is difficult to find a unique definition of these methods. For a detailed insight into matheuristics the reader is referred to the work of Maniezzo *et al*. (2009).

2.2 Definition of the AWALBP

This thesis studies a class of general problem (GALBP) which we have entitled AWALBP: the *Accessibility Windows Assembly Line Balancing Problem*.

The main characteristic of such a problem is that it considers an assembly line with limited access of the workstations over the larger workpieces. The specificity of the studied problem consists in the necessity of taking into account:

- *Accessibility constraints* related to the position of the tasks, which may fall out of the accessible area of the workstations.

- *Task assignment constraints*, to ensure that each task can be performed at one compatible workstation and in one stage of the process where it is accessible from the workstation.

- *Movement constraints* to define the stepwise transportation pattern of the workpieces through the assembly line.

The AWALBP can be stated as follows. An assembly line must process a number of (potentially infinite) identical workpieces. Several workpieces are placed consecutively on the line and the distances between two consecutive workpieces are equal. The length of the workpieces is longer than the width of the workstations. This implies that a workstation cannot reach a whole workpiece, but only the portion of workpiece(s) that are inside its reachable area (accessibility window). As a result, each workpiece may be processed by several workstations at the same time, and each workstation may process either one workpiece or two consecutive workpieces at the same time.

Fig. 2 shows an example of an assembly line with accessibility windows. Observe that workstations 1 and 4 can access parts of one single workpiece, whereas workstations 2 and 3 can access parts of two consecutive workpieces simultaneously.

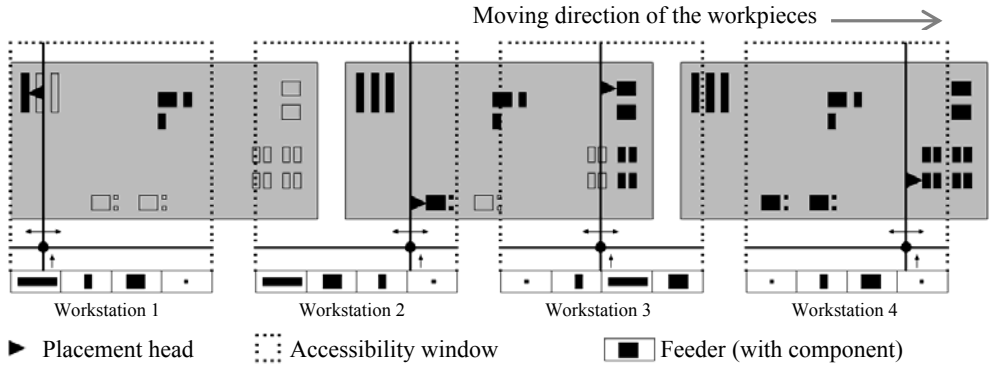

Figure 2: *An example of an assembly line with accessibility windows.*

The accessibility window of a workstation *i* is an interval $[L_i, R_i]$ of the assembly line (see Fig. 2) where the workstation can only access the limited portion of workpieces that are visible inside this interval. L_i and R_i are, respectively, the abscissae of the left and right limits of the reachable region of workstation *i*. Due to the restricted accessibility of workstations to workpieces, a task can only be executed if its position is accessible from its assigned workstation.

The assembly tasks are performed by operators or robots during *stationary stages*, in which the line is halted. In a task, a component is picked from the workstation and placed on a predefined

position of the workpiece. After all tasks have been executed in a specified stationary stage, the workpieces are moved forward by some common distance called *forward step*. The forward steps are cyclically repeated and there is a fixed number of forward steps per cycle. In this way, the transportation of the workpieces through the line is performed according to a pattern called *movement scheme*. A movement scheme comprises a number S of stationary stages separated between them by a forward step. Each of these forward steps defines a distance in the direction of the assembly line. The distance covered in a forward step s (s = 1, …, S) is denoted by $\Delta \cdot \delta_s$, where Δ is a length called elementary step, which depends on the technology of the line (Müller-Hannemann and Weihe, 2006), and δ_s is the number of elementary steps of the forward step s. The forward steps may cover the same distance (*fixed forward steps*) or different (*variable forward steps*). After S forward steps, the workpieces have been moved forward by a distance A, which is the distance between the left borders of two consecutive workpieces. Due to this cyclic behavior, a workpiece, which is at a certain position on the line, will be at the same position as the preceding workpiece once the whole movement scheme has been executed. Thus, the movement scheme defines the exact position of each workpiece in the stationary stages, which is given by:

- The initial position of the first workpiece, x, on the line in the first stationary stage.
- The number S of stationary stages (or equivalently, the number S of forward steps).
- The lengths $\Delta \cdot \delta_s$ of the forward steps, where δ_s is the number of elementary steps of the forward step s $(s = 1,...,S)$.

A movement scheme is feasible if, for each task, there exists at least one stationary stage such that the task is accessible from a workstation where it can be performed.

Fig. 3 depicts an example of a movement scheme with three stationary stages (S = 3). Each line is a snapshot representing the positions of the workpieces at each stationary stage (the fourth stationary stage is identical to the first one). The arrows on each snapshot indicate the moves of the workpieces from the previous stationary stage to the current one. At the start of the cycle, a new workpiece enters the line, and after the cycle (i.e., after the third stationary stage), a fully assembled workpiece leaves the line. Note that, in this example, the lengths of the forward steps are different ($\Delta \cdot \delta_1, \Delta \cdot \delta_2, \Delta \cdot \delta_3$). Note also that after the third forward step the workpieces have been conveyed exactly through a distance A.

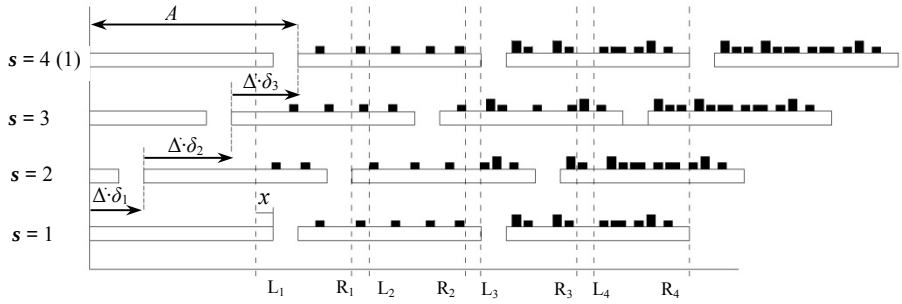

Figure 3: *Four snapshots of a cycle with three stationary stages.*

A number of additional characteristics may be considered in the AWALBP, including the ones described next:

- *Related vs. unrelated tasks*. The tasks may be related (or not) by precedence relationships, in such a way that a task can only be executed after its predecessor task has been completed.

- *A single vs. multiple compatible workstations for each task*. The tasks may be often classified into several types (for instance, a type of task may embrace the different tasks corresponding to place a same component in different locations). Each type of task may be potentially performed on a workstation belonging to a given set, depending on the characteristics of the workstations. If this set contains more than one workstation, a decision must be made to define the subset of workstations that will actually be able to perform the task (for example, deciding which components will be available at the feeders of each workstation). Finally, if this subset consists of multiple workstations, each task of the corresponding type must be assigned to one of the workstations of the subset. In all these cases, however, it is still necessary to determine the assignment of tasks to one of the stationary stages of the cycle.

- *A single vs. several robots per workstation*. The utilization of parallel robots in a same workstation may lead to a better balancing.

- *Dependent vs. independent task processing times*. The tasks may be workstation-dependent in the sense that the processing time for a task depends on the workstation to which it is assigned.

Finally, the objective function is to optimize a given efficiency objective (e.g., number of workstations, cycle time, cost or profit).

For more details on this type of line the reader is referred to previous work by Müller-Hannemann and Weihe (2006) and ***A MILP model for the Accessibility Windows Assembly Line Balancing Problem (Calleja et al., 2013)***.

2.2.1 Assumptions and classification

The main assumptions underlying the AWALB Problem are as follows:

- The accessibility windows do not overlap.
- All the forward movement steps must be multiple of a given elementary step Δ.
- All workpieces are identical.
- The workpieces have only two relevant dimensions (in the real-life variant considered by Müller-Hannemann and Weihe (2006) they are rectangular printed circuit boards).
- The distance between the left borders of two consecutive workpieces is constant, A.
- Each task has a predefined position on the workpiece and this position is defined by a single coordinate, since the workstations are limited by two values corresponding to the, say, horizontal axe of coordinates, but can access to any value corresponding to the vertical axe.
- The tasks must be executed without preemption.
- The processing time of the tasks at a given workstation is fixed (deterministic).

Finally, a summary of the main characteristics of AWALBP is presented in Table 1.

Table 1: *Main characteristics of the AWALBP.*

Distance of the forward steps	Fixed Variable	They are a multiple of a given elementary step Δ
Set of workstations able to perform a task	A single workstation Several workstations	
Precedencies	Exist Do not exist	
Objective function	Minimize	Number of workstations Cycle time Cost Combination of objectives
	Maximize	Efficiency Profit Combination of objectives

2.2.2 Optimization levels

In this section, a classification for the different optimization levels of the AWALBP is proposed. Such classification is based on the different *NP-hard* subproblems into which this problem can be divided, which gives rise to the following optimization levels:

L4. Line configuration. In this level the decision problem of determining the number and the type of the lines and workstations is addressed, along with the moving time and the acceleration/deceleration times of the lines, the available space for component feeders and toolbits.

L3. Machine configuration. This level entails the allocation of component types to feeders and the assignment of toolbit types to workstations. This determines which tasks can be performed at each workstation.

L2. Movement scheme. This level determines the movement pattern of the workpieces in a cycle. The initial position of the workpieces on the line, as well as the number and the lengths of the forwards steps have to be computed. This is required to determine in which pair of stationary stage and workstation a task can be performed.

L1. Task assignment. Here the assignment of each task to one compatible workstation and one of the stationary stages of the cycle has to be computed. In this way, an overall solution for the AWALBP is obtained.

The objective is the optimization of a specific throughput rate such as the cycle-time minimization.

Depending on which of the optimization levels are addressed, four variants for the problem are identified: AWALBP-L1, AWALBP-L2, AWALBP-L3 and AWALBP-L4. Each variant implies solving its own optimization level and its lower levels, assuming that the decisions of its superior levels have been made. For example, AWALBP-L1 refers to the problem of solving L1 when solutions of superior levels (L2, L3 and L4) are available. AWALBP-L2 concerns the simultaneous optimization of levels L1 and L2, assuming that the decisions of the superior

levels have been already taken. Likewise, the same reasoning is applied to define the variants AWALBP-L3 and AWALBP-L4.

2.3 The AWALBP-L2

As mentioned before, the AWALBP-L2 addresses the optimization of two problems: i) the movement scheme of the workpieces through the line and ii) the assignment of each task to one compatible workstation and one stationary stage of the cycle. At this optimization level, the decisions about the configurations of the workstations or the line have already been made.

In this doctoral thesis a case of AWALBP-L2 is addressed. The considered case corresponds to the real-life problem described in Müller-Hannemann and Weihe (2006), which arises in the automated assembly of large printed circuit boards on a line of modular pick-and-place machines. The case includes the following characteristics:

- For each task, a single workstation is compatible. Therefore, in the task assignment problem what has to be determined is the assignment of each task to one stationary stage of the cycle.
- The tasks are unrelated and workstation-independent, this is, there are not precedencies between the tasks and their processing times do not depend on the workstations.
- The distances covered in the forward steps may be different.
- Each workstation holds a single robot.

Hereafter, unless stated otherwise, references to AWALBP-L2 are to the specific case of Müller-Hannemann and Weihe (2006).

The problem can be stated as follows. A number of identical workpieces must be processed by an assembly line. The workpieces are launched equidistantly down the line. The distance between the left (right) borders of two consecutive workpieces is denoted by A. The assembly line is given by a number m of workstations. On each workstation i $(i = 1,...,N)$ a specified set of tasks J_i must be executed for each workpiece. The overall number of tasks is denoted by $N = \left| \bigcup_{i=1}^{m} J_i \right|$. Tasks have to be processed without preemption. Each task has a position on the workpiece. A task can only be processed if its position falls inside the accessibility window of its workstation. The accessibility window of a workstation is an interval $[L_i, R_i]$ of the assembly line, such that $L_1 = 0$, $R_1 > 0$, and $R_{i-1} < L_i < R_i$ for $i = 2,...,m$. For each task j $(j = 1,...,N)$ the triple (p_j, a_j, m_j) is known, where p_j is the processing time of task j, a_j is the horizontal distance from the task position to the right border of the workpiece, and m_j is the the workstation which has to perform this task. The assembly process decomposes into stationary stages, in which the line is halted. After a stationary stage is finished, the workpieces are simultaneously moved forward by some equal distance, and the next stationary stage begins.

The solution of the problem consists of:

i) a movement scheme $< x : \delta_1, \delta_2, ..., \delta_S >$, which includes:
   - The initial shift x of the workpieces of the line. It is defined by the distance of the right border of the first workpiece on the line with the respect to the left limit of the first

workstation at the beginning of the cycle, $0 \le x \le \min\left(R_1 + a^1_{\min}, A - \Delta\right)$, where $a^1_{\min} = \min_{j \in J_1} a_j$ corresponds to the task position that is closest to the right border of the workpiece (for the set of tasks that could be executed on workstation 1, J_1).

- The number S of stationary stages (which it is also the number S of forward steps).
- The sequence $\Delta \cdot \delta_1, ..., \Delta \cdot \delta_S$ of the lengths of the forward steps, where δ_s is the number of elementary steps of the forward step s $(s = 1, ..., S)$.

ii) the assignment of each task to one stationary stage of the cycle.

For a solution to be feasible, the three following conditions are required:

a) First, the sum of the lengths of all forward steps in a cycle must be equal to the distance A, i.e., $\Delta \cdot \sum_{s=1}^{S} \delta_s = A$.

b) All forward steps must be a multiple of the elementary step Δ.

c) Each task must be assigned to a stationary stage in which the task is accessible from its assigned workstation.

The optimization objective (1) is the minimization of the cycle time (*CT*). Between two stationary stages, there is a time T to take into account the acceleration and deceleration of the line, as well as the resetting of the robot arms. Thus the cycle time is equal to the sum of i) the time T multiplied by the number of stationary stages S plus ii) the time elapsed in the stationary stages constituting a cycle and iii) the time for transporting a workpiece at steady speed. Since the latter is a constant it is not considered for optimization purposes:

$$CT = T \cdot S + \sum_{s=1}^{S} C_s \tag{1}$$

where C_s is the completion time, for the whole line, corresponding to the stationary stage s $(s = 1, ..., S)$.

As it is generally assumed in assembly line balancing problems, without loss of generality we consider that all data are integers and thus the objective function value is also integer.

Chapter 3

State of the art

The optimization of printed circuit board (PCB) assembly problems is well studied (see for example the survey of Crama *et al*. (2002)). In the articles of Ammons *et al*. (1997), Johnson and Smed (2001) and Crama *et al*. (2002) a general classification of PCB assembly line problems is given and a hierarchical solving approach is proposed. However, the solving strategies proposed in such works cannot be directly used here since the regarded line type differs too much technically from the problem of this thesis, or it is assumed that each workstation can access all placement locations.

To the best of our knowledge the studied line types in the literature of line balancing do not fit the one presented in this thesis. Usually, in the literature it is assumed that each workpiece is transported from one workstation to the next and that each workstation has full access to all the tasks of the workpiece. This implies that, contrarily to our problem, the accessibility area of each workstation is at least as large as the length of a workpiece.

There are relatively few related works considering a cyclic movement scheme with accessibility windows. Several publications have arisen in the scope of industrial cooperation projects with Assembléon, a global supplier of surface mount technology solutions for the electronics manufacturing industry. In the following these works are outlined.

Martin (2002) presents a constraint programming (CP) model for the automated assembly of printed circuit boards on specific pick-and-place machines with accessibility windows and cyclic step-wise transport of the workpieces. However, the emphasis of such work is given to the transformation of the problem formulation into an understandable model in OPL language, and computational results are not provided.

Gaudlitz (2004) proposes several techniques to solve each subproblem of the AWALBP-L3, namely algorithms, integer linear programming (ILP) and CP models, and further proposes an overall solving approach. The proposed ILP formulations are used to model the task assignment (L1) and the component type allocation subproblems (L3) individually, but no mathematical work addressing the simultaneous optimization of L1 and L2 (i.e., an AWALBP-L2) is reported.

Müller-Hannemann and Weihe (2006) consider a real-world application with a cyclic movement scheme and accessibility windows in which for each task only one workstation is compatible. The authors describe the characteristics of the problem and define the feasibility conditions that a solution must satisfy. They then present a heuristic algorithm which, for a given movement scheme, assigns each task to exactly one stationary stage (i.e, an AWALBP-L1 is addressed). This approach reportedly provides near optimal results under the assumptions that i) the

processing times of the tasks do not differ by orders of magnitude from each other, and ii) the total number of tasks is orders of magnitude larger than the number of forward steps.

Tazari (2006) addresses an AWALBP-L1 where, in contrast to the scenario regarded by Müller-Hannemann and Weihe (2006), for each task a subset of the workstations is compatible (instead of just exactly one workstation). A two-stage algorithm is proposed. In the first stage, a branch-and-bound method is used to compute an optimal solution for the special case of unit task-lengths and zero reset time. This solution is used in the second stage as the starting solution of a local search with the aim to obtain an improved solution. An extensive computational study with real-world cases available to the author is conducted to compare the obtained solution with an ILP-based approach using CPLEX. The proposed algorithm reportedly comes close to or even hits the lower bound computed by CPLEX although it is faster than CPLEX by orders of magnitude.

Stille (2008) considers a specific application of assembly lines for PCB manufacturing of high-mix low-volume batches of production. The assembly line consists of multiple feeders that hold the component types. One or more of these feeders can be exchanged during the processing of a batch. In particular, it must be decided which component types to assign statically, and which ones on the exchangeable feeders. The problem is regarded as a generalized bin packing problem with additional constraints, for which an algorithm is proposed.

Van Duijnhoven (2013) deals with a robotic assembly line with accessibility windows and toolbit exchanges. A robot can only pick a component up if a compatible toolbit is mounted to it. The actual toolbit can be exchanged during the assembly process. This operation, however, takes a relatively long time, hence a task assignment solution with the minimum production time is desired. An algorithm that makes use of simulated annealing is proposed to allocate the toolbits to the robots, which enables the line to reduce its production time.

All of the last six authors study the same type of assembly line with accessibility windows as the one addressed in this thesis. However, the proposed solving strategies cannot be directly used since they address other optimization levels different from AWALBP-L2.

Based on the problem described by Müller-Hannemann and Weihe (2006), Corominas and Pastor (2009) propose a mixed integer linear programming (MILP) model for the particular case of AWALBP-L2 in which for each task there is only one compatible workstation, but computational experiments regarding the performance of the model are not developed.

With the exception of the mathematical formalization of Corominas and Pastor (2009), the following conclusion can be drawn from the literature review: the solution to the problem that considers the simultaneous optimization of the movement scheme (L1) and task assignment subproblems (L2) (i.e., the AWALBP-L2) has not been addressed before.

Consequently, in this doctoral thesis the AWALBP-L2 is defined and addressed via exact and approximate approaches. On the one hand, two mathematical programming models are presented in order to find the optimal solution. On the other hand, heuristics and metaheuristic methods hybridized with mathematical programming are proposed in order to solve the challenging instances that are out of reach of the former models.

Chapter 4

Solution methods

4.1 Introduction

A number of solution methods have been designed, implemented and evaluated in this thesis to solve the AWALBP-L2. These methods can be roughly divided into four main approaches, as described next.

1. *Mathematical programming models*. In the first place mathematical integer linear programming (MILP) models have been developed, with the aim to observe if the problem can be solved optimally in a practical time limit (one hour). Specifically, two mathematical formulations (denoted F1 and F2), have been proposed (section 4.2). An extensive computational experiment using the commercial solver IBM ILOG CPLEX was carried out to test the performance of the proposed models, but only small to medium-size instances could be solved optimally. This is not surprising since, as previously discussed, the AWALBP-L2 is by nature *NP-hard*.

2. *MILP bounding procedure*. The second method involves the incorporation of upper and lower bounds to a mathematical programming formulation of the problem (F2), with the aim to improve the solutions yielded by the MILP models of the first approach. For this purpose, a combined approach has been proposed, which consists of i) a matheuristic, denoted *Initial solution matheuristic*, which is used to obtain good feasible solutions and to compute bounds and ii) a MILP model, denoted *Solve model*, that incorporates the obtained bounds. The matheuristic is composed by a heuristic (*Move algorithm*) and an ILP model (*Task model*). The *Solve model* is an extension of the aforementioned MILP formulation F2, to which new constraints have been added in order to include bounds on the cycle time and the number of stationary stages. The computational experiment was carried out with the same one hour limit, and results showed a significant improvement in the percentage of instances solved optimally.

3. *Hybrid metaheuristics*. In order to solve the challenging instances that could not be solved by the two previous approaches, hybrid metaheuristics that integrate mathematical programming models and metaheuristic frameworks are developed. More specifically, three different hybrids metaheuristics are proposed – one based on simulated annealing (SA) and the other two based on tabu search (TS), relying on different neighborhood definitions. The two first approaches rely on a classical neighborhood definition, obtained by subjecting the candidate solution to small changes or moves. The latter approach (TS-CM), in contrast, draws ideas from of the corridor method (CM) regarding the way to generate and explore the neighborhood (Sniedovich and Voβ, 2006). In the proposed TS-CM, exogenous

constraints are designed and imposed on the original formulation of the problem, and, subsequently, the constrained problem (denoted *Solve-corridor model*) is solved using a MILP solver. The procedure iteratively builds new corridors around the solution found in each corridor and, therefore, explores adjacent portions of the search space.

4. *Combinations of hybrid metaheuristics and MILP*. Finally, to further improve the quality of the solution of the problem, the fourth approach considers combining the use of the *Solve model* of approach 2 and the metaheuristics of approach 3. Following the same idea as in the second approach, a matheuristic is first used to generate an initial solution, and next an improving solution is sought by either launching first the *Solve model* and next a hybrid metaheuristics (combination 4a) , or the other way round (combination 4b).

With the exception of the first one, all of the proposed methods involve the combination of mathematical programming (MP) models and heuristics or metaheuristics. A general classification of existing methods combining exact and metaheuristic algorithms has been proposed by Puchinger and Raidl (2005). Fig. 4 gives an overview of this classification. The following two main categories are distinguished:

– *Collaborative Combinations*, where the algorithms exchange information, but are not part of each other. Exact and heuristic algorithms may be executed sequentially, intertwined or in parallel.

– *Integrative Combinations*, where one technique is a subordinate embedded component of another technique. Thus, there is a master algorithm, which can be either an exact or a metaheuristic algorithm, and at least one integrated slave.

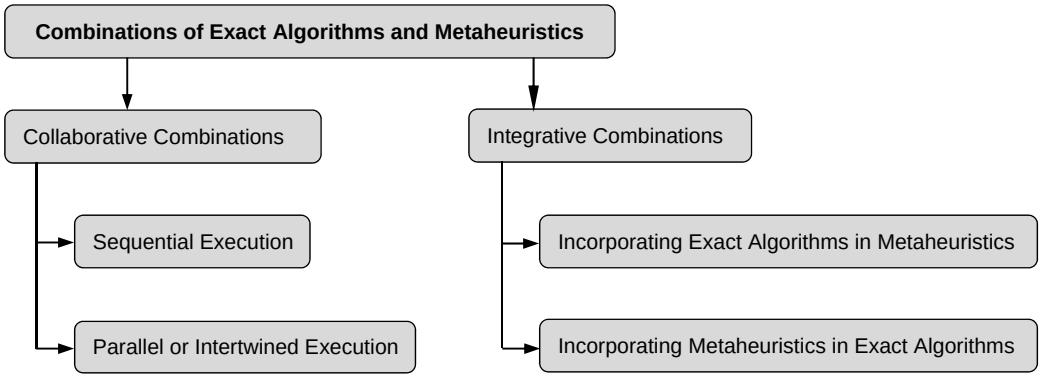

Figure 4: *Classification of exact/metaheuristic combinations.*

Fig. 5 depicts a summary of the proposed solution methods. The first approach, labelled with (1), corresponds to exact methods entailing the execution of the MILP models F1 and F2 with a one hour time limit. Approaches (2) and (4b) can be seen as bounded exact methods which not only limit the available computational time but also the explored solution space. Methods (3) and (4a) are approximate and rely on metaheuristics hybridized with a MILP model. Owing to the fact that, with the exception of (1) all of the proposed methods comprise a mathematical programming model and a heuristic or a metaheuristic algorithm, they can be regarded as matheuristics. According to the classification of Puchinger and Raidl (2005) (Fig. 4), approach (2) can be seen as a collaborative sequential combination between a matheuristic and a MILP model, in which the matheuristic is executed as a part of a preprocess before the MILP model.

candidate solution to small changes or moves. The latter approach (TS-CM), in contrast, draws ideas from of the corridor method (CM) regarding the way to generate and explore the neighborhood (Sniedovich and Voβ, 2006). In the proposed TS-CM, exogenous constraints are designed and imposed on the original formulation of the problem, and, subsequently, the constrained problem (denoted *Solve-corridor model*) is solved using a MILP solver. The procedure iteratively builds new corridors around the solution found in each corridor and, therefore, explores adjacent portions of the search space.

4. *Combinations of hybrid metaheuristics and MILP*. Finally, to further improve the quality of the solution of the problem, the fourth approach considers combining the use of the *Solve model* of approach 2 and the metaheuristics of approach 3. Following the same idea as in the second approach, a matheuristic is first used to generate an initial solution, and next an improving solution is sought by either launching first the *Solve model* and next a hybrid metaheuristics (combination 4a) , or the other way round (combination 4b).

With the exception of the first one, all of the proposed methods involve the combination of mathematical programming (MP) models and heuristics or metaheuristics. A general classification of existing methods combining exact and metaheuristic algorithms has been proposed by Puchinger and Raidl (2005). Fig. 4 gives an overview of this classification. The following two main categories are distinguished:

- *Collaborative Combinations*, where the algorithms exchange information, but are not part of each other. Exact and heuristic algorithms may be executed sequentially, intertwined or in parallel.

- *Integrative Combinations*, where one technique is a subordinate embedded component of another technique. Thus, there is a master algorithm, which can be either an exact or a metaheuristic algorithm, and at least one integrated slave.

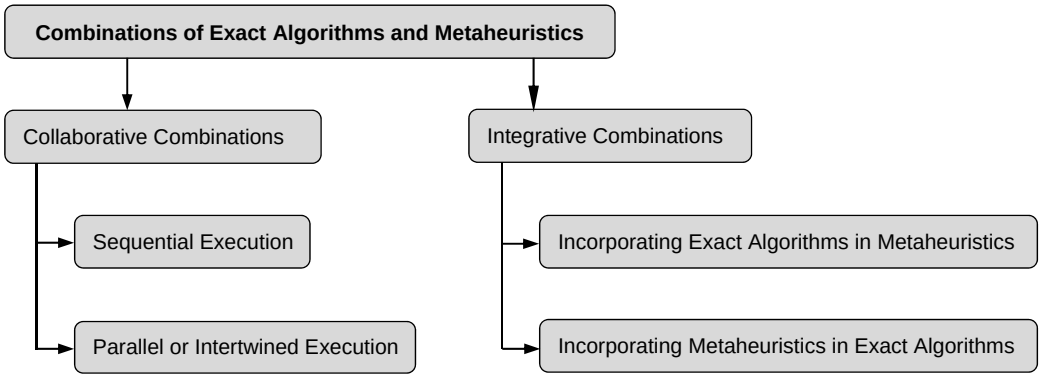

Figure 4: *Classification of exact/metaheuristic combinations.*

Fig. 5 depicts a summary of the proposed solution methods. The first approach, labelled with (1), corresponds to exact methods entailing the execution of the MILP models F1 and F2 with a one hour time limit. Approaches (2) and (4b) can be seen as bounded exact methods which not only limit the available computational time but also the explored solution space. Methods (3) and (4a) are approximate and rely on metaheuristics hybridized with a MILP model. Owing to the fact that, with the exception of (1) all of the proposed methods comprise a mathematical programming model and a heuristic or a metaheuristic algorithm, they can be regarded as matheuristics. According to the classification of Puchinger and Raidl (2005) (Fig. 4), approach (2) can be seen as a collaborative sequential combination between a matheuristic and a MILP model, in which the matheuristic is executed as a part of a preprocess before the MILP model.

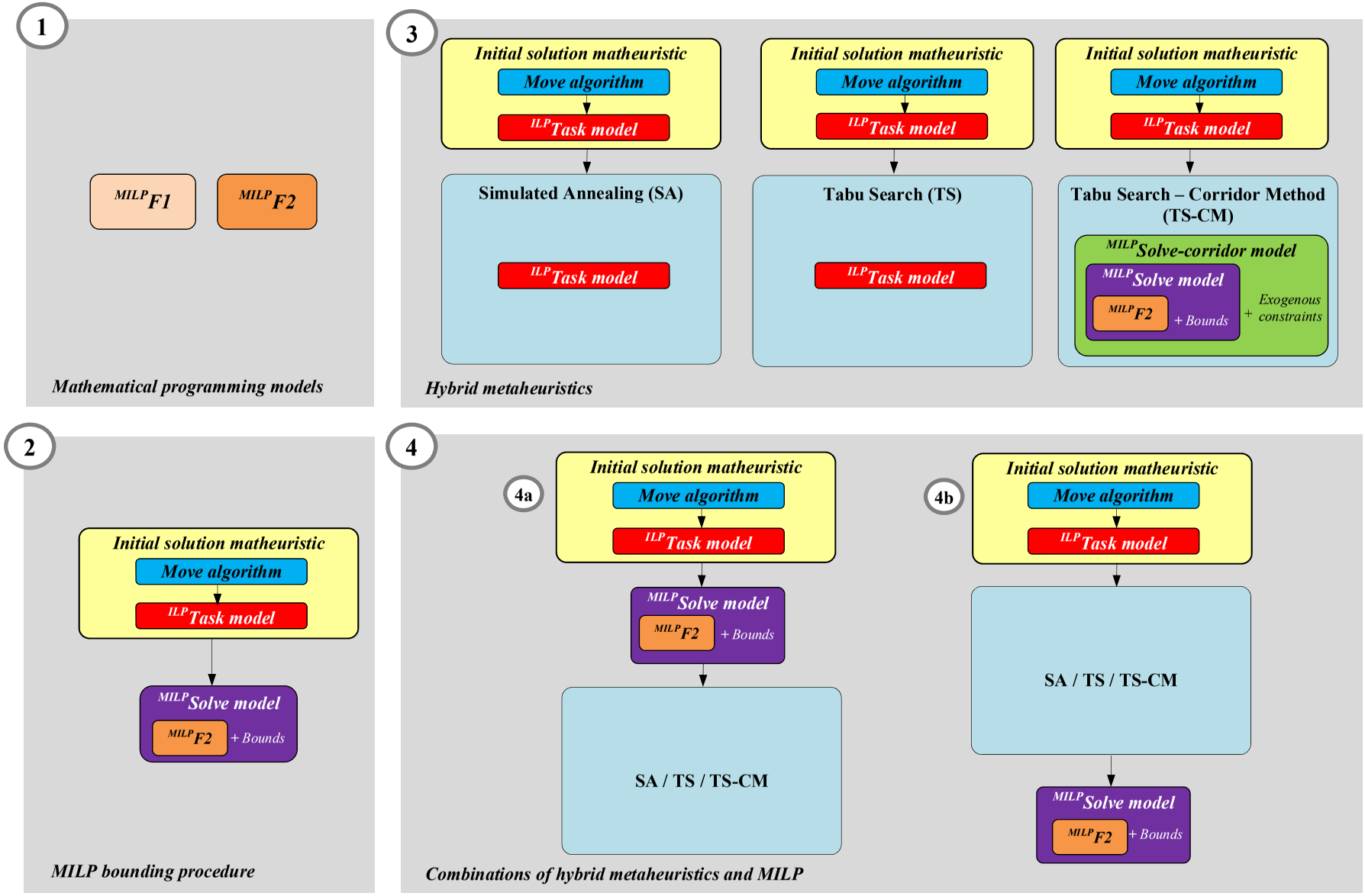

Figure 5: *Overview of the optimization solution methods proposed in this thesis.*

Next, the three metaheuristics developed in (3) were created as integrative combinations incorporating exact algorithms in a SA, TS or a TS-CM metaheuristic framework. Finally, the approach proposed in (4) involves a collaborative sequential combination between one of the three aforementioned metaheuristics and a MILP model, where the model is either applied before or after the metaheuristic.

All the proposed methods have been evaluated and compared via computational experiments. The description of these computational experiments and the analysis of the results are reported further in Chapter 5.

4.2 Mathematical programming models

In order to formalize and solve the AWALBP-L2 optimally, two mixed integer linear mathematical programming (MILP) models have been proposed, which are referred hereafter as formulations F1 and F2. Each of these models simultaneously solves the *task assignment* and *movement scheme* subproblems involved in the AWALBP-L2.

The first formulation, F1, is essentially based on the model of Corominas and Pastor (2009). The modifications made to this model include the addition of new constraints and the refinement of variable domains.

An alternate MILP formulation, F2, has been developed. The model features significant changes with respect to F1. The main transformations refer to the suppression of two sets of integer variables, which have been replaced by a new set of binary variables instead. As a result, the constraints where these variables appear have also been modified accordingly.

The performance of the proposed models was tested using IBM ILOG CPLEX with a one hour of computational limit.

The complete proposed mathematical formulations and the result of a computational experiment are reported in ***A MILP model for the Accessibility Windows Assembly Line Balancing Problem (AWALBP) (Calleja et al., 2013).***

4.3 MILP bounding procedure

For the AWALBP-L2 the size of the instances that were practically solvable with the proposed MILP models is rather limited. Therefore, a solution approach that aims at reducing the solution space has been proposed. The proposed approach consists of:

i) A matheuristic, denoted *Initial solution matheuristic*, to generate good feasible solutions and compute upper and lower bounds on the cycle time and the number of stationary stages. The matheuristic is executed for each possible value of the initial position x and basically consists of:

   - An algorithm, denoted *Move algorithm*, which computes, for a given value of the initial position x, a feasible movement scheme with the minimum number of stationary stages.

   - An ILP model, denoted *Task model*. It computes the optimal task assignment for the obtained movement scheme.

ii) A MILP model, called *Solve model*, which incorporates the computed bounds.

This approach relies on a key property featured by the proposed *Move algorithm*. More specifically, the algorithm has the property of providing a feasible movement scheme with the minimum number of stationary steps for a given initial position x. This is a very important characteristic since it is useful for the computation of bounds: it indirectly gives a lower bound on the number of stationary stages and it allows deriving a lower bound on the cycle time.

The operation of the proposed MILP bounding procedure follows four steps:

Step 1. Initial feasible solutions are computed by using the *Initial solution matheuristic*, which is launched for all possible values (multiples of Δ) of the initial distance x. As a result, this procedure provides as many initial feasible solutions as values of x.

Step 2. Among all the solutions obtained in the previous step, the solution with the minimum cycle time, Sol^{CT}, and the solution with the minimum number of stationary stages are identified, which are used to derive upper and lower bounds on the cycle time and the number of stationary stages. The bounds are added to the *Solve model* in order to restrict the solution space to those solutions with an objective function value strictly lower than the known solution Sol^{CT}.

Step 3. The optimality of the feasible solution Sol^{CT} (identified in *Step 2*) is checked by comparing its objective function value with the lower bound on the cycle time. If these values coincide, this certifies that such solution is optimal. Otherwise, the *Solve model*, which incorporates all bounds derived in *Step 2*, is launched.

Step 4. The *Solve model* is launched with a one-hour computational time limit with the aim to obtain a final solution for the problem. Basically, the *Solve model* is based on the MILP formulation F2 whose dimension is reduced thanks to both the addition of the bound constraints and the usage of a lower number of variables and constraints.

With the designed method either a feasible or an optimal solution is always obtained, as described next.

An optimal solution is certified if one of the three following cases holds:

(i) The values of the objective function and the lower bound on the cycle time coincide, which certifies that the initial feasible solution Sol^{CT} is optimal.

(ii) The *Solve model* is proved unfeasible, which means that there is no solution whose objective function value is lower than that of Sol^{CT} and thus this solution is optimal.

(iii) The *Solve model* yields an optimal solution (which improves the initial solution Sol^{CT}).

A feasible solution is obtained, but not proven optimal, if one of the two following cases holds:

(i) The *Solve model* finds a solution which is better than Sol^{CT}, but its optimality is not certified.

(ii) The *Solve model* does not find a feasible solution in the allowed time limit. In this case, Sol^{CT} is kept as the result of the overall method.

The design of the combined method and the experimental results are detailed in ***Combining matheuristics and MILP to solve the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2) (Calleja et al., 2014a)***. Additionally, the article provides a proof to demonstrate that the proposed *Move algorithm* gives a movement scheme with the

minimum number of stationary stages. Furthermore, it proves that the assignment of tasks to stationary stages in the AWALBP-L1 is a *NP-hard* problem.

4.4 Hybrid metaheuristics

The main motivation behind the hybridization of metaheuristics with other techniques is to exploit the complementary characteristics of the different optimization techniques considered. Initially, pure metaheuristics had a considerable success since they proved to be one of the most practical approaches for many problems. However, after years of optimization expertise it became clear that pure metaheuristics had reached their limits, and hence the current interest in their hybridization (Blum *et al*., 2011).

The hybrid metaheuristics proposed in this thesis hybridize a metaheuristic, which is the master mechanism that guides the search, and one mathematical model that acts as an embedded slave.

Such hybrids use two different types of neighborhood definition in the search process:

Type a. The first neighborhood type is defined by the application of small changes or moves to a current movement scheme. Owing to the fact that the optimal assignment of tasks to stationary stages, for a given movement scheme, can be obtained fast with a MP model (the *Task model* introduced in chapter 4.3) the search is focused in the space of the movement schemes, and not in the space of complete solutions. For this reason we have designed a neighborhood that operates the search in the space of the movement schemes. More specifically, the following neighborhoods were used: i) transference of one elementary step from a forward step to another forward step (N_1), ii) insertion of a new forward step by transferring one elementary step from an existing forward step to a new forward step, (N_2), and iii) modification of the value of the initial position x (N_3).

Type b. The second neighborhood type, in contrast, draws ideas from the Corridor Method (CM) proposed by Sniedovich and Voβ (2006). The central idea of the CM is to define constraints on the target problem, such that efficient exact methods can be designed to solve the neighborhood search problem efficiently. We define the neighborhoods by iteratively building a *corridor* around a current movement scheme via the imposition of exogenous constraints on a MILP model. The aim is to identify smaller portions of the solution space which are amenable to solving with a MILP solver. Such portions, or *corridors*, can be defined by constraining the domains of the variables that are present in a current solution. Consequently, A MILP model, denoted *Solve-corridor model*, has been developed to be used at each iteration of the search. Three types of corridor structures have been proposed, *C1*, *C2*, and *C3*, which construct a neighborhood around specified variables of a current movement scheme, as follows:

- *Corridor C1* constructs a neighborhood around the variables δ_S of a current movement scheme, by including all movement schemes whose forward steps have a length within a specified distance from the current lengths.

- *Corridor C2* builds a neighborhood around δ_S and around the number of forward steps S.

- *Corridor C3* constructs a neighborhood around δ_S, the number of forward steps S, and the initial position x.

Three hybrid metaheuristics have been designed: one based in simulated annealing (denoted SA) (Kirkpatrick *et al*., 1983) and the other two based on tabu search (denoted TS and TS-CM, respectively) (Glover, 1986) which use different neighborhood definitions, as follows:

(i) SA: A hybrid metaheuristic based on simulated annealing with a move-based neighborhood definition (type a) (the proposed SA approach is described in ***Balancing assembly lines with accessibility windows. Problem description and heuristic solving procedure (Calleja et al., 2014b).***

(ii) TS: A hybrid metaheuristic based on tabu search with a move-based neighborhood definition (type a).

(iii) TS-CM: A hybrid metaheuristic based on tabu search with a CM-based neighborhood definition (type b).

The proposed hybrid metaheuristics SA and TS use the *Task model* as a subordinate embedded slave. At each iteration, the *Task model* is applied to compute the optimal cycle time for the current neighbor movement scheme, which provides a complete current solution for the problem. The obtained cycle time value determines whether the candidate movement scheme (along with its optimal task assignment) will be accepted or rejected as the new current solution in the local search of the SA or the TS metaheuristic.

On the other hand, the TS-CM hybrid metaheuristic uses the *Solve-corridor model* as an embedded slave. The proposed hybrid follows the general scheme of a TS metaheuristic with the difference that the neighborhoods are not defined via local changes or moves, but constructed by adding exogenous constraints onto the embedded *Solve-corridor model*. At each iteration, the *Solve-corridor model* receives a current solution as input. Based on this solution, bounds on the cycle time and the number of stationary stages are derived and incorporated to the model. Exogenous constraints are also imposed in order to construct the corridors and to define two tabu lists and the aspiration criterion. Subsequently, the model is used to solve the resulting reduced portion of the solution space.

The research, proposals and experimental results are reported in ***Hybrid metaheuristics for the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2)***.

4.5 Combinations of hybrid metaheuristics and MILP

With the aim to further improve the quality of the solution of the AWALBP-L2, sequential combinations of the afore-presented hybrid metaheuristics and the *Solve model* have been proposed, as explained next. An initial solution is generated with the *Initial solution matheuristic*, and next an improving solution is searched by combining the use of the *Solve model* and the hybrid metaheuristic in two alternative ways: i) using the *Solve model* with the initial solution obtained with the *Initial solution matheuristic* and then trying to improve the solution obtained by the model using one of the proposed hybrids. Or ii) executing one hybrid and then using the obtained solution as the initial solution for the *Solve model*.

Table 2 depicts the proposed combinations of hybrid metaheuristics and MILP. The two rows show, respectively, the two combinations types considered. In the first row MILP is applied before the hybrid, whereas in the second row MILP is applied after. In each combination type, running time of the MILP *Solve model* is limited to 900, 1800 or 2700 s, whereas the *Initial solution matheuristic* and the hybrid are executed in the remaining run time with a one hour time limit.

Table 2: *Combinations of the proposed hybrid metaheuristics and MILP.*

Initial solution matheuristic	+	*Solve model* 900 s / 1800 s / 2700 s	+	SA-, TS-, or TS-CM- hybrid metaheuristic
		SA-, TS-, or TS-CM- hybrid metaheuristic	+	*Solve model* 900 s / 1800 s / 2700 s

Hereafter, the following notation is used: $MILP_{time}$+*Hybrid* denotes the combination type where the *Solve model* is executed before the hybrid, with $MILP_{time} \in \{900, 1800, 2700\}$ seconds and $Hybrid \in \{SA, TS, TS\text{-}CM\}$. Accordingly, *Hybrid*+$MILP_{time}$ denotes the combination where the model is executed after the hybrid.

The computational results of the combinations are presented in the article ***Hybrid metaheuristics for the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2)***.

Chapter 5

Computational results

5.1 Experimental conditions

To evaluate and compare the performance of the solution methods described in Chapter 4, an extensive computational experiment was carried out, for which small, medium and large-sized instances of AWALBP-L2 were considered. Since no benchmark set of data was available in the literature, a set of 1,200 realistic instances generated at random was used (this set can be found at https://www.ioc.upc.edu/EOLI/research/). The instances are essentially based on the description of the real-world test cases given by Gaudlitz (2004). As reported by this source, in this type of industrial applications the workpiece length may be up to 2.5 times larger than the width of the workstation, the number of workstations may typically range from 7 to 20 and the number of tasks may be between 100 and 800. Accordingly, an extended data set was generated, including workpieces with lengths up to four times larger than the width of the workstations, a number of workstations comprised between 5 and 40 and a number of tasks varying from 50 to 1,000.

Specifically, the following ranges of data are considered: six ranges of workpiece length A_0={11-15, 16-20, 21-25, 26-30, 31-35, 36-40}, four ranges of number of workstations m = {5-10, 11-20, 21-30, 31-40} and five ranges of number of tasks N={50-200, 201-400, 401-600, 601-800, 801-1000}. It was then obtained a total number of $6 \cdot 4 \cdot 5 = 120$ range combinations, by randomly selecting one value within each range. Subsequently, for each combination 10 instances were generated randomly, resulting in a final set of 1,200 instances.

Additionally, the instances have the following characteristics. The width of the accessibility windows is 10 length units (lu) and the length of the elementary step Δ is 1 lu. The time T is 200 time units (tu). The processing time of tasks were randomly generated between 100 and 150 tu, and were assigned to the workstations according to a equiprobable random policy. The positions of tasks were also randomly generated along the workpiece length A_0. The distance between two consecutive workpieces in the line is 1 lu and thus $A = A_0 + 1$.

The algorithms were coded and run in Java 7 and the mathematical programming models were tested using IBM CPLEX 12.2 in an Intel Core 3.33 GHz PC with 4 GB of RAM under Windows 7 (64 bits). The overall allowed computational time per instance was limited to one hour. The absolute optimality gap was set to $1\text{-}10^{-6}$ since without loss of generality, all data are integer and therefore the value of the objective function is also integer.

The parameter values of hybrids SA and TS were fine-tuned based using CALIBRA (Adenso-Díaz and Laguna, 2006), a systematic procedure that calibrates the parameter values of heuristic

or metaheuristic algorithms. Regarding the proposed TS-CM metaheuristic, a preliminary test was carried out to examine the performance of the three proposed corridors, *C1*, *C2* and *C3* (recall Chapter 4.4) and *C3* provided the best performance in terms of the improvement of the objective function with respect to the initial solution. This corridor was thus selected to be used in the computational experiment, with a computational time limit for a TS-CM iteration of 300 s.

The computational experiment implied the application of a total of 24 methods. Table 3 lists the names of the methods conducted and their corresponding approach according to the classification given in Chapter 4.1.

Table 3: *Proposed solution methods.*

Nº.	***Name***	***Approach***	***Description***
1	F1	1	Mathematical programming models
2	F2		
3	F3	2	MILP bounding procedure
4	SA	3	Hybrid metaheuristics
5	TS		
6	TS-CM		
7	$MILP_{900}$ + SA	4a	Combinations of hybrid metaheuristics and MILP
8	$MILP_{1800}$ + SA		
9	$MILP_{2700}$ + SA		
10	$MILP_{900}$ + TS		
11	$MILP_{1800}$ + TS		
12	$MILP_{2700}$ + TS		
13	$MILP_{900}$ + TS-CM		
14	$MILP_{1800}$ + TS-CM		
15	$MILP_{2700}$ + TS-CM		
16	SA + $MILP_{900}$	4b	
17	SA + $MILP_{1800}$		
18	SA + $MILP_{2700}$		
19	TS + $MILP_{900}$		
20	TS + $MILP_{1800}$		
21	TS + $MILP_{2700}$		
22	TS-CM + $MILP_{900}$		
23	TS-CM + $MILP_{1800}$		
24	TS-CM + $MILP_{2700}$		

5.2 Analysis of the results

In order to compare the different approaches proposed, for each method the following evaluation metrics are considered:

- *Percentage of optimal solutions found by the method itself* ($\%OS_{method}$): the percentage of instances for which optimality has been proven by the own method.
- *Number of optimal solutions (NOS):* the number of optimal solutions for which optimality has been proven by comparison with the known optimum.
- *Percentage of optimal solutions* (*%OS*): the percentage of instances for which optimality has been certified by comparison with the known optimum, with $\%OS = 100 \cdot (NOS / 1200)$.
- *Percentage of feasible solutions (%FS)*: the percentage of instances that are feasible but not proven optimal.
- *Percentage of unsolved instances (%UI)*: the percentage of instances for which a solution could not be found after one hour of computational time, thus $\%OS + \%FS + \%UI = 100$.
- *Maximal, average and minimal gap*, ($Gap_{\max}$, Gap_{av}, $Gap_{\min}$): the maximal, average and minimal gap with respect to the best bound, respectively. These gaps have been computed for the 1,200 instances of each method. For each instance, the relative gap is defined as $\left((BS - BB) / BS\right) \cdot 100$, where *BS* is the objective function value of the solution found by the method and BB is the best bound value known obtained among all proposed methods. Specifically, the value of *BB* is the maximal value among the following: i) the best bound computed by CPLEX among the MILP models of approaches 1 and 2, ii) the lower bound on the cycle time, LB^{CT}, and iii) the best bound computed by CPLEX among all combinations of hybrid metaheuristics and MILP of method 4b.

Overall results

Table 4 presents the performance results obtained by all methods defined in Table 3 for the set of 1,200 instances. From row 1 to 8, the results are given in terms of *NOS*, *%OS*, $\%OS_{method}$, *%FS*, *% UI*, $Gap_{\max}$, Gap_{av}, and $Gap_{\min}$, respectively. Since the methods of categories 4a and 4b produced very similar results (their %*OS* do not differ significantly) at this point only the method in each category with the maximal *%OS* is discussed, namely MILP_{2700} + SA and TS-CM + MILP_{900}. The detailed results for all methods belonging to approach 4 are given at the end of this section (see Tables 6 and 7). As it can be observed in Table 4 the MILP models were significantly outperformed by all other methods. The best performance was obtained with methods of category 4. Specifically, the best *%OS* corresponds to methods MILP_{2700} + SA and TS-CM + MILP_{900}, (81.08% and 80.92%, respectively). Among the hybrid metaheuristics, methods TS and TS-CM outperformed SA, achieving an optimal solution in 79.75% and 80.67% of the cases, compared to 73.08% in the case of SA. Additionally, the values of Gap_{max} and Gap_{av} are very similar for methods of approaches 3 and 4 (between 22.05% and 23.12% for Gap_{max} and between 1.13% and 1.85% for Gap_{av}). Another method that had a good performance is F3, which provided an overall optimal solution of 78.75%, having a Gap_{max} of 23.12% and a Gap_{av} of 1.59%. On the other hand, methods F1 and F2 performed the worst, generating an optimal solution in 41.00% and 55.58% of the cases. Furthermore, Gap_{max} is considerably high (above 80%), as well as Gap_{av} (55.36% and 20.15% for F1 and F2, respectively. Finally, F1 was the only method for which solutions could not be found after one hour of computational time (53.92% of the cases).

Table 4: *Performance evaluation of the proposed solution methods.*

Category	*1*		*2*	*3*			*4a*	*4b*
Description	*MILP models*		*MILP bounding procedure*	*Hybrid metaheuristics*			*Combining hybrid metaheuristics and MILP*	
Method	**F1**	**F2**	**F3**	**SA**	**TS**	**TS-CM**	**$MILP_{2700}$ + SA**	**TS-CM + $MILP_{900}$**
NOS	492	667	945	877	957	968	973	971
%OS	41.00	55.58	78.75	73.08	79.75	80.67	81.08	80.92
$\%OS_{method}$	39.50	53.33	77.50	57.42	57.17	57.25	78.58	78.08
%FS	5.16	44.50	21.25	26.92	20.25	19.33	18.92	19.08
%UI	53.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Gap_{max}	82.93	83.47	23.12	23.12	22.05	22.05	23.12	22.05
Gap_{av}	55.36	20.15	1.59	1.85	1.16	1.13	1.23	1.13
Gap_{min}	0.01	0.01	0.01	0.03	0.01	0.01	0.01	0.01

The results of Table 4 suggest that the application of bounding techniques and the integration of MILP models with metaheuristics allow better performance compared to the direct application of the proposed MILP models and MILP bounding procedure, both for producing a higher percentage of optimal solutions as for achieving a lower optimality gap.

In the following the results obtained for each approach are discussed in more detail.

MILP models

As mentioned before, the performance of F1 was significantly improved by the proposed reformulation F2. Specifically, %*OS* increased by 14.58%. Additionally, for each of the instances that could not be solved by F1 after one hour of computational time, F2 yielded either a feasible or an optimal solution. Furthermore, the average computational time decreased by 19.03%, from 2,407 to 1,949s. With this approach only small to medium-size instances could be solved optimally.

MILP bounding procedure

With respect to the previous MILP models, the *MILP bounding procedure* showed a significant improvement not only in the number of optimal solutions but also in terms of computational time. The *%OS* rose to 78.75%, and all instances that were solved optimally with methods F1 and F2 were also solved optimally with the proposed approach. Additionally, the average computational time decreased by 53.25%.

Table 5 details the average results for the *MILP bounding procedure*. A noteworthy result is that the proposed *Initial solution matheuristic* generated a high number of initial solutions (457 out of 1,200 instances, 38.08%) with an objective function value matching the lower bound on the cycle time (recall *Step 3* of the solving procedure described in Section 4.3), which shows that the proposed matheuristic provides good feasible initial solutions. In order to further test the quality of the matheuristic its obtained solutions are compared with the known optimal solutions of the problem. Specifically, the *Initial solution matheuristic* provided a *NOS*, *%OS* and Gap_{av} of 519, 43.25% and 4.03%. Finally, the computational times are only around 5 milliseconds on average.

Table 5: *Average results obtained for the MILP bounding procedure.*

Initial solution matheuristic	*Solve model*		*Total*	
$CT = LB^{CT}$ 38.08%	Unfeasible model	5.17%	**77.50%**	**Optimal solution**
	Optimal solution	34.25%		
	Improved feasible solution	19.42%	**22.50%**	**Feasible solution**
	No improved solution after 1h	3.08%		

As seen in Table 5, the *Solve model* resulted unfeasible for 5.17% of the instances. In this case, it is certified that a feasible solution with a lower value of the objective function does not exist and that therefore the initial solution is optimal. In addition, 34.25% of the instances could be improved and were certified as optimal by the model, yielding a total percentage of 77.50%. When compared to known optima, the overall *%OS* obtained for the method rises to 78.75% (recall Table 4). Furthermore, the model provided 19.42% of feasible solutions which, in all cases, have a better value of the objective function than the initial solution. Finally, 3.08% of the instances were not solved after 3,600 s of computational time. In this latter case, the initial feasible solution was kept as the result.

Regarding the average deviation Gap_{av} from the best bound *BB*, for the 1,200 solutions of the *MILP bounding procedure* Δ_{av} is 1.59% (recall Table 4). Additionally, in the 3.08% cases where the *Solve model* does not provide a solution after the allowed one-hour time limit, the average gap value is 14.53%.

Hybrid metaheuristics and their combinations with MILP

All methods of this approach use the *Initial solution matheuristic* as their first step. As mentioned before, with this matheuristic 457 out of 1,200 instances were certified as optimal. Therefore, unless stated otherwise, the results presented hereafter are based on the remaining 743 instances.

Table 6 shows the comparative results for the proposed methods with respect to the *MILP bounding procedure* on the 743 instances considered. Among these 743 instances, there are 519 known optimal solutions (obtained among all the approaches) which are used to check the performance of the proposed methods. The table groups the results in four main rows. The first row shows the results of the *MILP bounding procedure* and the remaining rows show the results for SA, TS, TS-CM and their combinations with the *Solve model*. For each experiment, the first column ($\%OS_{method}$) states the percentage of instances proven optimal by the method. The second column (*%OS*) gives the percentage of instances proven optimal by comparing the obtained solution with the known optimal solutions. Finally, the third column (Gap_{av}) provides the average relative gap for the 743 instances considered, with respect to the best lower bound available.

What emerges from the obtained results is that a higher percentage of optimal solutions is obtained when the proposed hybrid metaheuristics are combined with MILP than when executed alone (in all cases regarding $\%OS_{method}$ and *%OS* for SA and TS, and in all cases with the exception of $MILP_{2700}$+TS-CM and TS-CM+$MILP_{2700}$ for TS-CM). Specifically, the best optimality percentage was found using the combination TS+$MILP_{2700}$ ($\%OS_{method}$ of 67.03) and $MILP_{2700}$+SA (*%OS* of 69.45). On the other hand, a better Gap_{av} is obtained for six different procedures (1.83; among these six procedures $MILP_{900}$+TS-CM provided the best result -69.31- in terms of percentage of overall optimal solutions, *%OS*).

Table 6. *Results for the hybrid metaheuristic and their combinations with MILP.*

	% Optima		Gap_{av}
	$\%OS_{method}$	*%OS*	
MILP bounding procedure	63.66	65.68	2.56
SA	31.22	56.53	2.99
TS	30.82	67.29	1.88
TS-CM	30.96	68.78	1.83
$MILP_{900}$ + ***SA***	61.78	68.37	2.12
$MILP_{1800}$ + ***SA***	63.80	68.64	2.05
$MILP_{2700}$ + ***SA***	65.41	69.45	1.99
$MILP_{900}$ + ***TS***	61.78	68.78	1.87
$MILP_{1800}$ + ***TS***	63.93	69.04	1.88
$MILP_{2700}$ + ***TS***	65.01	68.78	1.92
$MILP_{900}$ + ***TS-CM***	62.05	69.31	1.83
$MILP_{1800}$ + ***TS-CM***	63.93	68.91	1.88
$MILP_{2700}$ + ***TS-CM***	64.47	68.24	2.01
SA + $MILP_{900}$	62.72	66.89	2.31
SA + $MILP_{1800}$	64.47	67.70	2.25
SA + $MILP_{2700}$	66.22	67.83	2.20
TS + $MILP_{900}$	64.74	68.78	1.83
TS + $MILP_{1800}$	65.81	68.78	1.83
TS + $MILP_{2700}$	67.03	68.91	1.83
TS-CM + $MILP_{900}$	64.60	69.18	1.83
TS-CM + $MILP_{1800}$	65.55	69.18	1.86
TS-CM + $MILP_{2700}$	65.95	68.64	1.98

Next the methods yielding the best percentage of optimal solutions certified with known optima ($MILP_{2700}$+SA) and average relative gap ($MILP_{900}$+TS-CM) are discussed in detail. Table 7 summarizes the results for $MILP_{2700}$+SA (column 1) and for $MILP_{900}$+TS-CM (column 2) compared to those obtained for the *MILP bounding procedure* (column 3). The first row (*% equal CT*) shows the percentage of solutions which provided the same objective function value as in *MILP bounding procedure*. The second row (*% improvement*) gives the percentage of instances that outperform the solution of *MILP bounding procedure*. Rows 3 (*ave*.) and 4 (*max*.) show, respectively, the average and the maximum improvement among such instances. Conversely, the percentage of solutions that worsen the objective is given in row 5 *(% decrease*), and its average and maximum worsening values are shown in rows 6 (*ave*.) and 7 (*max*.), respectively. In both experiments, results show a high percentage of instances that equal (around 75%) or improve (around 25%) the objective function value of *MILP bounding procedure* whereas the percentage of instances that worsen the objective function value is kept low (0.67% and 1.08% for $MILP_{2700}$+SA and $MILP_{900}$+TS-CM, respectively). Row 8 shows the average gap (Gap_{av}) of the 743 instances considered with respect to the best lower bound available. Additionally, row 9 gives the maximum gap, Gap_{max}. Finally, the percentage of optimal solutions, obtained by comparison with known optima (within the 743 instances considered) is shown in row 10 (*% OS*).

Table 7: *Computational results for $MILP_{2700}$ +SA and $MILP_{900}$ + TS-CM with respect to MILP bounding procedure.*

	$MILP_{2700}$ + SA	$MILP_{900}$+TS-CM	MILP bounding procedure
% equal CT	76.18	73.76	-
% improvement	23.15	25.17	-
ave.	2.69	3.15	-
max.	10.41	11.81	-
% decrease	0.67	1.08	-
ave.	1.87	1.39	-
max.	3.51	4.03	-
Gap_{av}	1.99	1.83	2.56
Gap_{max}	23.12	22.05	23.12
% OS	69.45	69.31	65.68

With the proposed approach, the average gap of the solutions that could not be solved optimally by any method (224 out of the set of 1,200 instances), is 5.83%. Up to date, considering all the methods proposed in this thesis, the AWALBP-L2 has been solved optimally for 81.33% of the 1,200 instances.

Effects of the characterictics of instances on performance

To study the effects of the characteristics of instances on the performance (in terms of number of optimal solutions, *NOS*), the results were grouped according to increasing values of the parameters workpiece length A_0, number of workstations m and number of tasks N. Fig. 6 shows the *NOS* obtained according to increasing values of A_0 and N. In the upper graphic of Fig. 6 the instances have been grouped first along the five ranges of N. For each range of N, six ranges of A_0 have been considered, and the total number of instances per combination in any combination of N-A_0 is 40. In the lower graphic of Fig. 6 ranges have been grouped first by A_0 and then by N. As can be seen in Fig. 6, for each range of N the number of optimal solutions decreases as the value of A_0 increases. A_0 is the most influencing parameter on performance, since for the lower values of workpiece length (i.e., up to 15, 20 and 25 lu for methods F1, F2 and all other methods, respectively) almost all instances are solved, no matter the number of tasks, but for larger lengths the percentage of optimal solutions falls considerably. Additionally, for medium to large workpiece lengths the number of optimal solutions decreases as the number of tasks increase. Specifically, if we consider the overall number of known optimal solutions *KOS* (976), all instances with workpieces length up to 15 lu (this is, 1.5 times the width of the workstation) where optimally solved. For lengths between 16 and 25 lu almost all instances were solved optimally (99.5%), and a high percentage for instances with lengths between 26-30 was also obtained (89.5%). On the other hand, for larger lengths the percentage of optimal solutions dropped considerably (63.0% and 36.5% for instances with 31-35 and 36-40 lu, respectively).

Fig. 7 shows the effect on performance of the number of workstations and the workpiece length, for all proposed methods. In the upper graphic of Fig. 7 the instances are ordered first by increasing ranges of m and then by A_0, whereas the lower graphic groups them first by A_0 and then by m. As can be observed in both graphics, for instances with small to medium workpieces lengths the number of workstations has no effect on performance. More specifically, a high number of instances up to 15 and 20 lu are solved optimally for methods F1 and F2, respectively, no matter the number of workstations. The same occurs to instances up to 25 lu for the rest of methods. However, for instances with lengths larger than 26 lu, the number of optimal solutions found falls significantly as the number of workstations increases.

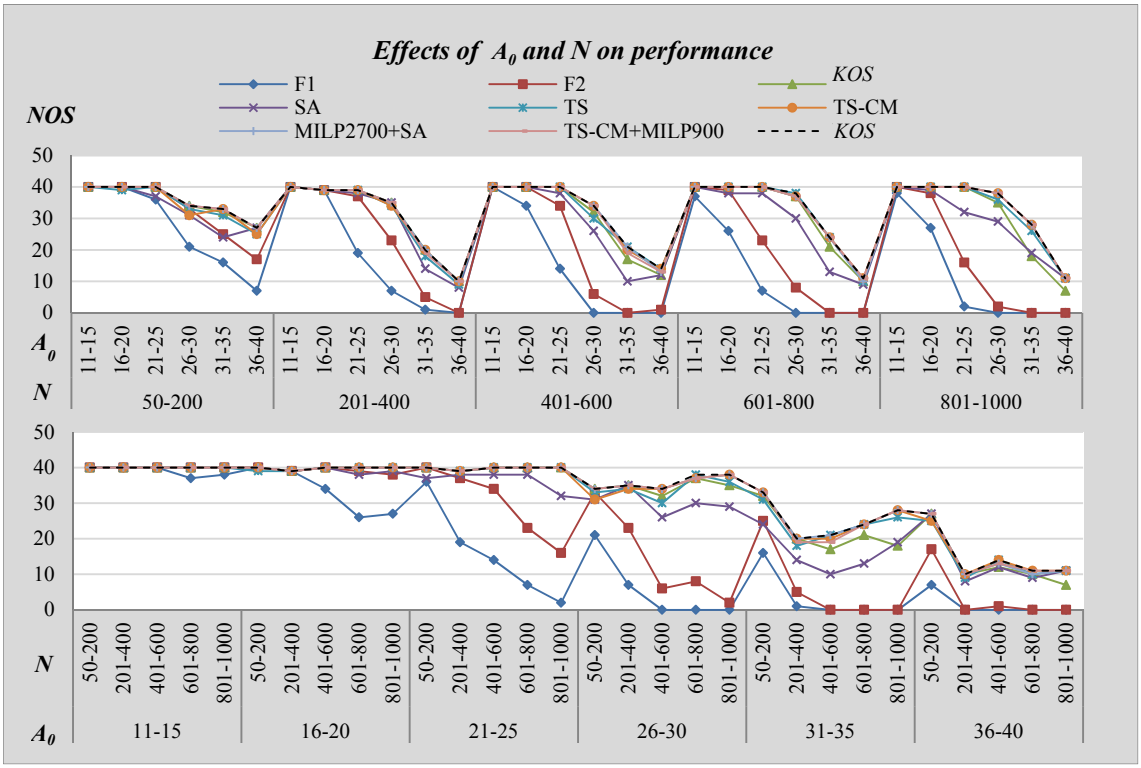

Figure 6: *Optimal solutions according to increasing ranges of A_0 and N.*

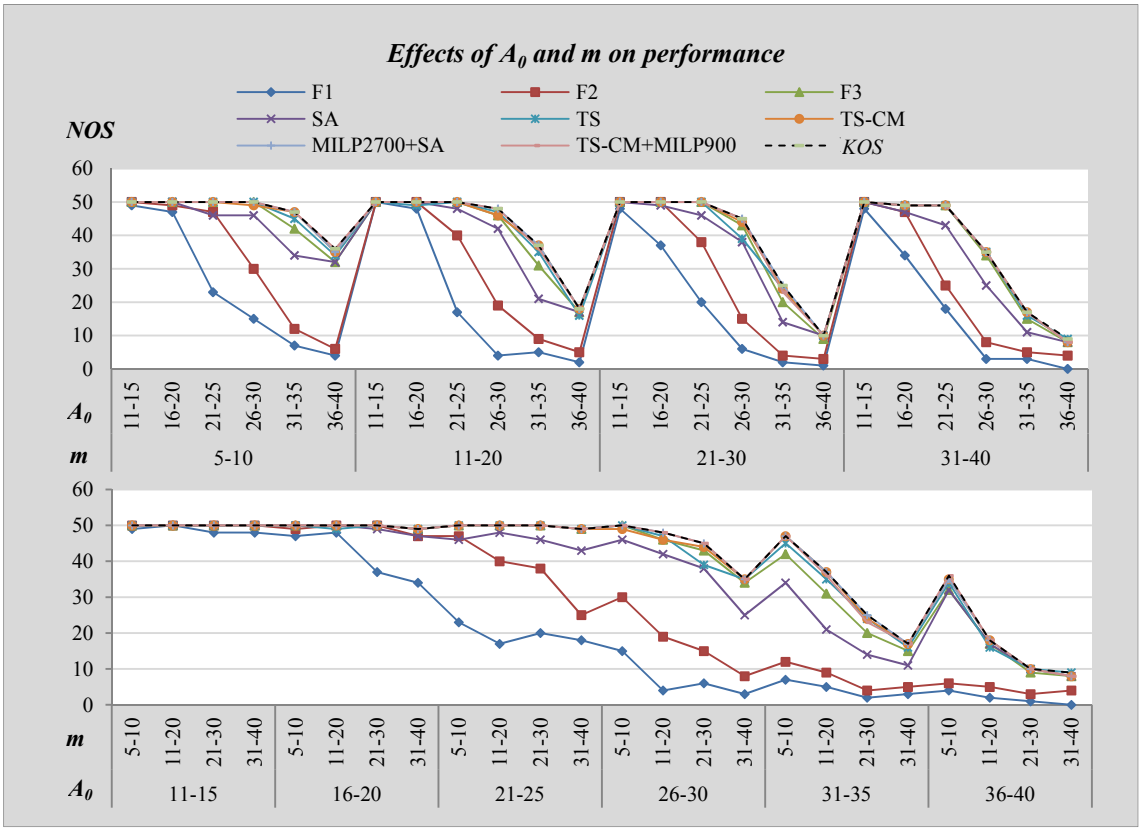

Figure 7: *Optimal solutions according to increasing ranges of A_0 and m.*

Fig. 8 shows the influence of increasing ranges of the number of tasks N with respect to the number of workstations m. As it can be observed in the upper graphic of Fig.8, parameter m generally has the strongest influence on performance since for each range of N the number of optimal solutions NOS decreases steadily as m increases. This situation, though, is reversed for methods F1 and F2, for which the NOS drops significantly as N increases, for all ranges of m.

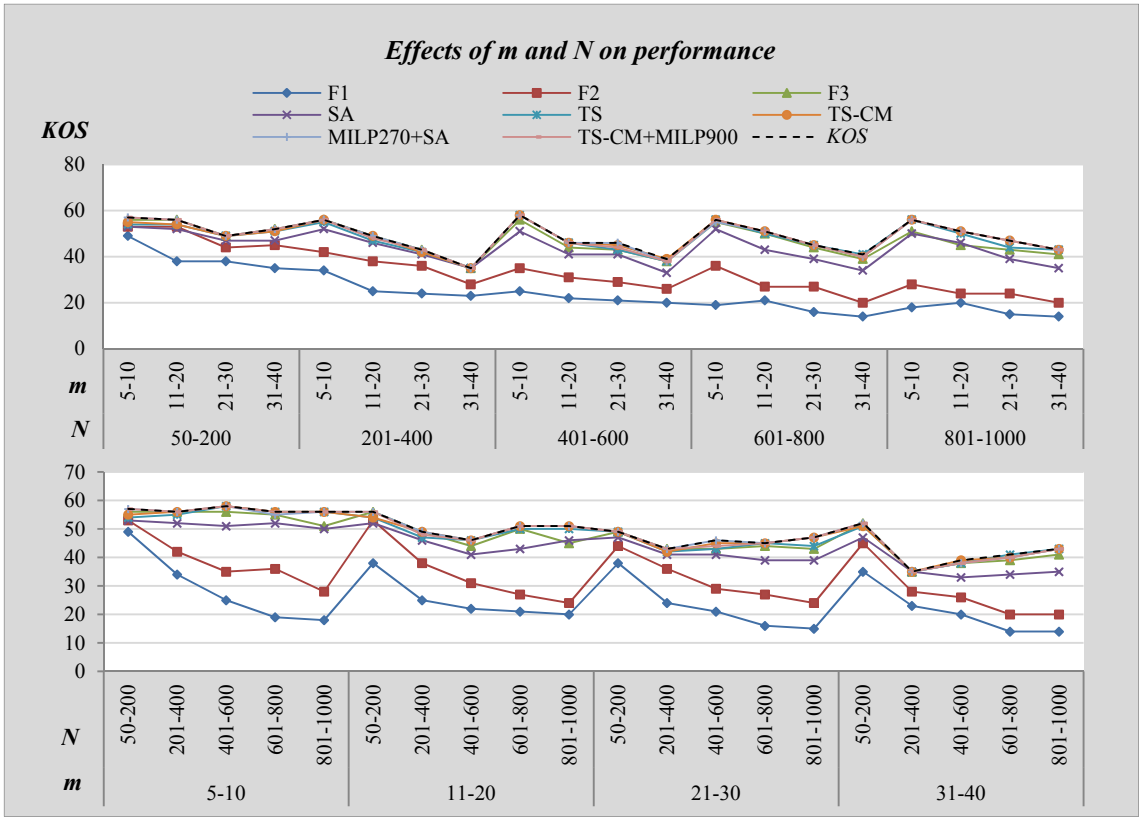

Figure 8: *Optimal solutions according to increasing ranges of m and N.*

Finally, in order to analyze the size limits of the instances that could be solved optimally, the overall percentages of optimal solutions grouped by each individual parameter A_0, m and N are depicted in Fig. 9. From left to right, the graphs show the percentage of known optimal solutions, $\%KOS$, for increasing ranges of A_0, m and N, respectively. As it can be observed in Fig.9, A_0 is the parameter with the highest impact on performance followed by m, as the performance results decrease as those parameters increase. Conversely, parameter N appears to impact in an opposite way, since for 50 to 400 tasks the percentage of optima decreases, whereas for instances with more than 400 tasks, performance increases as the number of tasks increases.

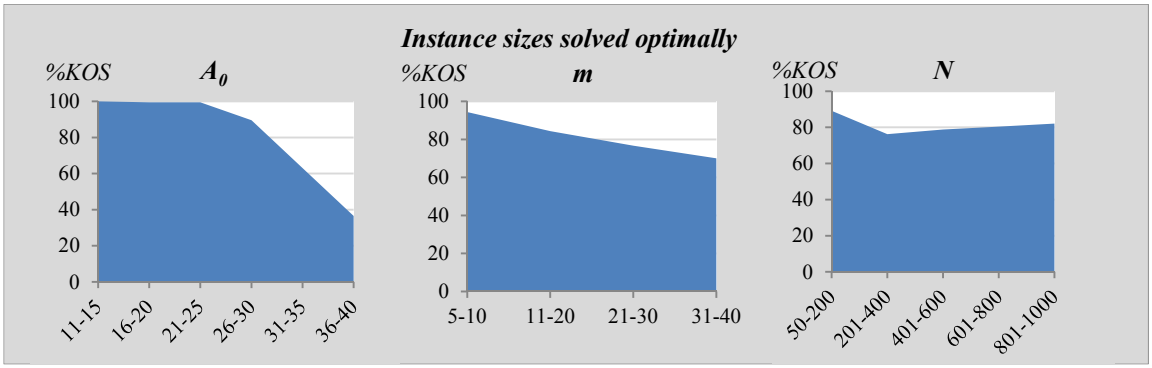

Figure 9: *Performance results according to increasing size instances.*

Chapter 6

Conclusions, publications and future research

6.1 Conclusions

In this doctoral thesis a variant of an assembly line balancing problem of practical significance has been addressed, which has been entitled AWALBP: the *Accessibility Windows Assembly Line Balancing Problem*. The main characteristic of this problem is that the width of the workstations is smaller than the length of the workpieces. This means that each workstation can access in each stationary stage only a restricted set of tasks since its accessibility area is smaller than the workpiece's width. The problem is highly complex and each of its optimization levels implies the solution of one or several *NP-hard* subproblems. Specifically, the optimization level tackled in this thesis is the AWALBP-L2. The case addressed entails the solution of the following two subproblems: (i) the computation of a feasible movement scheme and (ii) the assignment of each task to one stationary stage of the cycle.

The literature review showed that relatively few studies have addressed line balancing problems with accessibility windows and a cyclic transportation pattern of the workpieces. In this thesis, the AWALBP has been described and a classification of its main optimization levels has been proposed. A specific case of level AWALBP-L2 has been formalized, and four solution approaches have been designed and implemented.

With these approaches, the objectives of this thesis have been achieved:

a) Regarding the classification and formalization of the AWALB problem, its optimization levels were identified, and the considered case of AWALBP-L2 was formalized via two alternate mathematical programming formulations, F1 and F2.

b) With respect to the exact methods developed, models F1 and F2 were implemented and tested, but only small to medium-sized instances could be solved to optimality, yielding a 55.58% of optimal solutions. To tackle the challenging instances, *MILP bounding procedure* was developed, improving the percentage of optimal solutions to 78.75%.

c) Finally, concerning hybrid solution methods for the problem, three metaheuristics hybridized with mathematical programming models were developed, based on simulated annealing and tabu search. Two types of neighborhoods were proposed, one relying on the classical move-based type of neighborhood, and another one inspired by the paradigm of

the corridor method. The best result was obtained for a metaheuristic running tabu search with a corridor method, raising the percentage of optima to 80.67%. Furthermore, sequential combinations of a hybrid metaheuristic and a MILP model were also considered, and the best result was obtained with executing the MILP model first and then improving the obtained solution with a simulated annealing. With this approach, the percentage of optimal solutions found increased to 81.08%.

Up to date, among all the proposed solution methods for the considered case of AWALBP-L2, the overall percentage of optimal solutions is 81.33%. If we focus on the feasible solutions of the best approaches (i.e, approaches 2 to 4), solutions with a moderate gap have been obtained, being 5.83% the average deviation of a solution with respect to the best bound available.

In order to achieve the objectives of this thesis, the following contributions have been developed:

1. The AWALBP has been defined and its main levels have been classified. A formal description of the variant AWALBP-L2 has been provided.

2. A literature review on line balancing problems with accessibility windows and cyclic movement patterns has been conducted.

3. A collection of benchmark instances has been created and uploaded online in order to facilitate further research.

4. A number of exact and hybrid solution methods have been designed, implemented and tested for the solution of the addressed case of AWALBP-L2. Among the proposed methods, the following developments could be highlighted:

    4.1 An algorithm which provides a movement scheme with the minimum number of stationary stages. This feature is very important since it allows i) obtaining a lower bound on the number of stationary stages and ii) developing methods that incorporate bounds.

    4.2 A mathematical programming model for the assignment of tasks to stationary stages whose resolution is very fast. The high efficiency of this model allows finding the optimal assignment of tasks for a given movement scheme, and thus it enables the exploration in the space of the movement schemes.

    4.3 An original hybrid metaheuristic using tabu search with corridor method. The proposed method is original in the way how it exploits a MILP model within a tabu search framework. Such model incorporates exogenous constraints to iteratively define a “corridor” around a current movement scheme and to incorporate the tabu lists onto the formulation of the problem.

5. Extensive computational experiments have been conducted for the evaluation of the proposed solution methods. The main findings have been highlighted and proposals for future research work have been provided (see Section 6.3).

The study of the AWALBP-L2 allowed for the development of appropriate methods for its solution, which helps to improve the management of the production processes and to enhance the competitiveness of the industrial network. Furthermore, the fundamental ideas on which the proposed methods are based are open in nature and extend encouraging perspectives either for

tackling other variants of the AWALBP or for addressing other combinatorial optimization problems, as most of the production management problems are.

6.2 Derived works

The written works derived from the research undertaken in this thesis are listed below.

Articles in journals included in the JCR

1. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. A MILP model for the Accessibility Windows Assembly Line Balancing Problem (AWALBP). *International Journal of Production Research,* 51 (12), 3549-3560, 2013.

2. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. Combining matheuristics and MILP for the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2). *Computers and Operations Research,* 48, 113-123, 2014a.

3. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. Balancing assembly lines with accessibility windows. Problem description and heuristic solution procedure. *DYNA*, 89 (5), 552-559, 2014b.

4. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. Hybrid metaheuristics for the Accessibility Windows Assembly Line Balancing Problem (in review process at *European Journal of Operational Research*).

Communications to conferences

5. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. Exact and heuristic approaches for the Visibility Windows Assembly Line Balancing Problem (VWALBP), in *Proceedings of the 12th Annual Congress of the French National Society of Operations Research and Decision Science, 12è Congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2011)*, 583-584, Saint-Étienne, France, 2-4 March, 2011.

6. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. Heurísticas para el Visibility Windows Assembly Line Balancing Problem (VWALBP). *Book of full papers: 5th International Conference on Industrial Engineering and Industrial Management*. *XV Congreso de Ingeniería de Organización (CIO 2011)*, 201-205, Cartagena, Spain, 7-9 September, 2011.

7. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. Enhanced MILP model for the Accessibility Windows Assembly Line Balancing Problem (AWALBP). In *Proceedings of the 13rd Annual Congress of the French National Society of Operations Research and Decision Science, 13è Congrès Annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2012)*, 117-118, Angers, France, 11-13 April, 2012.

8. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. Modelo de PLEM mejorado para el Accessibility Windows Assembly Line Balancing Problem (AWALBP). In *Proceedings of the 6th International Conference on Industrial Engineering and Industrial Management: XVI Congreso de Ingeniería de Organización (CIO 2012)*, 879-886, Vigo, Spain, 18-20 July, 2012.

9. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. Using tabu search and MILP for the Accessibility Windows Assembly Line Balancing Problem (AWALBP). In *Proceedings of the XXXIV Congreso Nacional de Estadística e Investigación Operativa (SEIO 2013)*, 117, Castellón, Spain, 11-13 september, 2013.

10. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. Using simulated annealing and MILP for the Accessibility Windows Assembly Line Balancing Problem (AWALBP). In *Proceedings of the XXVI EURO-INFORMS Joint International Conference*, 26th European Conference on Operational Research (EURO 2013), 38, Rome, Italy, 1-4 July, 2013.

11. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. The Accessibility Windows Assembly Line Balancing Problem (AWALBP): A review of advances and trends. In *Proceedings of the 20th Conference of the International Federation of Operational Research Societies (IFORS 2014)*, 191. Barcelona, 13-18 July, 2014.

12. Corominas, A., Calleja, G., García-Villoria, A., and Pastor, R. MILP-based Tabu Search using Corridor Method for an assembly line balancing problem with accessibility windows. In *Proceedings of the 20th Conference of the International Federation of Operational Research Societies (IFORS 2014)*, 191-192, Barcelona, 13-18 July, 2014.

Books

13. Calleja, G., Corominas, A., García-Villoria, A., and Pastor, R. Balancing assembly lines with accessibility windows. *SpringerBriefs in Operations Management, Springer*. Editor: Sethi, S. (in preparation).

6.3 Future research

The *Accessibility Windows Assembly Line Balancing Problem* involves multiple optimization levels and new features of practical relevance that can be addressed. The following extensions or alternative assumptions can be considered:

Several workstations compatible for each task. This extension adds a higher complexity to the problem. Now it is necessary not only to decide in which stationary stage a task has to be assigned, but also in which one of the available workstations.

Precedence relationships among tasks. When dealing with workpieces larger than the workstations some situations arise that do not occur in assembly lines without restricted accessibility windows. For example, even in the case of one single workstation compatible per task, it is necessary to explicitly take into account the precedence relationships among tasks that are executed in different workstations.

Multiple robot arms at each workstation. What emerges from the analysis of the literature is that research on line balancing problems considering multiple robot arms per workstation is scarce or at least it does not involve large workpieces. The research would focus first on testing existing methods proposed for other balancing problems and subsequently either adapt or design new efficient procedures for the AWALBP-L2.

Regarding the solution approaches proposed for the problem, the following extensions could be considered:

Parallel or intertwined execution. In this thesis, collaborative combinations of sequential executions of MILP and metaheuristics have been proposed, in which the computational time

has been divided into two parts. An interesting future line of research concerns the collaborative combination of MILP and metaheuristics in a parallel or intertwined way. In the parallel execution, the MILP solver and the metaheuristic are launched in parallel, and they may pass along information to each other when something relevant for the algorithm occurs (e.g., a new incumbent solution or a new bound is found). Conversely, in the intertwined execution the methods are launched sequentially. In this case, either the allowed time for the methods can be divided in many parts, or each method can be executed until something relevant for the algorithm occurs.

References

Adenso-Díaz, B., and Laguna, M. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54 (1), 99-114, 2006.

Adham, A. Zainuddin, H., Siali, F., and Azizan, N. Assembly line balancing in manufacturing processes: Using simulation model. *Advanced Materials Research*, 748, 1183-1187, 2013.

Aghajani, M., Ghodsi, R., and Javadi, B. Balancing of robotic mixed-model two-sided assembly line with robot setup times. *The International Journal of Advanced Manufacturing Technology*, DOI: 10.1007/s00170-014-5945-x, article in press, 2014.

Akpinar, S., and Bayhan, G. Performance evaluation of ant colony optimization-based solution strategies on the mixed-model assembly line balancing problem. *Engineering Optimization*, 46 (6), 842-862, 2014.

Amen, M. Heuristic methods for cost-oriented assembly line balancing: A comparison on solution quality and computing time. *International Journal of Production Economics*, 69, 225-264, 2001.

Amen, M. Cost-oriented, assembly line balancing: model formulations, solution difficulty, upper and lower bounds. *European Journal of Operational Research*, 168 (3), 747-770, 2006.

Ammons, J.C., Carlyle, M., Cranmer, L., Depuy, G., Ellis, K., McGinnis, L.F., Tovey, C.A., and Xu, H. Component allocation to balance workload in printed circuit card assembly systems. *IIE Transactions*, 29, 265-275, 1997.

Andrés, C., Miralles, C., and Pastor, R. Balancing and sequencing tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, 187 (3), 1212-1223, 2008.

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

Baldacci, R., Maniezzo, V., and Mingozzi, A. An exact method for the car pooling problem based on lagrangean column generation. *Operations Research*, 52, 3, 422-439, 2004.

Bartholdi, J. Balancing two-sided assembly lines: a case study. *International Journal of Production Research*, 31, 10, 2447-61, 1993.

Bartholdi, J., and Eisenstein, D. A production line that balances itself. *Operations Research*, 44 (1), 21-234, 1996.

Battaïa, O., and Dolgui, A. A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, 142 (2), 259-277, 2013.

Battaïa, O., Dolgui, A., Guschinsky, N., and Levin, G. A decision support system for design of mass production machining lines composed of stations with rotary or mobile table. *Robotics and Computer-Integrated Manufacturing*, 28, 672-680, 2012.

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

Bautista, J., and Pereira, J. Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research*, 177 (3), 2016-2032, 2007.

Bautista, J., and Pereira, J. Procedures for the time and space constrained assembly line balancing problem. *European Journal of Operational Research*, 212 (3), 473-481, 2011.

Baybars, I. A survey on exact algorithms for the simple assembly line balancing problem. *Management Science*, 32 (8), 909-932, 1986a.

Baybars, I. An efficient heuristic method for the simple assembly line balancing problem. *International Journal of Production Research*, 24 (1), 149-166, 1986b.

Becker, C., and Scholl, A. A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168 (3), 694-715, 2007.

Blum, C., and Miralles, C. On solving the assembly line worker assignment and balancing problem via beam search. *Computers & Operations Research*, 38 (1), 328-339, 2011.

Blum, C., Puchinger, J., Raidl, G., and Roli, A. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11 (6), 4135-4151, 2011.

Boctor, F. A multiple-rule heuristic for assembly line balancing. *Journal of Operational Research Society*, 46, 62–69, 1995.

Boschetti, M., Maniezzo, V., Roffilli, M., Bolufé, A. Matheuristics: optimization, simulation and control. In *Lecture Notes in Computer Science*. Hybrid metaheuristics, 6th international workshop, HM, 5818, 171-177, 2009.

Borisovsky, P., Dolgui, A., and Kovalev, S. Algorithms and implementation of a set partitioning approach for modular machining line design. *Computers & Operations Research*, 39 (12), 3147-3155, 2012.

Boussaïd, I., Lepagnot, J., and Siarry, P. A survey on optimization metaheuristics, *Information Sciences,* 237, 82-117, 2013.

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

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

Bratcu, A., and Dolgui, A. A survey of the self-balamcing production lines ("bucket brigades"). *Journal of Intelligent Manufacturing*, 16 (2), 139-158, 2005.

Bryton, B. *Balancing of a continuous production line*. M.Sc. Thesis, Northwestern University, Evanston, Illinois, 1954.

Buxey, G.M. Assembly line balancing with multiple stations. *Management Science*, 20 (6), 1010-1021, 1974.

Capacho, L., and Pastor, R. The ASALB Problem with processing alternatives involving different tasks: definition, formalization and resolution. *Lecture Notes in Computer Science*, 3982, 554-563, 2006.

Chica, M., Cordón, O., Damas, S., and Bautista, J. A multiobjective GRASP for the 1/3 variant of the time and space assembly line balancing problem. *Trends in Applied Intelligent Systems*, 6098, 656-665, 2010.

Corominas, A., and Pastor, R. A MILP model for the Visibility Windows Assembly Line Balancing Problem (VWALBP): the case of the Müller-Hannemann & Weihe problem. Working paper. Universitat Politècnica de Catalunya. Available from: http://upcommons.upc.edu/e-prints/bitstream/2117/7047/1/IOC-DT-P-2009–09.pdf, 2009.

Corominas, A., Pastor, R., and Plans, J., 2008. Balancing assembly lines with skilled and unskilled workers. *Omega*, 36 (6), 1126-1132, 2008.

Corominas, A., Ferrer, L., and Pastor, R. Assembly line balancing: general resource-constrained case. *International Journal of Production Research*, 49 (12), 3527-3542, 2011.

Crama, Y., Van de Klundert, J., and Spieksma, F.C.R. Production planning problems in printed circuit board assembly. *Discrete Applied Mathematics*, 123 (1-3), 339-361, 2002.

Delorme, X., Dolgui, A., and Kovalyov, M. Combinatorial design of a minimum cost transfer line. *Omega*, 40 (1), 31-41, 2012.

Digiesi, S., Kock, A., Mummolo, G., and Rooda, J. The effect of dynamic worker behavior on flow line performance. *International Journal of Production Economics*, 120 (2), 368-377, 2009.

Dolgui, A., Guschinsky, N., Levin, G., and Proth, J. Optimisation of multi-position machines and transfer line. *European Journal of Operational Research*, 185 (3), 1375-1389, 2008.

Dolgui, A., and Proth, J. *Supply Chain Engineering: useful methods and techniques*. Springer, 2010.

Dong, D., Zhang, L., Xiao, T., and Mao, H. Balancing and sequencing of stochastic mixed-model assembly U-lines to minimise the expectation of work overload time. *International Journal of Production Research*, DOI: 10.1080/00207543.2014.944280, 2014.

Dou, J., Dai, X., and Meng, Z. A GA-based approach for optimizing single-part flow-line configurations of RMS. *Journal of Intelligent Manufacturing*, 22 (2), 301-317, 2011.

Erel, E., and Gökçen, H. Shortest-route formulation of mixed-model assembly line balancing problem. *European Journal of Operational Research*, 16 (1), 194-205, 1999.

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

Essafi, M., Delorme, X., Dolgui, A., and Guschinskaya, O. A MIP approach for balancing transfer line with complex industrial constraints. *Computers & Industrial Engineering*, 58 (3), 393-400, 2010.

Essafi, M., Delorme, X., and Dolgui, A. A reactive GRASP and Path Relinking for balancing reconfigurable transfer lines. *International Journal of Production Research*, 50 (18), 5213-5328, 2012.

Finnsgård, C., and Wänström, C. Factors impacting manual picking on assembly lines: an experiment in the automotive industry. *International Journal of Production Research*, 51 (6), 1789-1798, 2013.

Gamberini, R., Grassi, E.G.A., and Regattieri, A. A multiple single-pass heuristic algorithm solving the stochastic assembly line rebalancing problem. *International Journal of Production Research*, 47 (8), 2141-2164, 2009.

Gao, J., Sun, L., Wang, L., and Gen, M. An efficient approach for type II robotic assembly line balancing problems. *Computers & Industrial Engineering*, 56 (3), 1065-1080, 2009.

Gaudlitz, R. *Optimization algorithms for complex mounting machines in PC board manufacturing*. Doctoral thesis. Technical University of Darmstadt, 2004.

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

Glover, F., and Laguna, M. *Tabu Search*. Kluwer Academic Publishers, 1997.

Ghosh, S., and Gagnon, R. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research*, 27 (2), 637-670, 1989.

Gökçen, H., Kara, Y., and Atasagun Y. Integrated line balancing to attain Shokinka in a multiple straight line facility. *International Journal of Computer Integrated Manufacturing*, 23 (5), 402-411, 2010.

Gungor, A., and Gupta, S. A solution approach to the disassembly line balancing problem in the presence of task failures. *International Journal of Production Research*, 39 (7), 1427-1467, 2001.

Guschinskaya, O., Dolgui, A., Guschinsky, N., and Levin, G. A heuristic multi-start decomposition approach. *European Journal of Operational Research*, 189 (3), 902-913, 2008.

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

Hamta, N. Fatemi Ghomi, S., Jolai, F. and Bahalke, U. Bi-criteria assembly line balancing by considering flexible operation times. *Applied Mathematical Modelling*, 35 (12), 5592-5608, 2011.

Hansen, P., and Mladenović, N. An introduction to variable neighborhood search. In Voβ, S., Martello, S., Osman, I., and Roucairol, C., editors, *Metaheuristics: advances and trends in local search paradigms for optimization*, 433-438. Kluwer Academic Publishers, 1999.

Hao, N. *Sequencing and balancing of mixed model assembly line with window cycle time*. Doctoral thesis. Universitat Politècnica de Catalunya, Barcelona, Spain, 2005.

Hao, Y., and Wei, S. A genetic algorithm for multi-model assembly line balancing problem. *Proceeedings of the IEEE International Symposium on Assembly and Manufacturing (ISAM)*, 369-371, 2013.

Helgeson, W., and Birnie, D. Assembly line balancing using the ranked positional weight technique. *Journal of Industrial Engineering*, 12, 394–398, 1961.

Hoffmann, T. Eureka: a hybrid system for assembly line balancing. *Management Science*, 38 (1), 39-47, 1992.

Hu, X., Wu, E., Jinsong, B., and Jin, Y. A branch-and-bound algorithm to minimize the line length of a two-sided assembly line. *European Journal of Operational Research*, 206 (3), 703-707, 2010.

Inman, R., and Leon, M. Scheduling duplicate serial stations in transfer lines. *International Journal of Production Research*, 32 (11), 2631-2644, 1994.

Jayaswal, S., and Agarwal, P. Balancing U-shaped assembly lines with resource dependent task times: A Simulated Annealing approach. *Journal of Manufacturing Systems*, DOI: 10.1016/j.jmsy.2014.05.002, article in press, 2014.

Johnson, R. Assembly line balancing algorithms. *International Journal of Production Research*, 19, 277–287, 1981.

Johnson, M., and Smed, J. Observations on PCB assembly optimization. *Electronic Packaging & Production*, 41 (5), 38-42, 2001.

Junsong, L. Applied technology in assembly line balancing based on genetic algorithm and simulation. *Advanced Materials Research*, 886, 564 -567, 2014.

Kalayci, C., and Gupta, S. A tabu search algorithm for balancing a sequence-dependent disassembly line. *Production, Planning & Control. The Management of Operations*, 25 (2), 149-160, 2014.

Kara, Y., Paksoy, T., and Chang, C. Binary fuzzy goal programming approach to single model straight and U-shaped assembly line balancing. *European Journal of Operational Research*, 195 (2), 335-347, 2009.

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

Kazemi, S., Ghodsi, R., Rabanni, M., and Tavakkoli-Moghaddam, R. A novel two-stage genetic algorithm for a mixed-model U-line balancing problem with duplicated tasks. *International Journal of Advanced manufacturing Technology*, 55 (9-12), 1111-1122, 2011.

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

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. Optimization by simulated annealing. *Science*, 220, 671-680, 1983.

Kumar, N., and Mahto, D. Assembly Line Balancing: A review of developments and trends in approach to industrial application. *Global Journal of Researches in Engineering*, 13 (2), 29-50, 2013.

Lapierre, S.D., and Ruiz, A.B. Balancing assembly lines: An industrial case study. *Journal of the Operational Research Society*, 55 (6), 589-597, 2004.

Lusa, A. A survey of the literature on the multiple or parallel assembly line balancing problem. *European Journal of Industrial Engineering*, 2 (1), 50-72, 2008.

Maniezzo, V., Stützle, T., and Voβ, S. Matheuristics: hybridizing metaheuristics and mathematical programming. *Annals Information Systems*, 10. Springer, 2009.

Martin, R. *Modeling an optimization problem from the automated manufacturing of PC boards*. Diploma Thesis. Univerität Konstanz, 2002.

Miltenburg, G. Balancing and scheduling mixed-model U-shaped production lines. *International Journal of Flexible Manufacturing Systems*, 14 (2), 119-151, 2002.

Miltenburg; G., and Wijngaard, J. U-line balancing problem. *Management Science*, 40, 1378–1388, 1994.

Miralles, C., García-Sabater, J., Andrés, C., and Carlos, M. Advantages of assembly lines in sheltered work centres for disabled. A case study. *International Journal of Production Economics*, 110 (1-2), 187-197, 2007.

Miralles, C., García-Sabater, J., Andrés, C., and Carlos, M. Branch and bound procedure for solving the assembly line worker assignment and balancing problem: application to sheltered work centres for disabled. *Discrete Applied Mathematics,* 156 (3), 352-367, 2008.

Mohd-Hafizuddin, M., Ahmad-Nazif, N.K., Mohd-Needza, Y., and Azila-Nadiah, D. A study on line balancing in assembly line at automotive component manufacture. *Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management*, Istanbul, Turkey, July 3 –6, 2012.

Mozdgir, A., Mahdavi, I., Badeleh, I.S., and Solimanpur, M. Using the Taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing. *Mathematical and Computer Modelling*, 57 (1-2), 137-151, 2013.

Morrison, D. An application of the branch, bound, and remember algorithm to a new simple assembly line balancing dataset. *European Journal of Operational Research*, 236 (2), 403-409, 2014.

Müller-Hannemann, M., and Weihe, K. Moving policies in cyclic assembly line scheduling. *Theoretical Computer Science*, 351 (3), 425-436, 2006.

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

Özcan, U, and Toklu, B. Multiple-criteria decision-making in two-sided assembly line balancing: a goal programming and a fuzzy goal programming models. *Computers & Operations Research*, 36 (6), 1955-1965, 2009.

Pachghare, V., and Dalu, R. S. Assembly Line Balancing - A review. *International Journal of Science and Research*, 3 (3), 807-811, 2014.

Pape, T. Heuristics and lower bounds for the simple assembly line balancing problem type 1: Overview, computational tests and improvements. *European Journal of Operational Research*, 240 (1), 32-42, 2015.

Pastor, R., Andrés, C., Durán, A., and Pérez, M. Tabu search algorithms for an industrial multi-product and multi-objective assembly line balancing problem, with reduction of the task dispersion, *Journal of the Operation Research Society*. 53, 1317-1323, 2002.

Pastor, R., and Corominas, A. Assembly line balancing with incompatibilities and bounded workstation loads. *Ricerca Operativa*, 30 (93), 23-45, 2000.

Pastor, R., and Ferrer, L. An improved mathematical program to solve the simple assembly line balancing problem. *International Journal of Production Research*, 47 (11), 2943-2959, 2009.

Pastor, R. LB-ALBP: the lexicographic bottleneck assembly line balancing problem. *International Journal of Production Research*, 49 (8), 2425-2442, 2011.

Pastor, R., Chueca, I., and García-Villoria, A. A heuristic procedure for solving the lexicographic bottleneck assembly line balancing problema. (LB-ALBP). *International Journal of Production Research*, 50 (7), 1862-1876, 2012.

Pinto, P. A heuristic network procedure for the assembly line balancing problem. *Naval Research Logistics Quarterly*, 25 (25), 229-307, 1978.

Prenting, T.O., and Thomopoulos, N.T. *Humanism and technology in assembly line systems*. Hayden, Rochelle Park, NJ., 1974.

Puchinger, J., and Raidl, G. Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach in Lecture Notes in Computer Science*, 3562, 41-53, 2005.

Raidl, G. A unified view on hybrid metaheuristics. Hybrid metaheuristics, in *Lecture Notes in Computer Science*, 4030, 1-12, Springer, 2006.

Sabuncuoglu, I., Erel, E., and Gocgun, Y. Analysis of serial production lines: characterisation study and a new heuristic procedure for optimal buffer allocation. *International Journal of Production Research*, 44 (13), 2499-2523, 2006.

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

Salveson, M. The assembly line balancing problem. *Journal of Industrial Engineering*, 6 (3), 18-15, 1955.

Scholl, A. Simple assembly line balancing – heuristic approaches. *Journal of heuristics*, 2 (3), 217-244, 1997.

Scholl, A. *Balancing and sequencing of assembly lines*. Physica-Verlag, Heidelberg, Second edition, 1999.

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

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

Scholl, A., and Voβ, S. Simple assembly line balancing – Heuristic approaches. *Journal of Heuristics*, 2, 217-244, 1996.

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

Sewell, E. and Jacobson, S. A branch, bound, and remember algorithm for the simple assembly line balancing problem. *INFORMS Journal on Computing*, 24 (3), 433-442, 2012.

Sniedovich, M, and Voβ, S. The corridor method: a dynamic programming inspired metaheuristic. *Control and Cybernetics*, 35 (3), 551-578, 2006.

Stille, W. *Solution techniques for specific bin packing problems with applications to assembly line optimization*. Diploma thesis. Technical University of Darmstadt, 2008.

Talbot, F., Patterson, J., and Gehrlein, W. A comparative evaluation of heuristic line balancing techniques. *Management Science*, 32 (4), 431-453, 1986.

Tapkan, P., Özbakir, L., and Baykasoglu, A. Bees algorithm for constrained fuzzy multi-objective two-sided assembly line balancing problem. *Optimization Letters*, 38 (9), 11947-11957, 2011.

Tazari, S. *Algorithmic approaches for two fundamental optimization problems: workload balancing and planar Steiner trees*. Diploma thesis. Technical University of Darmstadt , 2006.

Toksari, M. Isleyen, S. Güner, E., and Baykoç, O. Simple and U-type assembly line balancing problems with a learning effect. *Applied Mathematical Modelling*, 32 (12), 2954-2961, 2008.

Toksari, M., Isleyen, S. Güner, E., and Baykoç, O. Assembly line balancing problem with deterioration tasks and learning effect. *Expert Systems with Applications*, 37 (2), 1223-1228, 2010.

Tonelli, F., Paolucci, M., Anghinolfi, D., and Taticchi, P. Production planning of mixed-model assembly lines: a heuristic mixed integer programming based approach. *Production, Planning & Control*, 24 (1), 110-127, 2013.

Tonge, F.M. *A heuristic program for assembly line balancing*. Prentice-Hall, Englewood Cliffs, NJ., 1961.

Topalogu, S., Salum, L., and Supciller, A. Rule-based modeling and constraint programming based solution of the assembly line balancing problem. *Expert Systems with Applications*, 39 (3), 3484-3493, 2012.

Van Duijnhoven, R. *Computing a toolbit pre-assignment for the AX machine*. Diploma thesis. Eindhoven University of Technology, 2013.

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

Xu, W., and Xiao, T. Strategic robust mixed-model assembly line balancing based on scenario planning. *Tsinghua Science & Technology*, 16 (3), 308- 314, 2011.

Yang, C., Gao, J., and Sun, L. A multi-objective genetic algorithm for mixed-model assembly line rebalancing. *Computers & Industrial Engineering*, 65 (1), 109-116, 2011.

Wang, F., and Wilson, R.C. Comparative analyses of fixed and removable item mixed model assembly lines. *IIE Transactions*, 18 (3), 313-317, 1986.

Wee, T. and Magazine, M. Assembly line balancing as generalized bin packing. *Operations Research Letters*, 1, 56-58, 1986.

Annex A1. Articles published in journals included in the JCR

ATTENTION ¡

Pages 52 to 88 of the thesis are available at the editor's web

- Gema Calleja, Albert Corominas, Alberto Garcia-Villoria and Rafael Pastor ***A MILP model for the Accessibility Windows Assembly Line Balancing Problem (AWALBP)***

  Article published in [International Journal of Production Research, Volume 56, 12, Pages 3549-3560] [DOI: 10.1080/00207543.2012.751514]© [Copyright Taylor & Francis Group]

  http://www.tandfonline.com/doi/abs/10.1080/00207543.2012.751514#.VS5eruHXtA8

- Gema Calleja, Albert Corominas, Alberto García-Villoria, Rafael Pastor ***Combining matheuristics and MILP to solve the Accessibility Windows Assembly Line Balancing Problem (AWALBP-L2)***

  Article published in [Computers & Operations Research, Volume 48, Pages 113-123] [DOI: 10.1016/j.cor.2014.03.009]© [Copyright Elsevier]

  http://www.sciencedirect.com/science/article/pii/S0305054814000641

- Gema Calleja, Albert Corominas, Alberto Garcia-Villoria and Rafael. Pastor ***Balancing assembly lines with accessibility windows. Problem description and heuristic solving procedure***

  Article published in [DYNA Volume 89, 5, Pages 552-559] [DOI: http://dx.doi.org/10.6036/7051]© [Copyright Revista de Ingeniería DYNA]

  http://www.revistadyna.com/search/balancing-assembly-lines-with-accessibility-windows-problem-description-and-solving-heuristic-proced

Annex A2. Other works

A2.1. Articles submitted to journals included in the JCR which are in process of review

Hybrid metaheuristics for the Accessibility Windows Assembly Line Balancing Problem (AWALBP)

Hybrid metaheuristics for the Accessibility Windows Assembly Line Balancing Problem (AWALBP)

Gema Calleja, Albert Corominas, Alberto García-Villoria and Rafael Pastor

Institute of Industrial and Control Engineering (IOC)
Universitat Politècnica de Catalunya (UPC)
Diagonal 647, 11th floor, 08028, Barcelona, Spain
{gema.calleja,albert.corominas,alberto.garcia-villoria,rafael.pastor}@upc.edu

Abstract. This paper addresses an assembly line balancing problem in which the length of the workpieces is larger than the width of the workstations. The problem differs from traditional variants of assembly line balancing in the sense that only a portion of the workpiece, or portions of two consecutive workpieces, can be reached from any workstation. Consequently, at any stationary stage of the cycle, each workstation can only process a portion of the tasks, namely, those which are inside the area of a workpiece that is reachable from the workstation. The objective is to find a (cyclic) movement scheme of the workpieces along the line and a task assignment to stationary stages of the production process, while minimizing the cycle time. We propose three hybrid approaches of metaheuristics and mathematical programming - one based on simulated annealing and the other two based on tabu search, relying on different neighborhood definitions. The two former approaches make use of a classical neighborhood, obtained by applying local changes to a current solution. The latter approach, in contrast, draws ideas from the corridor method to define a *corridor* around the current solution, via the imposition of exogenous constraints on the solution space of the problem. An extensive computational experiment is carried out to test the performance of the proposed approaches, improving the best results published to date.

Keywords: Assembly line balancing, accessibility windows, hybrid metaheuristics, simulated annealing, tabu search, corridor method

1. Introduction

As global competition and technological change accelerates, manufacturers have become increasingly interested in optimizing their production and assembly systems. In this paper, we consider a special case of assembly system that widely arises in advanced automated

environments, especially in the assembly of electronic components: the assembly line with accessibility windows. The line consists of a set of workstations sequentially arranged along a transport system, which must process a number of identical workpieces. Every workstation contains a feeder with several component types and is equipped with a robot arm, which performs tasks on the workpieces. Each workstation must process a specific set of tasks on each workpiece. The tasks correspond to pick-and-place actions; picking a component type from the feeder inside the workstation and placing it on a predefined position on the workpiece (see Fig. 1).

The workpieces are fed into the assembly line starting from a reference position x (see Fig. 2), and are moved in *forward steps*, according to a pattern called *movement scheme*. In every halt between two forward steps, the line stands motionless and the workstations perform tasks on the workpieces. Such a halt is called a *stationary stage*. The forward steps are cyclic: after S forward steps, there is an identical number of workpieces lying exactly at the same positions as in the start of the cycle. The length of each forward step must be a multiple of a distance Δ called *elementary step*, which depends on the technology of the line. After each cycle, a new workpiece enters the line. At the same time, a fully assembled workpiece leaves the line.

Fig. 2 illustrates an example of a cycle with three stationary stages (thus the fourth stationary stage is identical to the first stage). Each line is a snapshot representing the positions of the workpieces in the stationary stage. The initial position of the first workpiece in the beginning of the cycle is defined by the distance x. The arrows on each snapshot represent the forward steps. Note that, in this example, the lengths of the forward steps are different.

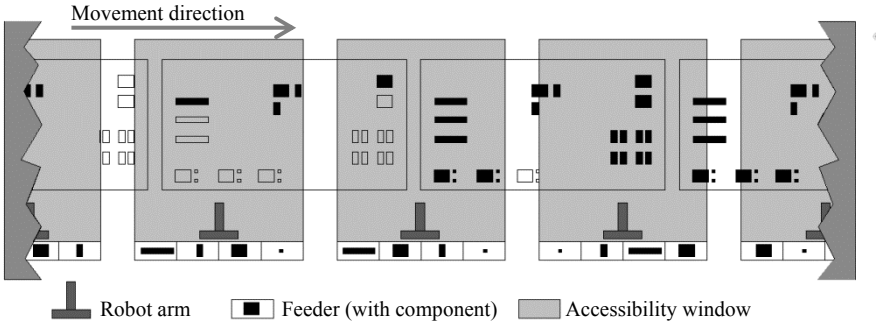

Figure 1. *An example of an assembly line with accessibility windows*

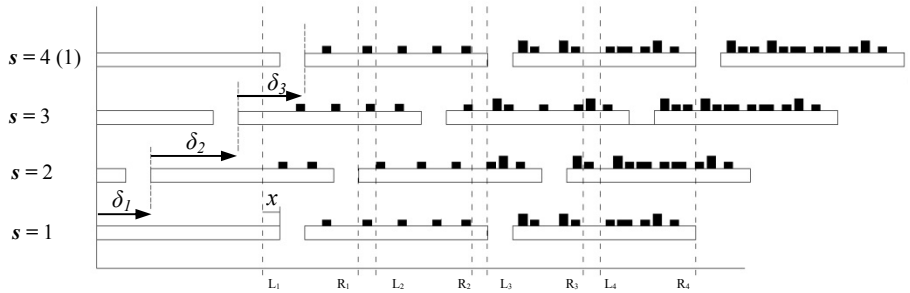

Figure 2. *Four snapshots of a cycle with three stationary stages*

Unlike common assembly lines, in this kind of line the length of the workpieces is longer than the width of the workstations. Consequently, one workpiece may be processed by several workstations at the same time, and one workstation may process portions of either one or two consecutive workpieces at the same time (recall Fig. 1). Therefore, a task can only be performed if it is situated inside the reachable interval $[L_i, R_i]$ (*accessibility window*) of the workstation i where it will be executed (see Fig. 2). This environment, where task positioning limits the

access to restricted areas of the workpiece, motivates the so-called accessibility windows assembly line balancing problem (AWALBP) (Calleja *et al*., 2013).

The AWALBP is a variant of the Generalized Assembly Line Balancing Problem (GALBP), which includes problems with specific real-world restrictions and has been subject to extensive research (see, for example, Becker and Scholl, 2006; Capacho *et al*., 2009; Martino and Pastor, 2010; Corominas *et al*., 2011; Battaïa and Dolgui, 2012; Tuncel and Topaloglu, 2013; and Sternatz, 2014). The optimization of AWALBP involves the solution of several *NP*-hard subproblems (Gaudlitz, 2004). With regard to the subproblems considered, the AWALBP can be tackled at four optimization levels (Calleja *et al*., 2013): the assignment of each task to one compatible workstation and stationary stage (AWALBP-L1); the initial position of the workpieces in the cycle, as well as the number and the length of the forward steps (AWALBP-L2); the component type allocation to feeders (AWALBP-L3); and the number and the type of workstations (AWALBP-L4). The objective is to minimize the cycle time. Each level addresses the optimization of its own level as well as its predecessors. For example, in AWALBP-L2 levels L1 and L2 are to be solved when solutions of L3 and L4 are given. A detailed description of AWALBP and its variants, along with a literature review has been presented in Calleja *et al*. (2013).

This paper deals with the case of AWALBP-L2 defined in Müller-Hannemann and Weihe (2006). To solve this problem, two different approaches have been proposed in the literature. On the one hand, a variety of mathematical programming models have been presented in order to find the optimal solution. Corominas and Pastor (2009) formulated the optimization problem as a mixed-integer linear programming (MILP) model. Based on such formulation, two enhanced MILP models were proposed by Calleja *et al*. (2013) and instances up to a certain size were solved optimally. On the other hand, a different approach to the problem considers hybridizing heuristics and mathematical programming to solve the instances that are out of reach of the former models, which is presented in Calleja *et al*. (2014).

What emerges from the computational results on AWALBP-L2 (Calleja *et al.,* 2013, 2014) is that computing an optimal solution of the problem might become intractable for large size instances. For this reason, metaheuristic or hybrid solution methods could be envisioned to solve this problem. In the last few years, so-called hybrid optimization approaches have become increasingly popular for tackling complex optimization problems (Blum *et al*., 2011). One of the latest trends of hybridization is the interoperation of metaheuristics with mathematical programming techniques (Boschetti *et al*., 2009). In this line, the word *matheuristic* has been coined to indicate those solution approaches that exploit the complementary strengths of exact and (meta)heuristic components (Maniezzo *et al*., 2009). Manifold possibilities of hybridization within a matheuristic arise. According to their control strategy, such hybrids can be classified into integrative (coercive) and collaborative (cooperative) combinations (Puchinger and Raidl, 2005). In integrative combinations, one technique is considered as a subordinated, embedded component of another technique, following a master-slave scheme. Collaborative algorithms, in contrast, exchange information but are not part of each other.

In this paper, we propose three hybrid metaheuristics (or matheuristics, according to the aforesaid definitions) in which mathematical programming models are used in a metaheuristic frame - one based on simulated annealing (SA) and the other two based on tabu search (TS). The proposed approaches differ in the way the neighborhood is defined. More specifically, the two former methods utilize a classical move-based neighborhood, whereas the latter one makes use of the corridor method (CM) (Sniedovich and Voβ, 2006) to draw a *corridor* around the current solution via the imposition of exogenous constraints on the problem formulation. Furthermore, combined approaches of the aforementioned hybrids with a mathematical programming model are proposed.

The remainder of the paper is organized as follows: In Sections 2 and 3, we describe the AWALBP-L2 considered in this work and introduce the proposed hybrid metaheuristics, respectively. In Sections 4, 5 and 6, we detail the proposed hybrids based on SA, TS and TS with CM, respectively. In Section 7, we present combined approaches of the aforementioned hybrids with a mathematical programming model. Comparative experimental results of the proposed hybrid metaheuristics and the best in the literature are shown in Section 8. Finally, Section 9 presents some concluding remarks.

2. Problem specification

We consider the specific case of AWALBP-L2 described in Müller-Hannemann and Weihe (2006). The considered case can be stated as follows. An assembly line is given with a number *m* of workstations. Each workstation *i* has an accessibility window to the workpieces delimited by the interval $[L_i, R_i]$ of the assembly line such that $L_1 = 0$ and $R_i > L_i > R_{i-1}$ for $i = 2,...,m$. Therefore, the accessibility windows of the workstations do not overlap. Each task can be executed only on one given workstation. On each workstation *i*, a specified set of tasks J_i must be executed for each workpiece. The total number of tasks is denoted by $N = \left|\bigcup_{i=1}^{m} J_i\right|$. For each task j $(j = 1,...,N)$ the triple (p_j, a_j, m_j) is known, where p_j is the processing time of task j, a_j is the distance from the task position to the right border of the workpiece, and m_j is the workstation that has to execute this task. Then, the solution of the problem decomposes into:

i) a movement scheme $< x : \delta_1, \delta_2, ..., \delta_S >$, which consists of:
   - the initial position *x* of the workpieces on the line.
   - the number *S* of stationary stages (which coincides with the number *S* of forward steps).
   - the values $\delta_1, \ldots, \delta_S$ of the length of the forward steps, where δ_s is the number of elementary steps of the forward step s $(s = 1,...,S)$.

ii) for each task, an assignment to one stationary stage of the cycle where the position of the task is accessible for the station of this task.

To be feasible, a solution must hold the following conditions. First, the sum of all forward steps in a cycle must be equal to the distance *A* between two right (left) borders of two consecutive workpieces. Second, all forward steps must be a multiple of Δ (the elementary step). Finally, the third condition is that each task must be assigned to a stationary stage in which the task is accessible from its workstation.

The objective function (1) is the minimization of the cycle time (*CT*). Between two stationary stages, there is a time *T* to take into account the acceleration and deceleration of the line as well as the resetting of the robot arms. Then the total time of the cycle is equal to the sum of i) the time *T* multiplied by the number of stationary stages *S* plus ii) the time elapsed in the stationary stages constituting a cycle and iii) the time for transporting a workpiece through the assembly line at steady speed (since the latter is a constant it is not regarded for optimization purposes):

$$CT = T \cdot S + \sum_{s=1}^{S} C_s \qquad (1)$$

where $\sum_{s=1}^{S} C_s$ is the total processing time corresponding to all S stationary stages constituting a cycle, and C_s is the completion time, for the whole line, corresponding to the stationary stage s ($s = 1,...,S$).

3. The proposed hybrid metaheuristics

The proposed hybrid metaheuristics can be seen as integrative algorithms where the metaheuristic is used as the master mechanism to guide the search process and one mathematical model acts as an embedded slave.

Two types of neighborhood definitions are used in the search process. In the first type the neighborhood is defined by applying local changes or moves to a current movement scheme. We focus the search in the space of the movement schemes based upon the observation that the problem can be solved by using the following decomposition approach: i) generation of a movement scheme, and ii) assignment of each task to one stationary stage. The reasoning behind this decomposition is the following: if a movement scheme is computed first, then the optimal assignment of tasks to stationary stages, for the given movement scheme, can be obtained fast with a mathematical programming model, denoted *Task model* (Calleja *et al*. 2014) (see Annex). Therefore, the problem can be reduced to find an optimal movement scheme.

In contrast, in the second type the neighborhood is defined by building a *corridor* around a current solution in order to iteratively solve smaller portions of the target problem. More specifically, exogenous constraints are imposed on the original formulation of the problem and, subsequently, the constrained version is solved with a mathematical programming model.

4. Hybrid simulated annealing metaheuristic

Simulated annealing (SA) is a probabilistic optimization method which since its first introduction by Kirkpatrick *et al*. (1983), has been recognized as a simple yet powerful metaheuristic that provides excellent solutions to a wide variety of hard combinatorial optimization problems (Suman and Kumar, 2006).

Basically, SA is a local search procedure that tries to avoid being trapped in local optima by allowing probabilistically moves to worse solutions. The algorithm starts from an initial solution, which is initially the current solution *y*, and by initializing the value of a parameter *t* called *temperature*. Then, at each iteration, a solution *y'* from the neighborhood of the current solution *N(y)* is randomly selected. If the neighbor is not worse than the current solution, then the neighbor is accepted and replaces the current solution. In the case that it is worse, the neighbor can also be accepted, with a probability that depends on i) how much worse is the neighbor, and ii) the value of the temperature *t*. Initially, the algorithm starts at a high temperature *t* (that is, the probability of accepting deteriorating moves is high), which then gradually decreases and approaches zero. The number of iterations for which the temperature remains constant before being reduced is *itt*. The SA algorithm is presented in Fig. 3.

SA

Let *f(y)* be the objective function to be minimized of the solution *y*
Let *N(y)* be the neighborhood of the solution *y*
Let *A*(*t*) be a new temperature value obtained from the temperature *t*

1. Initialize the parameters:
   t_0 (initial temperature)
   itt (number of iterations during which the temperature remains constant)
2. $t := t_0$
3. *y*:= Generation of the initial solution
4. **while** the stopping criterion is not satisfied **do**
5. | **for** (*i*: = 0; *i* < *itt*; *i* := *i* +1):
6. | | *y'*:= randomly select *y'* from *N(y)*
7. | | **if** *f*(*y'*) ≤ *f*(*y*) **then** *y* := *y'*
8. | | **else** *y* := *y'* with a probability exp(-(*f*(*y'*)-*f*(*y*))/*t*)
9. | | **end**
10. | **end**
11. | *t* :=*A*(*t*)
12. **end**
13. **return** the best solution found

Figure 3. *General scheme of simulated annealing*

The proposed hybrid combines the general scheme of SA (Fig. 3) with the *Task model*. As mentioned in Section 3, the search is performed in the space of the movement schemes. In each iteration, the *Task model* is employed to compute the optimal cycle time for the current neighbor movement scheme, which provides a complete current solution for the problem. The obtained cycle time value determines whether the candidate movement scheme (along with its optimal task assignment) will be accepted or rejected as the new current solution in the SA local search.

The efficiency of the general scheme of SA depends on some key decisions. Some of these decisions are problem-specific, whereas some others are generic to SA. Specific decisions for the AWALBP-L2 include the definition of neighborhood of a solution (*N*(*y*)), and the generation of the initial solution. General decisions are the cooling schedule to decrease the temperature *A*(*t*) and the stopping criterion of the algorithm. In the following we outline such decisions.

4.1 Neighborhood of movement schemes

The proposed SA hybrid makes use of three neighborhood structures, N_1, N_2, and N_3, as follows. N_1 consists in transferring one elementary step from a forward step to another forward step. N_2 consists in inserting a new forward step by transferring one elementary step from an existing forward step to a new one. Finally, N_3 considers the neighbors obtained by varying the value of the initial position *x* in the interval $0 \le x \le \min\left(R_1 + a^1_{\min}, A - \Delta\right)$, where $a^1_{\min} = \min_{j \in J_1} a_j$. Note that in the two first neighborhood types, a forward step with only one elementary step may achieve length zero if its only elementary step is transferred (and thus such forward step disappears from the movement scheme). Therefore, the number of forward steps may vary. More specifically, it can remain equal or decrease in N_1, and it can remain equal or increase in N_2. Feasibility loss following transference or insertion of elementary steps can occur if the resulting movement scheme contains some tasks whose position is not accessible at any stationary stage. In any case, we consider only those neighbors which are feasible. At each iteration of the SA algorithm, it is

selected at random from which of the three neighborhoods a neighbor of the current movement scheme will be obtained. The values of the probabilities associated to the neighborhood selection are to be fine-tuned (see Section 8).

4.1 Initial solution

An initial solution is obtained by using the *Initial solution matheuristic* proposed in Calleja *et al*. (2014). It consists of i) an algorithm to generate, for a given value of x, a feasible movement scheme and ii) a mathematical model (the *Task model*), to compute the optimal assignment of tasks to stationary stages (for the generated movement scheme). The initial solutions obtained with this procedure appear to be of good quality and the necessary computational time is in average as small as a few milliseconds. Among the obtained solutions, computed for all values of x multiples of Δ, a solution with the minimum cycle time is identified. In case of having several solutions with the minimum cycle time, a solution with the minimum number of stationary stages is selected. The information given by the obtained initial solution is used to compute a lower bound on the value of the cycle time, $LB1^{CT}$, which is used to certificate whether a current solution is optimal (see Section 4.4). The bound $LB1^{CT}$ is computed as follows. Since the *Initial solution matheuristic* has been proven to provide solutions with the minimum number of stationary stages (see proof in Calleja *et al*. 2014), the solution with the minimum number of stationary stages among all those obtained with the mentioned matheuristic, Sol^S, gives a lower bound on the number of stationary stages, $LB1^S$. Then, we derive a lower bound on the cycle time, $LB1^{CT}$, by summing lower bounds on the two terms that compose the objective function (see Eq. (1)): (i) $T \cdot LB1^S$ plus (ii) a lower bound on the completion time of the stationary stages, which we name W_{max}, corresponding to the processing time of the most loaded workstation on the line. This is, $W_{max} = \max_{i=1,\ldots,m} \sum_{j \in J_i} p_j$.

4.2 Cooling schedule

The cooling schedule specifies how the temperature of the SA algorithm is decreased as the search progresses. We use geometric cooling, one of the most popular schedules used in the literature, that is, $A(t) = \alpha \cdot t$, where $0 < \alpha < 1$ (Downsland and Adenso-Díaz, 2003, Henderson *et al*., 2003). The value of the α parameter, as well as the initial temperature t_0 and the number of iterations during the temperature remains constant, *itt*, are to be fine-tuned, as explained in Section 8.

4.3 Stopping criterion

The algorithm stops when one of the following conditions is reached: i) a specified maximum time has elapsed, or ii) the objective function value of a solution coincides with $LB1^{CT}$ and thus the solution is proven optimal.

5. Hybrid tabu search metaheuristic

Tabu search (TS) is a metaheuristic originally proposed by Glover (1986) that has been successfully applied in many difficult combinatorial optimization problems (Glover, 1997, Pedersen *et al*., 2009). Like SA, TS can be seen as a local search that allows non-improving moves. The innovative idea of TS is the explicit use of memory structures, that record not only information about the current solution, but also information about the recent search trajectory followed to reach the current solution. Essentially, a TS algorithm moves at each iteration from

a solution y to a solution in its neighborhood $N(y)$, and may accept worse neighbors than the current solution. To prevent endless cycling and guide the search into unexplored areas, some formerly visited solutions, or attributes of them, are temporarily declared tabu or prohibited. The number of iterations that an attribute remains tabu is called its tabu tenure. The tabu status of a solution, though, can be overridden if a specified aspiration criterion is met; for example, if a tabu solution is better than the best solution found so far. The general TS algorithm is presented in Fig. 4. For a thorough presentation of the method, we refer the interested readers to Glover (1989, 1990) and Gendreau (2003).

TS

1. Define the neighborhood $N(y)$
2. Let y be an initial solution and $y^*:=y$
3. **while** the stopping criterion is not satisfied **do**
4. | Let y' be the best solution from $N(y)$ which is allowed by aspiration or is not tabu
5. | **if** y' is better than y^*, **then** $y^* := y'$ **end**
6. | Add the current move in the tabu list (removing its last move if it is full)
7. | $y := y'$
8. **end**
9. **return** y^*

Figure 4. *General scheme of tabu search*

The proposed hybrid TS relies on the general TS guidelines presented by Glover (1989, 1990), as shown in Fig. 4. As in the proposed SA-hybrid, we build the neighborhood around the movement schemes. Subsequently, the *Task model* is used to find an optimal assignment of tasks to stationary stages of the current movement scheme, which provides a complete current solution. A similar approach embedding a LP model in a probabilistic tabu search to solve a facility layout problem with unequal area departments has been proposed in Kulturel-Konak (2012). As in our paper, a mathematical programming model is used to evaluate the non-tabu solutions of the neighborhood of the current solution with the difference that instead of evaluating each and every element of the neighborhood, it considers only evaluating a random sample to reduce computational effort.

In our approach we consider the same initial solution generation, neighborhood structures and stopping criterion as in the proposed SA. The remainder elements of the proposed TS-based hybrid, i.e., the tabu lists, tabu attributes and aspiration criterion, are defined in the following subsections.

1.1 Tabu lists and tabu attributes

The tabu list is directly related to the neighborhood structure used to solve the problem. We consider the three neighborhoods N_1, N_2, and N_3 proposed in Section 4.1. In each iteration, the best neighbor movement scheme is searched within the three neighborhoods. Neighborhoods N_1 and N_2 are similar structures since they are both generated by transferring one elementary step to an existing or a new forward step. Neighborhood N_3, though, is a different structure based on the value of the initial position x. Therefore, we consider two different tabu lists, a first tabu list for the neighbors selected from N_1 or N_2, and a second tabu list for those selected from N_3, as follows. We call *transmitter forward step* the forward step which transfers one elementary step. Similarly, a *receiver forward step* is the one which receives an elementary step. Then, the first tabu list, T_1, contains attributes consisting of four elements: i) initial position value, ii) the number S of forward steps iii) the transmitter forward step s and its length δ_s, and iv) the

receiver forward step *s'* and its length $\delta_{s'}$. The second tabu list (T_2), though, contains only the two first aforementioned elements.

A numerical example is shown in Table 1. Let $< 5 : 2,3,6,2 >$ be a current movement scheme (column 1), with an initial position $x = 5$ and four forward steps with 2, 3, 6 and 2 elementary steps, respectively. Column 2 states the neighborhood type from which a neighbor will be generated. In the case of neighborhood N_2, two subtypes are distinguished: N_2(a), where an elementary step is inserted *between* the forward steps of the current movement scheme, and N_2(b), where the elementary step is inserted *after the last* forward step of the current movement scheme. Column 3 gives an example of a neighbor movement scheme obtained from each neighborhood type. Column 4 indicates in which tabu list, T_1 or T_2, the attribute will be recorded. Finally, Column 5 details the attribute to be stored in the tabu list when the neighbor movement scheme is set tabu. For example, if the best neighbor is < 5: 1, 4, 6, 2 >, which has been obtained from N_1, then the tabu attribute is $< x = 5, S = 4, \delta_1 = 2, \delta_2 = 3 >$ and it is added to the T_1 tabu list. In the case that the best neighbor belongs to N_2, the same tabu attribute and tabu list are considered. Note that, in N_2(b), the receiver forward step does not exist in the current movement scheme and thus the last element of the tabu attribute is considered as zero $(\delta_5 = 0)$. Finally, if the best neighbor is the one obtained from N_3 then the tabu attribute is $< x = 5, S = 4 >$ and it is added to the T_2 tabu list.

Current movement scheme	Neighborhood	Neighbor movement scheme	Tabu list type	Tabu attribute
< 5: 2, 3, 6, 2 >	N_1	< 5: 1, 4, 6, 2 >	T_1	$< x = 5, S = 4, \delta_1 = 2, \delta_2 = 3 >$
	N_2(a)	< 5: 2, 2, 1, 6, 2 >	T_1	$< x = 5, S = 4, \delta_2 = 3, \delta_3 = 6 >$
	N_2(b)	< 5: 2, 2, 6, 2, 1 >	T_1	$< x = 5, S = 4, \delta_2 = 3, \delta_5 = 0 >$
	N_3	< 7: 2, 3, 6, 2 >	T_2	$< x = 5, S = 4 >$

Table 1. *An example of the different neighbor movement schemes and their tabu attributes*

The lengths of tabu lists are to be fine-tuned, as explained in Section 8.

1.2 Aspiration criterion

We use the most commonly used aspiration criterion in the TS literature (Gendreau and Potvin, 2005) which allows a tabu move when it results in a solution better than the current best-known solution.

2. Hybrid tabu search - corridor method metaheuristic

A hybrid approach combining TS and a Corridor Method (CM) is presented next. The CM is a matheuristic introduced by Sniedovich and Voβ (2006), which intertwines mathematical programming techniques with metaheuristic features. The central idea of the CM relies on the iterative use of an exact method to solve optimally restricted portions of the solution space of a given problem. Such portions of the original space are defined by building a *corridor* around a current solution via the imposition of exogenous constraints.

The proposed hybrid TS-CM follows the scheme of the TS metaheuristic proposed by Glover (1989, 1990) (recall Fig. 4), as described in Section 5. However, the proposed hybrid TS-CM differs from the aforementioned one in the sense that now the neighborhoods are not defined via local changes or moves, but constructed by adding exogenous constraints onto an embedded MILP model. Such MILP model is subsequently used to solve the resulting portion of the problem space. Furthermore, additional constraints are also imposed in order to model the tabu lists and the aspiration criterion. An approach making use of TS as a master strategy and a branch and bound solver as an embedded mechanism for solving relaxed instances of the generalized assignment problem has been proposed by Woodcock and Wilson (2010). As in our paper, the authors use mathematical programming techniques to move from a current solution to a new one. However, to the best of our knowledge, we are not aware of any work in the literature hybridizing TS with CM in the way presented here.

The proposed procedure starts from an initial solution obtained with the *Initial solution matheuristic* of Calleja *et al*. (2014) and explores the neighboring solution space in search of an improving solution.

At each iteration, the MILP model receives a current solution as input. Based on this solution, bounds on the cycle time and the number of forward steps are computed and incorporated to the model. Next, the constrained version of the problem defined by the corridor is solved by using the MILP model within a limited computational time.

The overall algorithm terminates when one of the following criteria is reached: i) a maximum running time, or ii) the problem is solved to proven optimality since a solution is obtained whose objective function value coincides with the lower bound of the cycle time, $LB1^{CT}$.

6.1 *Tabu list and aspiration criterion*

A short-term memory structure is used as a tabu list, which stores the attributes of the movement schemes recently visited. More specifically, we propose a composite attribute for the tabu list, $< Tabu^{S}, Tabu^{x} >$, where $Tabu^{S}$ and $Tabu^{x}$ express, respectively, the number $\hat{S}$ of stationary stages and the initial position $\hat{x}$ of the movement scheme of the current solution. Then, the set of tabu attributes in a tabu list is defined by $\left\{< Tabu_h^{S}, Tabu_h^{x} >: h = 1,...,TT\right\}$, where TT is the number of attributes contained in the tabu list.

The proposed TS-CM uses the aspiration criterion described in Section 5.2, which overrides the tabu status of a solution if its objective function is better than that of the best-known solution so far.

2.2 The embedded corridor method

The basic elements of the CM are: a problem P generally belonging to the class of *NP*-hard problems, a very large feasible solution space Y and an exact method M capable of solving P to optimality if the size of the solution space is not too large. The CM imposes exogenous constraints on the original formulation of the problem in such a way that smaller manageable portions of the solution space are identified. These exogenous constraints define a *corridor*, i.e., a set of solutions, around a given current solution $\hat{y} \in Y$. The nature of the imposed constraints should be such that they are compatible with both the structure of the problem P and the method M used to solve them.

Let us assume that method *M* is a MILP model. One way to identify smaller manageable portions of the solution space to be explored with the model is to constrain the domains of the variables that are present in a current solution. In the following we outline how a CM can be applied to a MILP model. Let us suppose that we are given a current solution $\hat{y}$ with a number Ψ of decision variables $(\hat{y}_1, \hat{y}_2, ..., \hat{y}_\Psi)$. In order to impose constraints on the variable domains, we need to limit the distance between the value of a variable y_n and its current value $\hat{y}_n$. Therefore, a neighborhood around a current solution can be generated by drawing corridors as follows:

$$N(\hat{y}) := \{(y_1, y_2, ..., y_\Psi) \in Y : (\hat{y}_n - R_n) \le y_n \le (\hat{y}_n + R_n), n = 1, ..., \Psi\} \quad (2)$$

where R_n $(n = 1, ..., \Psi)$ is a parameter used to define the corridor width.

Equation (2) limits the solution space only to those solutions whose distance from the current solution, for each variable y_n, is not greater than a given maximum value R_n.

Finally, in order to incorporate the neighborhood definition to the original MILP formulation of the problem, the following constraints are imposed:

$$y_n \ge \hat{y}_n - R_n \qquad (n = 1, ..., \Psi) \quad (3)$$

$$y_n \le \hat{y}_n + R_n \qquad (n = 1, ..., \Psi) \quad (4)$$

At each iteration, constraints (3)-(4) are therefore imposed onto the original model, which is introduced in Section 6.4, and the new constrained version is solved by applying a suitable algorithm.

In the following we introduce the notation required to formulate a fitting model for the CM. Let us consider the movement scheme of the current solution with an initial position $\hat{x}$ and $\hat{S}$ forward steps of lengths $< \hat{\delta}_1, \hat{\delta}_2, ..., \hat{\delta}_{\hat{S}} >$. Let us suppose that we generate a corridor of width R^δ around each variable δ_s, such that $\delta_s \in \left[\hat{\delta}_s - R^\delta, \hat{\delta}_s + R^\delta\right]$. As a result of the transference of elementary steps in the generation of neighbor movement schemes, there either may be some forward steps which become empty (i.e., their length is zero and thus disappear) or some whose length is necessarily greater than zero (i.e., it is known a priori that they will exist in any neighbor movement scheme). Therefore, the actual number of variables δ_s that a neighbor solution will have is not known in advance. In order to model the number of variables needed, we define the following additional data.

Let *US* be the upper bound on the number of forward steps, empty or not, that may be generated for the feasible solutions contained in the corridor, while respecting the corridor width:

$$US = \min\left(A / \Delta, \hat{S} + \sum_{s=1}^{\hat{S}} \min\left(R^\delta, \hat{\delta}_s\right)\right) \quad (5)$$

In the proposed model, *US* is used to upper bound the number of variables δ_s, such that $s = 1, ..., US$.

By *ES* we denote the set of forward steps whose existence *can* be assured a priori in all the feasible solutions of the space delimited by the corridor:

$$ES = \left\{ s = 1,...,\hat{S} : \hat{\delta}_s - R^{\delta} \geq 1 \right\} \tag{6}$$

Let *NES* then be the set of forward steps whose existence *cannot* be assured a priori in all the feasible solutions of the space delimited by the corridor:

$$NES = \{1,...,US\} \setminus ES \tag{7}$$

Finally, we derive an upper bound on the number of non-zero forward steps, UB^S, that a neighbor movement scheme may have, by the sum of (i) the current number of forward steps, (ii) the total number of elementary steps that can be transferred by the forward steps of the current movement scheme whose values are greater than R^{δ}, and (iii) the total number of elementary steps that can be transferred by the forward steps of the current movement scheme whose values are greater than one but equal to or smaller than R^{δ}:

$$UB^S = \hat{S} + |ES| \cdot R^{\delta} + \sum_{s=1,...,\hat{S} \mid \hat{\delta}_s \leq R^{\delta}} \left(\hat{\delta}_s - 1 \right) \tag{8}$$

Fig. 5 depicts a numerical example for the computation of UB^S. Let us assume that $< 0 : 6,7,2,1 >$ is the movement scheme of the current solution $\left(\hat{x} = 0, \hat{S} = 4, \hat{\delta}_1 = 6, \hat{\delta}_2 = 7, \hat{\delta}_3 = 2, \hat{\delta}_4 = 1 \right)$ and that a corridor of width $R^{\delta} = 3$ is built around the values of the forward steps. The idea is to construct a neighbor movement scheme in such a way that the maximum number of non-zero forward steps is obtained. This can be done by transferring as many elementary steps as possible from each forward step in such a way that the latter keeps at least one elementary step. In the example of Fig. 5, such transfers generate a neighbor movement scheme with 11 forward steps, which gives the value for UB^S.

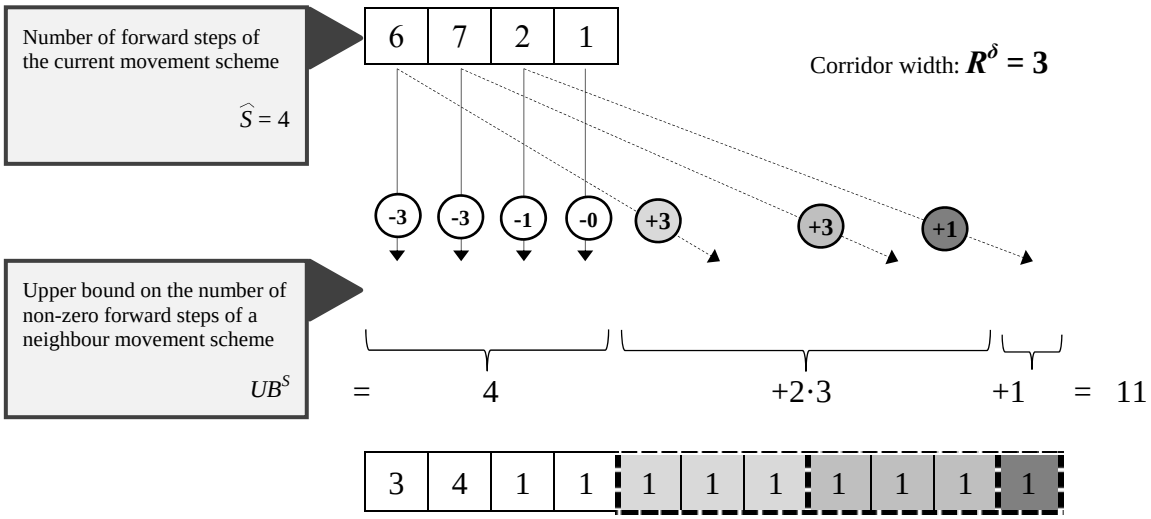

Figure 5. *Example of the computation of parameter UB^S*

In order to further illustrate the required notation let us consider a numerical example of a line with $\Delta = 1$ and $A = \sum_{s=1}^{\hat{S}} \Delta \cdot \hat{\delta}_s = 5$ (Table 2). Let us suppose that $< 0 : 4,1 >$ is the movement scheme of the current solution $\left(\hat{x} = 0, \hat{S} = 2, \hat{\delta}_1 = 4, \hat{\delta}_2 = 1\right)$. Let us assume that we build a corridor of width R^{δ} =1 around the forward step values of the movement scheme. The neighbor movement schemes admissible in this corridor are given in the second column of Table 2. Then, the neighbor movement schemes that can be generated inside a corridor of width R^{δ} =1 around the current forward steps may contain at most four forward steps (empty or not) and thus $US = 4$. As can be seen from the generated neighbors, there may be some forward steps whose length is necessarily greater than zero if the current length of the forward step is greater than the corridor width. In this example, the set of non-zero forward steps is $ES = \{1\}$. Conversely, the rest of forward steps of a neighbor movement scheme may be of length zero and thus $NES = \{2,3,4\}$. Finally, the upper bound on non-zero forward steps is $UB^S = 3$.

Current movement scheme	Neighbors within a corridor of width $R^{\delta} = 1$	Data values
<0: 4, 1 >	< 0: 5, 0, 0, 0 > < 0: 3, 2, 0, 0 > < 0: 3, 1, 1, 0 > < 0: 4, 0, 1, 0 > < 0: 4, 0, 0, 1 > < 0: 3, 0, 1, 1 > < 0: 3, 1, 0, 1 >	$US = 4$ $ES = \{1\}$ $NES = \{2,3,4\}$ $UB^S = 3$

Table 2. *A numerical example illustrating the values of parameters ES, NES, US and* UB^S

6.3 Definition of corridors

In order to apply a corridor around a current solution we need to select which variables of the current solution will be restricted. The width of such corridor will allow only for the exploration of those solutions that are at a maximum distance from a current one. More specifically, we apply corridors to some of the variables that define the movement scheme of a current solution. Several possibilities for the construction of corridors arise, depending on which variables are selected. Specifically, we consider three alternate corridors, denoted *C1*, *C2* and *C3*, which are explained next.

6.3.1. Corridor C1

Given a current movement scheme, corridor *C1* constructs a neighborhood around the variable $\delta_S \left(s = 1,...,US\right)$ by including all movement schemes whose forward steps have a length within a distance R^{δ} from the current lengths.

In the corridor, the number of forward steps is lower bounded by LB^S and thus the maximum number of elementary steps that a forward step may achieve is given by the expression $\left(A / \Delta - LB^S + 1\right)$.

6.3.2 Corridor C2

The second type of corridor, *C2*, builds a neighborhood around $\delta_S\ (s=1,...,US)$ and around the number of forward steps S.

An additional parameter, R^S, is added to express the corridor around the current number of forward steps $\hat{S}$. Consequently, the following data are modified:

$$LB^S = \max\left(\hat{S} - R^S, |ES|, LB1^S\right) \tag{9}$$

$$UB^S = \min\left(\hat{S} + R^S, \hat{S} + |ES| \cdot R^\delta + \sum_{s=1,...,\hat{S} \mid \hat{\delta}_s \le R^\delta} \left(\hat{\delta}_s - 1\right)\right) \tag{10}$$

where (9)-(10) define the corridor and then the expression for *US* is modified as follows:

$$US = \min\left(A/\Delta, \hat{S} + \left(\hat{S} - |ES|\right) + \min\left(R^S, \sum_{s=1}^{\hat{S}} \min\left(R^\delta, \hat{\delta}_s - 1\right)\right)\right) \tag{11}$$

6.3.3 Corridor C3

The third corridor structure considers the construction of a neighborhood around the lengths of the forward steps δ_s, the number of forward steps *S* and the initial shift *x*. Again, we consider an additional parameter, R^x, to express the width corridor around the variable *x*. The following parameters arise:

$$X^- = \left(\hat{x} - R^x\right) \bmod \left(\min\left(R_1 + a^1_{\min}, A - \Delta\right) + 1\right) \tag{12}$$

$$X^+ = \left(\hat{x} + R^x\right) \bmod \left(\min\left(R_1 + a^1_{\min}, A - \Delta\right) + 1\right) \tag{13}$$

Once the values X^- and X^+ have been defined, two cases may arise:

a) $X^- \le X^+$. The corridor around *x* is defined as: $X^- \le x \le X^+$

b) $X^- > X^+$. In this case, the corridor is a wrap-around interval where the values admitted for the variable *x* are:

$x \in \left\{X^+, X^+ + 1, X^+ + 2, ..., \min\left(R_1 + a^1_{\min}, A - \Delta\right), 1, 2, ..., X^-\right\}$.

To define the corridor, we introduce the variable $r \in \{0,1\}$ and the following constraints:

$$x \ge X^- - MR^x_1 \cdot r \tag{14}$$

$$x \le X^+ + MR^x_2 \cdot (1 - r) \tag{15}$$

where:

$$MR^x_1 = X^-$$

$$MR^x_2 = \min\left(R_1 + a^1_{\min}, A - \Delta\right) - X^+$$

6.4 *A MILP model for the corridor method*

In this section we present a MILP model, denoted *Solve-corridor*, to be used at each iteration of the proposed hybrid TS-CM. Such MILP model is used to define a neighborhood (incorporating corridors), and to obtain the best neighbor that is not tabu or fulfills the aspiration criterion. The model is inspired on the *Solve model* of Calleja *et al*. (2014). The new contributions to the formulation correspond to the imposition of exogenous constraints to define the corridor, and the addition of the tabu lists and the aspiration criterion.

In the following, we present the *Solve-corridor* MILP model, which includes a corridor of type *C1* around the variables δ_s $(s = 1,...,US)$. The proposed model can be easily adapted to include corridors of types *C2* and *C3*, by incorporating the modifications stated in sections 6.3.2 and 6.3.3, respectively.

Data

N number of tasks $(j = 1,...,N)$

m number of workstations $(i = 1,...,m)$

m_j workstation where task j has to be executed $(j = 1,...,N)$

$[L_i, R_i]$ accessibility window of workstation i $(i = 1,...,m)$, where $L_1 = 0$ and $R_i > L_i > R_{i-1}$, $(i = 2,...,m)$

A_0 workpiece's length

A distance between the right borders of two successive workpieces of the assembly line ($A > A_0$)

T time to take into account acceleration and deceleration between two consecutive stationary stages

Δ length of an elementary step

p_j processing time of task j $(j = 1,...,N)$

a_j $(0 \le a_j \le A_0)$, distance to the right border of the workpiece corresponding to the task j $(j = 1,...,N)$

J_0 set of tasks $(J_0 = \{1,2,...,N\})$

J_i set of tasks to be performed on workstation i, $J_i = \{j \in J_0 : m_j = i\}$, $(i = 1,...,m)$ where $\bigcup_{i=1,...,m} J_i = J_0$, $J_i \cap J_{i'} = \varnothing$, $\forall\, i, i' = 1,\dots,m \mid i \neq i'$

R^{δ} corridor width

$\hat{S}$ number of stationary stages of the TS current solution

$\hat{x}$ initial position of the workpiece with respect to the left limit of workstation 1 of the TS current solution

$\hat{\delta}_s$ number of elementary steps of the forward step s $\left(s = 1,...,\hat{S}\right)$ of the TS current solution

$$US = \min\left(A/\Delta, \hat{S} + \sum_{s=1}^{\hat{S}} \min\left(R^{\delta}, \hat{\delta}_s\right)\right)$$

$$ES = \left\{s = 1,...,\hat{S} : \hat{\delta}_s - R^{\delta} \geq 1\right\}$$

$$NES = \{1,...,US\} \setminus ES$$

LB^S lower bound on the number of stationary stages $\left(LB^S = \max\left(|ES|, LB1^S\right)\right)$, where $LB1^S$ is a lower bound obtained as described in Section 4.2

UB^S upper bound on the number of forward steps, $UB^S = \hat{S} + |ES| \cdot R^{\delta} + \sum_{s=1..\hat{S} | \hat{\delta}_s \leq R^{\delta}} \left(\hat{\delta}_s - 1\right)$

LB^{CT} lower bound on the cycle time $\left(LB^{CT} = T \cdot LB^S + W_{max}\right)$, where $W_{max} = \max_{i=1,...,m} \sum_{j \in J_i} p_j$

UB^{CT} upper bound on the cycle time, $UB^{CT} = T \cdot UB^S + \sum_{j=1}^{N} p_j$

CT^* cycle time of the best solution found so far within the TS

TT tenure or size of the tabu list

$Tabu_h^S, Tabu_h^x \left(h = 1,...,TT\right)$ define the list of tabu attributes (see Section 6.1): $\left\{< Tabu_h^S, Tabu_h^x >: h = 1,...,TT\right\}$

$kmin_j$ minimum number of times that a workpiece should be moved forward by A elementary steps such that task j is accessible in its workstation $(j = 1,...,N)$, where $kmin_j = \left\lceil \frac{L_{m_j} + a_j - \min(R_1 + a^1_{\min}, A - \Delta) - A + \Delta}{A} \right\rceil$, being $a^1_{\min}$ the closest distance of a task position j $\left(j \in J_1\right)$ to the right border of the workpiece ($a^1_{\min} = \min_{j \in J_1} a_j$)

$kmax_j$ maximum number of times that a workpiece should be moved forward by A elementary steps such that task j is accessible in its workstation $(j = 1,...,N)$, where $i \,|\, j \in J_i,\ kmax_j = \left\lfloor \frac{R_i + a_j}{A} \right\rfloor$

Variables

$x \in \mathbb{Z}^+$ initial position of the workpiece with respect to the left limit of workstation 1, where $0 \leq x \leq \min\left(R_1 + a^1_{\min}, A - \Delta\right)$

$\delta_s \in \mathbb{Z}^+$ number of elementary steps of the forward step s $\left(s = 1,...,US\right)$

$\eta_s \in \{0,1\}$ $\eta_s = 1$ *iff* the forward step *s* exists, $s \in NES$

$b_{jsk} \in \{0,1\}$ $b_{jsk} = 1$ *iff* task *j* is performed during stationary stage *s* after the workpiece has been moved forward *k* times by *A* elementary steps, $(j = 1,...,N; s = 1,...,US; k = kmin_j,...,kmax_j)$

C_s completion time, for the whole line, corresponding to the stationary stage *s* $(s = 1,...,US)$, where $\min_{j=1}^{N} p_j \le C_s \le W_{max}$ $(s \in ES)$ and $0 \le C_s \le W_{max}$, $s \in NES$

$y_{hg} \in \{0,1\}$ auxiliary variables $(h = 1,...,TT; g = 1,...,4)$. If the solution has the *h*-th tabu attribute in the tabu list, then the four variables y_{h1} , ..., y_{h4} take value 1.

$w \in \{0,1\}$ 1 *iff* the new solution fulfills the aspiration criterion

$u_1, u_2, v_{s1}, v_{s2} \in \{0,1\}$ auxiliary variables that are used to remove the current TS solution from the solution space $\left(s = 1,...,\hat{S}\right)$

Model

$$[MIN]\, z = T \cdot \left(|ES| + \sum_{s \in NES} \eta_s \right) + \sum_{s=1}^{US} C_s \tag{16}$$

$$LB^{CT} \le T \cdot \left(|ES| + \sum_{s \in NES} \eta_s \right) + \sum_{s=1}^{US} C_s \tag{17}$$

$$LB^{S} \le |ES| + \sum_{s \in NES} \eta_s \tag{18}$$

$$|ES| + \sum_{s \in NES} \eta_s \le UB^{S} \tag{19}$$

(a) movement scheme constraints

$$\hat{\delta}_s - R^{\delta} \le \delta_s \qquad s = 1,...,\hat{S} \tag{20}$$

$$\delta_s \le \min\left(\hat{\delta}_s + R^{\delta}, A/\Delta - LB^{S} + 1 \right) \qquad s \in ES \tag{21}$$

$$\delta_s \le \min\left(\hat{\delta}_s + R^{\delta}, A/\Delta - LB^{S} + 1 \right) \cdot \eta_s \qquad s = 1,...,\hat{S} : s \in NES \tag{22}$$

$$\delta_s \le \min\left(R^{\delta}, A/\Delta - LB^{S} + 1 \right) \cdot \eta_s \qquad s = \hat{S} + 1,...,US \tag{23}$$

$$\Delta \cdot \sum_{s=1}^{US} \delta_s = A \tag{24}$$

$$\delta_s \ge \eta_s \qquad s \in NES \tag{25}$$

(b) accessibility constraints

$$A \cdot k - a_j + x + \Delta \cdot \sum_{l=1}^{s-1} \delta_l \geq L_i - M_{js} \cdot (1 - b_{jsk})$$
$$s = 1,...,US; j = 1,...,N; k = kmin_j,...,kmax_j \tag{26}$$

$$A \cdot k - a_j + x + \Delta \cdot \sum_{l=1}^{s-1} \delta_l \leq R_i - M'_{js} \cdot (1 - b_{jsk})$$
$$s = 1,...,US; j = 1,...,N; k = kmin_j,...,kmax_j \tag{27}$$

where:

$$M_{js} = L_{m_j} - A \cdot kmin_j + a_j - \Delta \cdot (s-1)$$
$$M'_{js} = A \cdot \left(kmax_j + 1\right) + \min(R_1 + a^1_{\min}, A - \Delta) - R_{m_j} - a_j$$

(c) task assignment constraints

$$\sum_{s=1}^{US} \sum_{k=kmin_j}^{kmax_j} b_{jsk} = 1 \qquad j = 1,...,N \tag{28}$$

$$\sum_{j \in J_i} p_j \cdot \sum_{k=kmin_j}^{kmax_j} b_{jsk} \leq C_s \qquad i = 1,...,m; s = 1,...,US \tag{29}$$

$$\sum_{j \in J_i} \sum_{k=kmin_j}^{kmax_j} b_{jsk} \leq \left|J_i\right| \cdot \eta_s \qquad i = 1,...,m; s \in NES \tag{30}$$

$$\sum_{j=1}^{N} \sum_{k=kmin_j}^{kmax_j} b_{jsk} \geq \eta_s \qquad s \in NES \tag{31}$$

$$\sum_{j=1}^{N} \sum_{k=kmin_j}^{kmax_j} b_{jsk} \geq 1 \qquad s \in ES \tag{32}$$

(d) TS constraints

$$\left|ES\right| + \sum_{s \in NES} \eta_s \leq Tabu_h^S - 1 + M_{h1} \cdot y_{h1} \qquad h = 1,...,TT \tag{33}$$

$$\left|ES\right| + \sum_{s \in NES} \eta_s \geq Tabu_h^S + 1 - M_{h2} \cdot y_{h2} \qquad h = 1,...,TT \tag{34}$$

$$x \leq Tabu_h^x - 1 + M_{h3} \cdot y_{h3} \qquad h = 1,...,TT \tag{35}$$

$$x \geq Tabu_h^x + 1 - M_{h4} \cdot y_{h4} \qquad h = 1,...,TT \tag{36}$$

$$\sum_{g=1}^{4} y_{hg} \leq 3 + w \qquad h = 1,...,TT \tag{37}$$

$$T \cdot \left(\left|ES\right| + \sum_{s \in NES} \eta_s\right) + \sum_{s=1}^{US} C_s + 1 \leq CT^* + M_1^A \cdot (1 - w) \tag{38}$$

$$T\cdot\left(\left|ES\right|+\sum_{s\in NES}\eta_s\right)+\sum_{s=1}^{US}C_s\geq CT^*-M_2^A\cdot w \tag{39}$$

where $(h=1,...,TT)$:

$M_{h1}=UB^S-Tabu_h^S+1$

$M_{h2}=Tabu_h^S+1-LB^S$

$M_{h3}=\min\left(R_1+a_1^{\min},A-\Delta\right)-Tabu_h^x+1$

$M_{h4}=Tabu_h^x+1$

$M_1^A=UB^{CT}+1-LB^{CT}$

$M_2^A=CT^*-LB^{CT}$

$$\delta_s\leq\hat{\delta}_s-1+M_{s1}^{\delta}\cdot v_{s1} \qquad s=1,...,\hat{S} \tag{40}$$

$$\delta_s\geq\hat{\delta}_s+1-M_{s2}^{\delta}\cdot v_{s2} \qquad s=1,...,\hat{S} \tag{41}$$

$$x\leq\hat{x}-1+M_1^x\cdot u_1 \tag{42}$$

$$x\geq\hat{x}+1-M_2^x\cdot u_2 \tag{43}$$

$$u_1+u_2+\sum_{s=1}^{\hat{S}}\left(v_{s1}+v_{s2}\right)\leq 2\cdot\hat{S}+1 \tag{44}$$

where $(s=1,...,\hat{S})$:

$M_{s1}^{\delta}=\min\left(\hat{\delta}_s+R^{\delta},\left(A/\Delta\right)-LB^S+1\right)-\hat{\delta}_s+1$

$M_{s2}^{\delta}=\hat{\delta}_s+1-\max\left(\hat{\delta}_s-R^{\delta},0\right)$

$M_1^x=\min\left(R_1+a_{\min}^1,A-\Delta\right)-\hat{x}+1$

$M_2^x=\hat{x}+1$

The model captures the following features: The objective (16) is the minimization of the cycle time. Constraint (17) introduces a lower bound on the value of the objective function; (18) and (19) lower and upper bound, respectively, the number of the existing forward steps; (20-23) define the corridor, (24) states that the distance covered in the forward steps of a cycle corresponds to the distance between the right borders of two consecutive workpieces on the line; (25) forbid null forward steps by imposing that, if the number of elementary steps is zero, then the associated forward step *s* does not exist; constraints (26)-(27) guarantee that each task is accessible from the only station that is able to perform it, during the stationary stage in which the task will be executed; (28) impose that each task is assigned to one, and only one, stationary stage; (29), that the time corresponding to the stationary stages is not less than the processing time at any station; (30) avoid assigning a task to a non-existing stationary stage; (31)-(32) force that at least one task has to be assigned for each stationary stage; Constraints (33)-(37) represent the tabu constraints, in such a way that if the current movement scheme has the *h*-th tabu attribute, all the associated binary variables y_{hg} will have value 1. Such constraints prevent

moving to a solution that is marked tabu and is not allowed by the aspiration level; (38)-(39) express the aspiration criterion, so that the binary variable w has value 1 if and only if the solution fulfills the aspiration criterion; finally, constraints (40)-(44) remove the current movement scheme from the solution space of the mathematical model.

3. Combinations of hybrid metaheuristics and MILP

To further improve the quality of the solution of the AWALBP-L2, we propose combining the use of the MILP *Solve model* of Calleja *et al*. (2014) with the afore-presented hybrid metaheuristics. This model requires an initial solution in order to compute bounds. We generate an initial solution by using the *Initial solution matheuristic*, and next we search for an improving solution by combining the use of the *Solve model* and the hybrid metaheuristic in two alternative ways: i) using the *Solve model* with the initial solution obtained with the *Initial solution matheuristic* and then trying to improve the solution obtained by the model using one of the proposed hybrids. Or ii) executing one hybrid and then using the obtained solution as the initial solution for the *Solve model*.

Initial solution matheuristic	+	*Solve model*	900 s 1800 s 2700 s	+	SA-, TS-, or TS-CM- hybrid	
		SA-, TS-, or TS-CM- hybrid		+	*Solve model*	900 s 1800 s 2700 s

Table 3. *Combinations of the proposed hybrids and MILP*

Table 3 illustrates the considered combinations of hybrids and MILP. The two rows show, respectively, the two combinations types considered. In the first row MILP is applied before the hybrid, whereas in the second row MILP is applied after. In each combination type, we consider limiting the running time of the model to 900, 1800 or 2700 s, whereas the *Initial solution matheuristic* and the hybrid are executed in the remaining run time until a 3600 s total limit is reached. In any case, the proposed combined approach stops before the limit time if a solution with an objective function value equal to the lower bound on the cycle time, $LB1^{CT}$, is found.

4. Computational results

We present comparative results for the proposed hybrid metaheuristics and the existing results in the literature, namely the approach using the *Initial solution matheuristic* and the *Solve model* of Calleja *et al*. (2014). We aim to examine, in particular, the effectiveness of the proposed hybrids in finding high-quality solutions for large instances of the problem.

The hybrid procedures were implemented in Java and the mathematical models were solved using IBM ILOG CPLEX 12.2. The absolute optimality gap was set to $1\text{-}10^{-6}$ since, without loss of generality, all data are integers and thus the objective function value is also an integer. Experiments were performed in Intel Core 3.33 GHz workstations with 4 GB of RAM operating under Windows-7 (64 bits).

The performance of the proposed hybrids was tested on the same set of 1200 problem instances as in Calleja *et al*. (2013, 2014). These benchmarking instances can be downloaded from https://www.ioc.upc.edu/EOLI/research/. With respect to the difficulty in solving the problem, the most influential parameters are A_0, m and N, where A_0 is the workpiece length, m is the number of workstations in the assembly line and N is the number of tasks. For this reason, in the considered instances the mentioned parameters are distributed along the following ranges: A_0={11-15, 16-20, 21-25, 26-30, 31-35, 36-40}, m = {5-10, 11-20, 21-30, 31-40} and N={50-200, 201-400, 401-600, 601-800, 801-1000}. Additionally, the instances have the following characteristics. The width of the accessibility windows is 10 length units (lu) and the length of the elementary step Δ is 1 lu. The time T is 200 time units (tu). The processing time of tasks was randomly generated between 100 and 150 tu. The positions of tasks were also randomly generated along the workpiece length A_0. The distance between two consecutive workpieces in the line is 1 lu and thus $A = A_0 + 1$.

In order to test the quality of the proposed approaches, we carried out the following experiments. Firstly, each hybrid metaheuristic was tested alone. In the remaining of this section, we denote by *SA*, *TS* and *TS-CM* the hybrid metaheuristics based on simulated annealing, tabu search and tabu search with corridor method, respectively. Secondly, the combinations of hybrids and the MILP *Solve model* were also tested. By $MILP_{\text{time}}$+*Hybrid* we denote the combination type where the *Solve model* is executed before the hybrid, with $MILP_{\text{time}} \in \{900, 1800, 2700\}$ seconds and *Hybrid* $\in$ {*SA*, *TS*, *TS-CM*}. Accordingly, *Hybrid*+$MILP_{\text{time}}$ denotes the combination where the model is executed after the hybrid. All experiments are compared with respect to be best existing approach in the literature obtained in Calleja *et al*. (2014).

The values of the algorithmic parameters used in the implementation of the hybrids *SA* and *TS* were set based on computational experiments applying CALIBRA (Adenso-Díaz and Laguna, 2006), a systematic procedure used in the literature to find the best parameter values associated with heuristic or metaheuristic algorithms. Since the small to medium-size instances of AWALBP-L2 are not much sensitive to values of algorithmic parameters, we generated a training set of 48 large to very large-scale instances. The set was created by generating 2 instances for each of the 24 combinations of the following ranges: A_0 = {31-35, 36-40}, m = {5-10, 11-20, 21-30, 31-40}, and N= {401-600, 601-800, 801-1000}. The obtained parameters values are the following. As for *SA*, the values of the α, *itt* and t_0 parameters are 0.9875, 1400 and 115, respectively, and the probabilities associated to selection from neighborhoods N_1, N_2 and N_3 are 0.75, 0.1 and 0.15, respectively. As for *TS*, the tabu tenures are 24 for the tabu list associated to neighborhoods N_1 or N_2, and 8 for the tabu list associated to neighborhood N_3.

A preliminary test was carried out to examine the influence of the parameters of *TS-CM* on different instances sizes of the data set, being the very large instances the most sensitive.

Therefore, a set containing the 20 largest instances of the problem was used to test the performance of the three corridor structures *C1*, *C2* and *C3* proposed in Section 6.4, for different values of the tabu tenure (5, 10, 20) and corridor widths $\left(R^{\delta}=1,3,5; R^{S}=1,2; R^{x}=1,2\right)$. Thus 9 combinations of values were tested for *C1*, 18 for *C2* and 36 for *C3*, and *C3* provided the best performance in terms of the improvement of the objective function with respect to the initial solution. We therefore select the combination of *C3* that yielded the best results to be used in the *TS-CM* hybrid. Such combination has the following parameters values: the tabu tenure is 5 and the corridor widths around the forward steps, the number of stationary stages and the initial shift are, respectively, $R^{\delta}=3$, $R^{S}=1$ and $R^{x}=1$. Finally, we set the run time limit of a *TS-CM* iteration to 300 s.

Among the 1200 initial solutions obtained with the method proposed in Calleja *et al*. (2014), 457 initial solutions (38.08%) yielded an objective function value coincident with the computed lower bound on the cycle time, ($LB1^{CT}$), and thus were certified as optimal solutions. We therefore focus on the comparative results for the remaining 743 instances.

Table 4 displays comparative results for the proposed hybrids with respect to Calleja *et al*. (2014) on the 743 instances considered. Among these 743 instances, we know 519 optimal solutions obtained with all the methods tested so far (including the ones presented in this paper), which are used for optimality verification in the proposed methods. In the table, the results are grouped in four main rows. The first row shows the results corresponding to the best existing method of the literature (the *Solve model* of Calleja *et al*. (2014)) and the remaining rows, the results for *SA*, *TS*, *TS-CM* and their combinations with the *Solve model*. For each experiment, in the first column (*% optima/Method*) we provide the percentage of instances that were certified optimal by the method. The second column (*% optima/Known optimal solutions*) provides the percentage of instances that were certified optimal by comparing the obtained solution with the known optimal solutions. Finally, the third column (*% ave. GAP*) gives the average relative gap of the 743 instances considered, with respect to the best lower bound available. The relative gap is defined to be $\left(\left(|BF|-|BB|\right)/|BF|\right)\cdot 100$, where the best found value *BF* is the objective function value of the solution found by the procedure and the best bound value *BB* is the best bound known on the instance's solution. The value of *BB* is the maximum value among the following: i) the best bound computed by CPLEX among the models of (Calleja *et al*. 2013, 2014), ii) the theoretical lower bound on the cycle time, $LB1^{CT}$, proposed in (Calleja *et al*. 2014) and iii) the best bound computed by CPLEX among the *Hybrid*+$MILP_{\text{time}}$ experiments.

What can easily be inferred from these results is that, in terms of percentage of optimal solutions, a better performance is obtained when the proposed hybrids are combined with MILP than when executed alone (in all cases for *SA* and *TS*, and in 10 out of 12 cases for *TS-CM*). Specifically, the overall best optimality percentage was found using the combination *TS*+$MILP_{2700}$ (67.03 for optima certified by the method itself) and $MILP_{2700}$+*SA* (69.45 for optima certified by comparison with the known optima). On the other hand, a better relative gap percentage is obtained for six different procedures (1.83; among these three procedures $MILP_{900}$+*TS-CM* provided the best result -69.31- in terms of % of optimal solutions).

	% optima		% ave. GAP
	Method	**Known optimal solutions**	
Calleja et al. (2014)	63.66	65.68	2.56
SA	31.22	56.53	2.99
***SA* + *MILP*$_{900}$**	62.72	66.89	2.31
***SA* + *MILP*$_{1800}$**	64.47	67.70	2.25
***SA* + *MILP*$_{2700}$**	66.22	67.83	2.20
MILP*$_{900}$ + *SA	61.78	68.37	2.12
MILP*$_{1800}$ + *SA	63.80	68.64	2.05
MILP*$_{2700}$ + *SA	65.41	69.45	1.99
TS	30.82	67.29	1.88
***TS* + *MILP*$_{900}$**	64.74	68.78	1.83
***TS* + *MILP*$_{1800}$**	65.81	68.78	1.83
***TS* + *MILP*$_{2700}$**	67.03	68.91	1.83
MILP*$_{900}$ + *TS	61.78	68.78	1.87
MILP*$_{1800}$ + *TS	63.93	69.04	1.88
MILP*$_{2700}$ + *TS	65.01	68.78	1.92
TS-CM	30.96	68.78	1.83
***TS-CM* + *MILP*$_{900}$**	64.60	69.18	1.83
***TS-CM* + *MILP*$_{1800}$**	65.55	69.18	1.86
***TS-CM* + *MILP*$_{2700}$**	65.95	68.64	1.98
MILP*$_{900}$ + *TS-CM	62.05	69.31	1.83
MILP*$_{1800}$ + *TS-CM	63.93	68.91	1.88
MILP*$_{2700}$ + *TS-CM	64.47	68.24	2.01

Table 4. *Average results for the proposed experiments*

We focus on the best results obtained in terms of percentage of optimal solutions certified with known optima ($MILP_{2700}$+*SA*) and average relative gap ($MILP_{900}$+*TS-CM*). Table 5 summarizes the most relevant results for $MILP_{2700}$+*SA* (column 1) and for $MILP_{900}$+*TS-CM* (column 2) compared to the best existing results of Calleja *et al*. (2014) (column 3). The first row (*% equal CT*) shows the percentage of solutions which provided the same objective function value as in Calleja *et al*. (2014). Row 2 (*% improvement*) shows the percentage of instances that outperform the solution of Calleja *et al*. (2014). Rows 3 (*ave*.) and 4 (*max*.) show, respectively, the average and the maximum improvement among such instances. Conversely, the percentage of solutions that worsen the objective is given in row 5 *(% decrease*), and its average and maximum worsening values are shown in rows 6 (*ave*.) and 7 (*max*.), respectively. In both experiments, results show a high percentage of instances that equal (around 75%) or improve (by 25%) the objective function value of Calleja *et al*. (2014), whereas the percentage of instances that worsen the objective function value is kept low (between 0.67% and 1.08% for $MILP_{2700}$+*SA* and for $MILP_{900}$+*TS-CM*, respectively). In row 8 we examine the average gap (*% ave. GAP*) of the 743 instances considered with respect to the best lower bound available, which decreased from 2.56% to 1.99% in $MILP_{2700}$+*SA* and to 1.83% in $MILP_{900}$+*TS-CM*. Additionally, in row 9 we examine the maximum gap (*% max. GAP*), which remained equal in $MILP_{2700}$+*SA* but decreased to 22.05% in $MILP_{900}$+*TS-CM*. Finally, the percentage of optimal solutions, obtained by comparison with known optima (within the 743 instances considered) is shown in row 10 (*% total optima*), which increased from 65.68% to 69.31% in $MILP_{900}$+*TS-CM* and rose to 69.45% in $MILP_{2700}$+*SA*.

	$MILP_{2700}$ + SA	$MILP_{900}$+TS-CM	Calleja et al. (2014)
% equal CT	76.18	73.76	-
% improvement	23.15	25.17	-
ave.	2.69	3.15	-
max.	10.41	11.81	-
% decrease	0.67	1.08	-
ave.	1.87	1.39	-
max.	3.51	4.03	-
% ave. GAP	1.99	1.83	2.56
% max. GAP	23.12	22.05	23.12
% total optima	69.45	69.31	65.68

Table 5. *Computational results for $MILP_{2700}$ +SA and $MILP_{900}$+TS-CM with respect to Calleja et al. (2014)*

In order to assess the overall solution of the AWALBP-L2, we compare the results of the complete set of 1200 instances with respect to those obtained in Calleja *et al*. (2014). Specifically, the percentage of optimal solutions rose from 78.75% to 81.08% in $MILP_{2700}$+*SA*, and to 81.00% in $MILP_{900}$+*TS-CM*.

Finally, if we consider the optima obtained among all the proposed methods to date, we obtain that, the problem has been solved optimally for 81.33% of the instances.

5. Conclusions and perspectives

In this paper, we have presented three hybrid metaheuristics, based on simulated annealing, tabu search and tabu search with corridor method, to solve the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2) for the case where each task can only be performed in one workstation.

The proposed hybrids use a mathematical model in a metaheuristic frame. More precisely, the proposed hybrids follow a metaheuristic mechanism to guide the search and iteratively use an embedded mathematical model. While the hybrid SA and TS metaheuristics deploy move-based neighborhoods, the hybrid TS-CM features neighborhoods that are constructed within the mathematical model used to explore them. We have presented a hybrid metaheuristic where a tabu search is used to guide a MILP model over reduced portions of the original solution space. Borrowing the basic idea of the Corridor Method, such portions are defined by building corridors around a current solution, via the imposition of exogenous constraints. The resulting constrained version of the problem is then solved with the MILP model. To the best of our knowledge, this is the first time in the literature that such TS-CM hybridization is presented.

The performance of the proposed hybrids has been tested in an extensive computational experiment. They have been tested alone and in combination with a bounded mathematical programming model. The best result, in terms of percentage solutions certified with known optima, was obtained for a combination where the model is executed first and then the obtained solution is tried to be improved by a SA. Such alternative currently stands as the best method proposed for the AWALBP-L2.

The fundamental ideas on which the proposed hybrids are inspired are open in nature and extend interesting perspectives in combining mathematical programming with a metaheuristic

framework, either for improving the solutions of the problem presented here or for tackling other combinatorial problems.

References

Adenso-Díaz, B, & Laguna, M. (2006). Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54, 99-114.

Battaïa, O., & Dolgui, A. (2012). Reduction approaches for a generalized assembly line balancing problem. *Computers and Operations Research*, 39, 2337-2345.

Blum, C., Puchinger, J., Raidl, G., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11, 4135-4151.

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

Boschetti, M., Maniezzo, V., Roffilli, M. & Bolufé, A. (2009). Matheuristics: Optimization, Simulation and Control. In: *Lecture Notes in Computer Science*, 5818, 171-177. *Hybrid Metaheuristics - 6th International Workshop, HM 2009*, *Proceedings*.

Calleja, G., Corominas, A., García-Villoria, A., & Pastor, R. (2013). A MILP model for the Accessibility Windows Assembly Line Balancing Problem (AWALBP). *International Journal of Production Research,* 51, 3549-3560.

Calleja, G., Corominas, A., García-Villoria, A., & Pastor, R. (2014). Combining matheuristics and MILP for the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2). *Computers and Operations Research,* 48, 113-123.

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

Corominas, A. & Pastor, R. (2009). A MILP model for the Visibility Windows Assembly Line Balancing Problem (VWALBP): the case of the Müller-Hannemann & Weihe problem. Technical report. Universitat Politècnica de Catalunya. Available from: http://upcommons.upc.edu/e-prints/bitstream/2117/7047/1/IOC-DT-P-2009-09.pdf.

Corominas, A., Ferrer, L., & Pastor, R. (2011). Assembly line balancing: general resource-constrained case. *International Journal of Production Research*, 49 (12), 3527-3542.

Downsland, K.A., & Adenso-Díaz, B. (2003). Heuristic design and fundamentals of the Simulated Annealing. *Inteligencia Artificial*, 19, 93-102.

Gaudlitz, R. (2004). Optimization algorithms for complex mounting machines in PC board manufacturing. Technical University of Darmstadt, Germany. Diploma thesis.

Gendreau, M. (2003). An introduction to Tabu Search. Chapter 2 in Handbook of metaheuristics, Eds. Glover & Kochenberger, Kluwer Academic Publishers, 37-54.

Gendreau, M., & Potvin, J. Y. (2005). *Tabu Search.* Chapter 6 in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques*. Eds. Burke and Kendall, Kluwer Academic Publishers, 165-186.

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

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*, 20, 4-32.

Glover, F. (1997). Tabu Search and Adaptive Memory Programming – Advances, Applications and Challenges". Chapter 1 in Interfaces in Computer Science and Operations Research. Eds. R.S.Barr, R.V.Helgason, and J.L. Kennington (eds), Kluwer, 1-75.

Henderson, D., Jacobson, S.H., & Johnson, A.W. (2003). The Theory and Practice od Simulated Annealing". Chapter 10 in *Handbook of Metaheuristics*, Eds. Glover and Kochenberger, Kluwer Academic Publishers, 287-319.

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

Kulturel-Konak, S. (2012). A linear programming embedded probabilistic tabu search for the unequal-area facility layout problem with flexible bays. *European Journal of Operational Research*, 223(3), 614-625.

Maniezzo, V., Stützle, T. & Voβ, S. (2009). Matheuristics: Hybridizing metaheuristics and mathematical programming. *Annals of Information Systems*, 10, Springer.

Martino, L. & Pastor, R. (2010). Heuristic procedures for solving the general assembly line balancing problem with setups. *International Journal of Production Research*, 48 (6), 1787-1804.

Müller-Hannemann, M. & Weihe, K. (2006). Moving policies in cyclic assembly line scheduling. *Theoretical Computer Science*, 351, 425-436.

Pedersen, M.B., Crainic, T.G., & Madsen, O.B.G. (2009). Models and Tabu Search Metaheuristics for Service Network Design with Asset-Balance Requirements. *Transportation Science*, 43, 158-177.

Puchinger, J. & Raidl, G. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. *Artificial Intelligence and Knowledge Engineering Applications: A Bionspired Approach in Lecture Notes in Computer Science*, 3562, 41-53.

Sniedovich, M. & Voβ, S. (2006). The corridor method: a dynamic programming inspired metaheuristic. *Control and Cybernetics*, 35, 551-578.

Sternatz, J. (2014). Enhanced multi-Hoffmann heuristic for efficiently solving real-world assembly line balancing problems in automotive industry. *European Journal of Operational Research*, 235, 740-754.

Suman, B., & Kumar, P. (2006). A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, 57, 1143-1160.

Tuncel, G., & Topaloglu, S. (2013). Assembly line balancing with positional constraints, task assignment restrictions and station paralleling: A case in an electronics company. *Computers & Industrial Engineering*, 64, 602-609.

Woodcock, A.J., & Wilson, J.M. (2010). A hybrid tabu search/branch & bound approach to solving the generalized assignment problem. *European Journal of Operational Research*, 207, 566-578.

Annex

The *Task model*

Once a feasible movement scheme has been obtained (in which each task is accessible from its workstation in at least one of the stationary stages), this movement scheme can be used as an input for an ILP model (the *Task model*) which optimally assigns each task to one of stationary stage (Calleja *et al*., 2014). The complete *Task model* is given next.

Data

m number of workstations $(i = 1,...,m)$

N number of tasks $(j = 1,...,N)$

J_0 set of tasks $(J_0 = \{1,2,...,N\})$

J_i set of tasks to be performed in workstation i, where $\bigcup_{i=1,..,m} J_i = J_0$ and $J_i \cap J_{i'} = \varnothing$ $(i = 1,\ldots,m; i' = 1,...,m; i \neq i')$

p_j processing time of task j $(j = 1,...,N)$

S number of forward steps in a cycle

Π_j set of stationary stages where task j is accessible from the workstation where it can be performed $(j = 1,...,N)$.

Variables

$y_{js} \in \{0,1\}$ $y_{js} = 1$ *iff* task j is performed in the stationary stage s $(j = 1,..,N; s \in \Pi_j)$

C_s completion time corresponding to the stationary stage s $(s = 1,...,S)$

Model

$$[MIN]\ z = \sum_{s=1}^{S} C_s \tag{45}$$

$$\sum_{s \in \Pi_j} y_{js} = 1 \qquad j = 1,...,N \tag{46}$$

$$\sum_{j \in J_i \,|\, s \in \Pi_j} p_j \cdot y_{js} \leq C_s \qquad i = 1,...,m;\ s = 1,...,S \tag{47}$$

The objective (45) is the minimization of the completion time of the stationary stages. Constraints (46) impose that each task is assigned to one, and only one, stationary stage, and (47) ensure that the time corresponding to the stationary stages is not less than the processing time at any workstation.

Note: This manuscript is a corrected version of an in-review article. A mistake was detected and corrected during the revision process of the manuscript. Consequently, in this document Tables 4 and 5 have been accordingly modified, resulting in slightly different percentages (the corrected values differ at most by four tenth of a per cent with respect to the in-review version of the article).

A2.2. Communications to international conferences

Exact and heuristic approaches for the Visibility Windows Assembly Line Balancing Problem (VWALBP)

In proceedings of the 12th Annual Congress of the French National Society of Operations Research and Decision Science, 12è Congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2011), 583-584, Saint-Étienne, France, 2-4 March, 2011.

Exact and Heuristic Approaches for the Visibility Windows Assembly Line Balancing Problem (VWALBP)

Gema Calleja[1], Albert Corominas[1], Alberto García-Villoria[1], Rafael Pastor[1]

[1] IOC Research Institute
Universitat Politècnica de Catalunya (UPC)
Av. Diagonal 647, 11th floor , 08028, Barcelona, Spain
{gema.calleja,albert.corominas,alberto.garcia-villoria,rafael.pastor}@upc.edu

Keywords: Assembly line balancing, mixed-integer linear programming, matheuristics.

1. Introduction

As part of production systems, throughput optimization of assembly lines has attracted great research attention to accomplish the real-world problems related to them.

In this paper, we consider the problem that we name Visibility Windows Assembly Line Balancing Problem (VWALBP) [1], which arises in some actual automated production lines. In contrast to traditional assembly lines, the length of the workpieces may be larger than the visibility windows of the workstations, and because of this, only a limited portion of the unit can be reached from any station at any time.

The aim of this work is to solve the VWALBP, which was originally stated in Müller-Hannemann&Weihe [2]. For a set of sample instances, we first try to solve the problem optimally using a Mixed Integer Linear Programming (MILP) model. Since this problem is known to be NP-Hard, we expect that increasingly large-scale instances may lead to prohibitive computational times. An extensive effort is being made to explore the input size that the model is able to solve. Second, we propose heuristic and matheuristics approaches based on the MILP model to explore the instances that are out of its reach.

2. The VWALBP

In the VWALBP stated in Müller-Hannemann&Weihe [2], the dimensions of the workpieces are larger than the size of the visibility windows of the stations. In this kind of assembly line, the cycle decomposes into a number of stationary stages, in which the workpieces stand motionless. The tasks can only be performed during a stationary stage. Once the tasks have been all processed, the line, with the workpieces on it, moves forward. The assignment of tasks to each station is part of the input.

The output of this problem consists of computing the number of stationary stages, the offset between stages in a cycle and the assignment of the tasks to the stages. Finally, the optimization criterion is to minimize the cycle time of the line.

3. Approaches for the VWALBP

We are currently testing the MILP model using ILOG IBM CPLEX 12.2. In order to identify the limits where the model may be applied, we generate a set of medium-sized instances considering an increasing number of tasks (100 to 500), stations (5 to 20) and workpiece lengths (15 to 25 length units (*lu*)). We set the visibility windows of the stations to 10*lu*.

First computational results show that model performance is mainly limited by the workpiece length, followed by the number of tasks. More specifically, instances for pieces 15*lu* long with 300 tasks and 20 stations could be solved optimally, whereas no solution could be found within an hour for pieces 25*lu* long with more than 100 tasks.

Based on the experience obtained so far, we are exploring several research lines. We are first trying to enhance the CPLEX's performance on the MILP model, by customizing the parameters values, and also by considering specific reformulations of the model, such as the addition of redundant constraints. Secondly, we are also considering developing heuristics to build feasible solutions to start with before the MILP model is solved.

In order to address larger instances that fall out of the reach of the model, we propose a matheuristic approach inspired on the corridor method (CM) [3], which starts from a feasible solution and uses the model to generate improved solutions, within a neighborhood defined by additional constraints.

Furthermore, we have successfully implemented an efficient task assignment submodel which allows us to approach the optimization by means of exploring the space of motion patterns since we can, given the motion pattern, to find in a short time, the corresponding optimal assignment of tasks.

The results of the current computational experience will determine the research direction to be undertaken.

4. Conclusions and perspectives

The inherent complexity of assembly line optimization problems usually implies that exact approaches lead to unreasonable computational times when dealing with large real-world instances.

In this work we address the VWALBP, in which the dimensions of the workpieces are larger than the size of the visibility windows of the stations. We first use a MILP model to try to solve it optimally. Then, for large instances that cannot be solved in a reasonable timel, we consider heuristic and matheuristic-based approaches to generate improving solutions.

References

[1] A. Corominas and R. Pastor. A MILP model for the Visibility Windows Assembly Line Balancing Problem (VWALBP) : the case of the Müller_Hannemann & Weihe problem. 2010. Technical report.

[2] M. Müller-Hannemann and K. Weihe. Moving policies in cyclic assembly line scheduling. *Theoretical Computer Science*, 351 : 425-436, 2006.

[3] V. Maniezzo, T. Stützle and S. Voβ. Matheuristics : Hybridizing Metaheuristics and Mathematical Programming. *Springer*, 10 : 11, 2009.

Heurísticas para el Visibility Windows Assembly Line Balancing Problem (VWALBP)

In proceedings of the 5th International Conference on Industrial Engineering and Industrial Management, *XV Congreso de Ingeniería de Organización, 201-205, Cartagena, Spain, 7-9 September, 2011.*

Heurísticas para el Visibility Windows Assembly Line Balancing Problem (VWALBP)

Gema Calleja[1], Albert Corominas[1], Alberto García-Villoria[1], Rafael Pastor[1]

[1] Instituto de Organización y Control (IOC). Universitat Politècnica de Catalunya (UPC).
Av. Diagonal 647, Planta 11, 08028 Barcelona
{gema.calleja,albert.corominas,alberto.garcia-villoria,rafael.pastor}@upc.edu

Palabras clave: ventanas de visibilidad, equilibrado de líneas

1. Introducción

En este trabajo se trata el problema conocido como problema de equilibrado de líneas de montaje con ventanas de visibilidad o Visibility Windows Assembly Line Balancing Problem (VWALBP), que ocurre en varios entornos de producción automatizados. En particular este problema surge, por ejemplo, en la producción de placas de circuito impreso (PCIs) en líneas *pick&place*. Este tipo de líneas consta de varias estaciones en paralelo que montan los componentes en posiciones predefinidas sobre la superficie de la placa. El montaje se realiza de modo cíclico (en cada ciclo se completa una pieza) y consiste en escoger (*pick*) un componente de un alimentador, trasladarlo hacia la placa, y colocarlo (*place*) en su posición correspondiente.

A diferencia de los problemas tradicionales de equilibrado de líneas, en los que se suele asumir que cada estación tiene acceso a toda una pieza entera, el VWALBP presenta la siguiente particularidad: la longitud de la pieza puede ser mayor que el ancho de la estación que la procesa y, en consecuencia, cada estación solamente puede acceder a la porción limitada de las piezas que está dentro de su *ventana de visibilidad*.

El VWALBP fue descrito por Müller-Hannemann y Weihe (2006) y formalizado con un modelo de programación lineal entera mixta (PLEM) por Corominas y Pastor (2010). Este modelo fue utilizado por Calleja et al. (2011) y se resolvieron ejemplares de un tamaño hasta cierto límite.

En la actualidad se están desarrollando nuevas líneas de investigación basadas en procedimientos heurísticos para resolver los ejemplares de mayor tamaño que quedan fuera del alcance del modelo de PLEM. En esta comunicación se presenta la estrategia heurística considerada en la investigación en curso, estructurada como sigue. En la sección 2 se describe el problema. En las secciones 3 y 4 se presentan la estrategia de resolución y los resultados

computacionales, respectivamente. Por último la sección 5 contiene las conclusiones y las futuras líneas de investigación.

2. Descripción del problema

En esta sección se describen las características del VWALBP. El output esperado del problema y el objetivo a optimizar se especifican en los apartados 2.1 y 2.2, respectivamente.

La característica más importante del VWALBP es la existencia de ventanas de visibilidad en la línea, de manera que una tarea únicamente puede ser procesada si está dentro de la ventana de visibilidad de la estación en la que debe ser realizada. La longitud de las piezas es mayor que el ancho de las estaciones, lo que significa que una misma estación puede procesar partes de dos piezas consecutivas y una misma pieza puede ser procesada por varias estaciones. La Figura 1 muestra un ejemplo de una línea de montaje con tres estaciones. Nótese que las estaciones 1 y 3 solamente pueden procesar una parte de una pieza, mientras que la estación 2 puede procesar simultáneamente partes de dos piezas consecutivas.

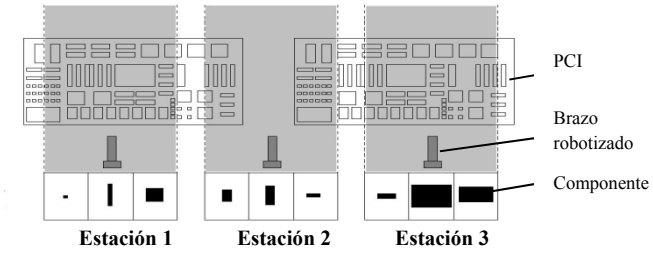

Figura 1. Ejemplo de una línea con tres estaciones. El área gris corresponde a las ventanas de visibilidad.

A continuación se describe el proceso de montaje. La línea debe procesar un número de piezas iguales. Las piezas se colocan sobre la línea con una separación fija entre ellas y son transportadas a través de las estaciones mediante una cinta transportadora. El número de estaciones es conocido. Como particularidad de este problema respecto de otros problemas de equilibrado se asume que cada estación debe procesar un conjunto preasignado de tareas.

El proceso cíclico de montaje consta de una serie de *etapas estacionarias*, separadas entre sí por un *desplazamiento de avance*. En una etapa estacionaria la línea, con las piezas sobre ella, está inmóvil. Cada estación realiza sucesivas tareas de *pick&place*. Una vez se han completado todas las tareas de una etapa estacionaria específica, la línea comienza el desplazamiento de avance. La cinta hace avanzar la línea (y al mismo tiempo las piezas) en un desplazamiento a determinar. El mínimo desplazamiento en que se podría mover la línea es un valor Δ llamado paso elemental que depende de la tecnología de la línea. Los desplazamientos de avance entre etapas (el número de pasos elementales que se desplaza la línea) no son necesariamente iguales. Mientras la línea está en movimiento no se permite realizar ninguna tarea sobre las piezas. Después, comienza la siguiente etapa estacionaria. Las etapas estacionarias y los desplazamientos de avance se repiten cíclicamente.

Entre dos etapas estacionarias consecutivas, existe un tiempo T necesario para acelerar/desacelerar la cinta.

Debido al comportamiento cíclico de la línea, la posición de las piezas en la línea en cada una de las etapas estacionarias queda determinada según un patrón denominado *esquema de avance*, constituido por:

- La posición de referencia x: es la distancia del borde derecho de la pieza respecto al límite izquierdo de la primera ventana de visibilidad en la primera fase estacionaria.
- El número S de etapas estacionarias (que es igual al número de desplazamientos del esquema de avance).
- Los desplazamientos de avance δ_s (s = 1,…, S).

La Figura 2 muestra un esquema de avance con tres etapas estacionarias (S = 3). Después del último desplazamiento de avance, las piezas han sido desplazadas exactamente en la distancia A, que corresponde a la distancia entre los bordes derechos de dos piezas consecutivas y coincide con la suma de los desplazamientos de avance:

$$A = \Delta \cdot \sum_{s=1}^{S} \delta_s \tag{1}$$

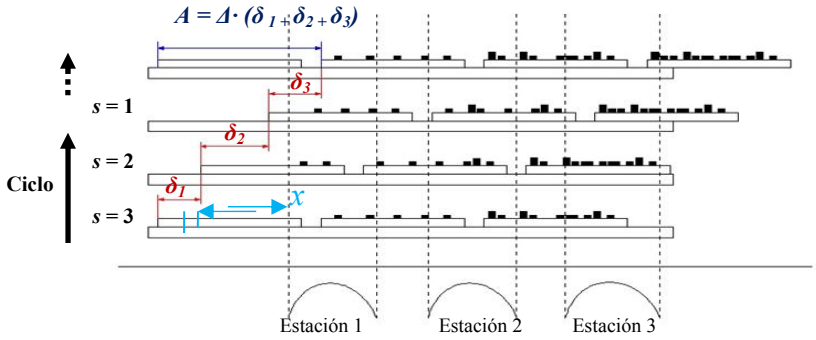

Figura 2. Esquema de avance cíclico de una línea con tres etapas estacionarias.

2.1. Output

El output deseado consiste en:

i) un esquema de avance y

ii) la asignación de cada tarea a una de las etapas estacionarias.

Recuérdese que las tareas están a priori asignadas a las estaciones (la asignación de tareas a estaciones es parte del input).

2.2. Objetivo

El objetivo del problema consiste en minimizar el tiempo de ciclo expresado en la ecuación (2), el cual consiste en la suma de:

i) el número S de etapas estacionarias multiplicado por el tiempo T

ii) las duraciones de cada etapa estacionaria s, C_s, que constituyen un ciclo.

$$[MIN] z = S \cdot T + \sum_{s=1}^{S} C_s \tag{2}$$

3. Estrategia de resolución

La estrategia de resolución se basa en descomponer el VWALBP en dos subproblemas: cálculo de un esquema de avance factible y asignación de tareas a etapas estacionarias, los cuales se

describen en los apartados 3.1 y 3.2, respectivamente. La solución del problema global se obtiene a partir de la unión de las soluciones obtenidas en los dos subproblemas.

3.1. Cálculo de un esquema de avance factible

El esquema de avance se calcula mediante un algoritmo diseñado a medida. Se ha desarrollado un heurístico que, dada una posición de referencia x inicial, genera un esquema de avance factible con el menor número posible de etapas estacionarias S. La motivación de obtener un esquema con el número mínimo de etapas de avance es reducir al máximo los tiempos de aceleración/desaceleración. De esta manera consideramos que se obtendrán soluciones buenas.

A continuación se describe el algoritmo desarrollado. El procedimiento debe determinar el valor de los desplazamientos del esquema de avance de la línea de forma que cada tarea sea visible en su estación correspondiente al menos en una de las etapas estacionarias. Para cada estación, y para cada tarea asignada a dicha estación, se calcula la máxima distancia que se podría desplazar la tarea para poder ser realizada, que corresponde a la distancia entre la posición de la tarea en la línea hasta la posición del límite derecho de su estación. De esta manera se evita que la tarea pueda desplazarse más allá de la ventana de visibilidad de su estación correspondiente. El mínimo valor de entre los máximos desplazamientos posibles en para todas las estaciones de la línea determina el valor del primer desplazamiento del esquema de avance (δ_1). Los desplazamientos siguientes (δ_2, ..., δ_S) se calculan siguiendo el mismo razonamiento, y el cálculo finaliza cuando el desplazamiento acumulado de los desplazamientos es igual a A/Δ.

En la Figura 3 se muestra el pseudocódigo utilizado para la generación del esquema de avance. El funcionamiento básico del algoritmo es el siguiente. En cada iteración se calcula el valor del desplazamiento δ_s (correspondiente al número de pasos elementales que se desplaza la línea) hasta que el desplazamiento acumulado sea igual a A/Δ. Para calcular δ_s se determina el mínimo de los máximos desplazamientos posibles para cada estación mediante el método *NumMaxPasosElem*, que devuelve el número máximo de pasos elementales que puede recorrer la tarea j para que sea visible en la máquina i. Para ello se parte de una posición de referencia inicial para la primera pieza en la línea (x). Se elige como valor de x la máxima distancia posible del borde derecho de la pieza respecto al límite izquierdo de la primera estación, $x = R_1 + a_1^{min}$, donde R_1 es la posición del límite derecho de la primera estación y a_1^{min} es la posición de la tarea más cercana al borde derecho de la pieza. Dado el número total de pasos elementales desplazados en el ciclo actual S^T, la estación i, el conjunto J_i de tareas asignadas a dicha estación, las coordenadas de su límite izquierdo L_i y derecho R_i, una tarea j asignada a la estación i y su posición a_j respecto al borde derecho de la pieza, se calcula el valor *aux* correspondiente a la distancia desde la posición de la tarea en la línea hasta el lado izquierdo de su estación correspondiente. El menor valor de *aux* corresponde a la tarea más cercana a la izquierda de su estación y por tanto determina el máximo desplazamiento que puede moverse la línea de forma que dicha tarea quede dentro de la ventana de visibilidad.

Una vez calculado el valor de δ_s para todas las tareas y para todas las estaciones, se actualiza el valor del desplazamiento acumulado S^T, y se repite iterativamente el cálculo para obtener el valor de los siguientes desplazamientos del esquema de avance. Finalmente, el procedimiento finaliza cuando se cumple la condición $S^T = A/\Delta$ y como resultado se obtiene un esquema de avance factible con S desplazamientos (δ_1, δ_2, ..., δ_S).

$s := 1; S^T := 0$

$fin := \left(S^T = A/\Delta\right)$

mientras (!fin) hacer

$\delta_s := \min\left(\frac{A}{\Delta} - S^T, \min_{\forall i} \min_{\forall j \in J_i}\left(NumMaxPasosElem\left(S^T, i, j\right)\right)\right)$

$S^T := S^T + \delta_s$

$fin := \left(S^T = A/\Delta\right)$

si (!fin) entonces $s := s + 1$ *fsi*

fmientras

NumMaxPasosElem(S^T, *i*, *j*) {

$aux := \left[L_i - \left(x + S^T \cdot \Delta - a_j\right)\right] \% A$

si $aux = 0$ entonces $aux := A$ *fsi*

$d := aux + \left(R_i - L_i\right)$

devolver $\delta_s := \left\lfloor d/\Delta \right\rfloor$

}

Figura 3. Pseudocódigo del cálculo del esquema de avance factible

3.2. Asignación de tareas a etapas estacionarias

Una vez generado un esquema de avance factible, el siguiente paso consiste en asignar las tareas a una de las etapas estacionarias en que es visible. Para ello se ha propuesto un modelo de PLEM que calcula, utilizando como input el esquema de avance generado en el punto anterior, una asignación óptima de cada tarea a una de las etapas estacionarias. A continuación se muestra el modelo utilizado:

***Datos**:*

m número de estaciones

N número de tareas

J_i conjunto de tareas de la estación i $\quad J_i \cap J_k \neq \varnothing \ \ \forall i,k$ (i=1,.., m, k=1,...,m, $i \neq k$)

p_j tiempo de proceso de la tarea j (j = 1, ..., N)

S número de desplazamientos en un ciclo

Π_j conjunto de etapas estacionarias en las que la tarea j es visible dentro de la ventana de visibilidad de la estación en la que debe realizarse (j = 1, ..., N)

Variables:

$y_{js} \in \{0,1\}$, y_{js} = 1 sii la tarea j se realiza en la etapa estacionaria siguiente al desplazamiento $s - 1$ (j = 1,.., N; $s \in \Pi_j$)

C_s duración, para toda la línea, correspondiente a la etapa estacionaria siguiente al desplazamiento $s - 1$ (s = 1,..., S)

***Modelo**:*

$$[MIN]\ z = \sum_{s=1}^{S} C_s \tag{3}$$

$$\sum_{s=1 \mid s \in \Pi_j}^{S} y_{js} = 1 \qquad j = 1,...,N \tag{4}$$

$$\sum_{j \in J_i \mid s \in \Pi_j} p_j \cdot y_{js} \le C_s \qquad i = 1,...,m;\ s = 1,...,S \tag{5}$$

A partir del esquema de avance, se conoce el conjunto de etapas estacionarias en las que cada tarea es visible dentro de su estación correspondiente, Π_j. El objetivo (3) es minimizar la duración de las etapas estacionarias. Las restricciones (4) imponen que cada tarea es asignada a una, y solamente una, de las etapas estacionarias del ciclo, y (5) aseguran que la duración correspondiente a las etapas estacionarias no es menor que el tiempo de proceso de cualquiera de las estaciones.

4. Experiencia computacional

Se generó un juego de datos a partir de diferentes rangos de valores para la separación entre piezas (A={11-16, 17-21, 22-26, 27-31, 32-36, 37-41}), el número máquinas (m={5-10, 11-20, 21-30, 31-40}), y el número de tareas (N={50-200, 201-400, 401-600, 601-800, 801-1000}). La longitud de la pieza se puede deducir del parámetro A, ya que se corresponde con el valor de A menos el valor del espacio entre dos piezas consecutivas en la línea (en nuestro caso dicho espacio es de 1 unidad de longitud –*ul*-, luego la longitud de la pieza es A -1). Para cada combinación de estos rangos se generaron 10 ejemplares, obteniendo así un juego total de 6x4x5x10=1.200 ejemplares. El ancho de las estaciones se fijó en 10 *ul*, y el tiempo T de aceleración/desaceleración se fijó en 200 unidades de tiempo. La posición de las tareas sobre la pieza se obtuvo de forma aleatoria a lo largo de la longitud de la misma. Por último, las tareas se asignaron a las estaciones de forma equiprobable, teniendo en cuenta que cada estación debe tener al menos una tarea asignada.

Se intentó resolver este juego de datos con el modelo de PLEM mediante el solver IBM ILOG CPLEX12.2, limitando el tiempo computacional a una hora. Se observa que la longitud de la pieza y el número de tareas son los parámetros que más influyen en la complejidad de resolución del problema: cuanto mayor es la longitud de la pieza respecto al ancho de las estaciones y mayor es el número de tareas, más difícil resulta hallar una solución óptima. El número de estaciones no parece tener una influencia significativa. Así, para piezas de hasta 15 *ul* el modelo es capaz de resolver ejemplares de hasta 1.000 tareas, mientras que para piezas de hasta 20 y 25 el límite es de hasta 800 y 600 tareas, respectivamente. Finalmente, para piezas de dimensiones mayores, el modelo alcanzó a resolver hasta 200 tareas.

Para obtener una estimación de la calidad de las soluciones producidas por este heurístico, comparamos las soluciones obtenidas con aquellos ejemplares de los que conocemos la solución óptima. Para el 69,9% de estos ejemplares, el heurístico devolvió una solución óptima. Para el resto, la diferencia respecto al valor óptimo de la función objetivo es en promedio de 7,4%. Además, este heurístico es extremadamente rápido: el tiempo de ejecución para la generación del esquema de avance es despreciable, y para la asignación de tareas a etapas estacionarias es en promedio del orden de centésimas de segundo. Se observó también que el número de etapas de avance de las soluciones devueltas por el heurístico coincide con el número de etapas de las soluciones óptimas excepto en un caso. En dicho caso la solución óptima tiene una etapa menos que en el heurístico, ya que la posición de referencia x en el modelo es parte del output a determinar. Así, parece razonable presuponer que los esquemas de avance con un número pequeño de etapas de avance forman parte de las buenas soluciones.

5. Conclusiones y futuras líneas de investigación

En este trabajo se presenta el problema de equilibrado de líneas de montaje de visibilidad (Visibility Windows Assembly Line Balancing Problem, VWALBP), en el que, a diferencia de los problemas tradicionales de equilibrado de líneas, cada estación solamente tiene acceso a una porción limitada de las piezas dentro del área correspondiente a su ventana de visibilidad. El objetivo consiste en obtener el esquema de avance de la línea y la asignación de tareas a etapas estacionarias de forma que el tiempo de ciclo sea mínimo.

La resolución del VWALBP se aborda mediante la utilización de un método heurístico para ejemplares de grandes dimensiones que quedan fuera del alcance del modelo de PLEM. El heurístico descompone el problema en dos partes. En primer lugar se genera un esquema de avance factible mediante un algoritmo. Después, en la segunda parte se utiliza dicho esquema de avance como input de un modelo exacto que calcula una asignación óptima de las tareas a una de las etapas estacionarias del ciclo.

La eficiencia con que se resuelve el modelo de asignación permite pensar en desarrollar otras heurísticas basadas en reducir el espacio de búsqueda al esquema de avance. Por ejemplo, generar un vecindario a partir de un esquema de avance inicial y a continuación utilizar el modelo de asignación para obtener la asignación óptima de tareas correspondiente a uno de los esquemas de avance generados. De este modo se obtiene un vecindario de soluciones al que se puede aplicar búsqueda local para tratar de mejorar la solución inicial.

Otra propuesta es considerar como punto de partida un esquema de avance inicial con el mayor número de etapas estacionarias (es decir, el caso en el que la línea se desplaza entre etapas el valor del paso elemental Δ). La heurística consistiría en generar iterativamente nuevos esquemas de avance agrupando, si es posible, las dos etapas estacionarias consecutivas que mayor ahorro proporcionen en el tiempo de ciclo al ser agrupadas. A partir de este heurístico se podría desarrollar un algoritmo tipo GRASP (Greedy Randomized Adaptative Search Procedure) donde la selección de la pareja de etapas a agruparse se aletoriza.

Otra posible línea de investigación es el desarrollo de mateheurísticas que, a partir de una solución heurística inicial, utilicen el modelo matemático para generar mejores soluciones, dentro de un vecindario definido por restricciones adicionales en el modelo. Un ejemplo de posible aplicación es el Corridor Method (CM), que permitiría reducir el espacio de búsqueda a porciones restringidas del espacio de soluciones del problema.

Referencias

Calleja, G.; Corominas, A.; García-Villoria, A.; Pastor, R.; (2011). Exact and Heuristic Approaches for the Visibility Windows Assembly Line Balancing Problem (VWALBP). Actas del 12º Congrès Annuel de la Société Française de Recherche Opérationnelle at d'Aide à la Décision (ROADEF), Volumen II, página 583.

Corominas, A.; Pastor, R. (2010). A MILP model for the Visibility Windows Assembly Line Balancing Problem (VWALBP): the case of the Müller-Hannemann & Weihe problem. Technical report.

Müller-Hannemann, M.; Weihe, K.; (2006). Moving policies in cyclic assembly line scheduling. Theoretical Computer Science, No. 351, 425-436.

Enhanced MILP model for the Accessibility Windows Assembly Line Balancing Problem (AWALBP)

In proceedings of the 13rd Annual Congress of the French National Society of Operations Research and Decision Science, 13è Congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2012), 117-118, Angers, France, 11-13 April, 2012.

Enhanced MILP model for the Accessibility Windows Assembly Line Balancing Problem (AWALBP)

Gema Calleja[1], Albert Corominas[1], Alberto García-Villoria[1], Rafael Pastor[1]

[1] IOC Research Institute
Universitat Politècnica de Catalunya (UPC)
Av. Diagonal 647, 11th floor , 08028, Barcelona, Spain
{gema.calleja,albert.corominas,alberto.garcia-villoria,rafael.pastor}@upc.edu

Keywords: Assembly line balancing, mixed-integer linear programming, matheuristics.

1. Introduction

The Accessibility Windows Assembly Line Balancing Problem (AWALBP) arises in those assembly lines where, as opposed to traditional assembly lines, the dimensions of the workpieces are larger than the width of the workstations. This means that, at any cycle, a workstation cannot access to one whole workpiece, but only to a restricted portion of one or two consecutive workpieces [1]. In our problem the cycle decomposes into stationary stages separated between them by forward steps, according to a cyclic movement scheme.

In a previous study [2], we proposed and applied a mixed-integer linear programming (MILP) model to solve AWALBP. In this paper, we present an enhanced MILP model to increase the number of instances solved by the earlier model. The main enhancements comprise i) use of reformulation techniques, and ii) incorporation of bound constraints, which resulted in significant performance improvement.

2. Enhanced MILP model for the AWALBP

Based on our previous model [2], we present an enhanced MILP model to solve AWALBP. The differences between the two models are described next. In the new formulation, new variables and new constraints were incorporated, whereas some other variables and constraints were discarded. More specifically, the integer variable k_j, representing the number of workpieces in the line that precede a workpiece when task j is performed in the latter, was discarded. A new binary variable, b_{jsk}, was incorporated instead, which equals 1 iff task j is performed in

stationary stage *s* and there are *k* workpieces in the line that precede the workpiece when task *j* is being performed in the latter. As a result, some specific sets of constrainsts were transformed and a new set of constraints arose.

On the other hand, we incorporated to the model upper and lower bounds on the cycle time and the number of stationary stages. The upper bounds were obtained from an initial solution using the heuristic introduced in [3], whereas the lower bounds were derived using pre-computed information from the data instances.

The enhanced MILP model clearly outperforms the earlier model in terms of number of optimal solutions, which increased by 30%. In order to solve larger instances that are still out of reach of the enhanced model, we use our heuristic approach [3], which turned out to perform very fast (less than 25 milliseconds on average) and produces good feasible solutions.

3. Computational experience

Since there are no existing benchmark sets in the literature, we used a randomly generated set of 1.200 instances with an increasing number of tasks (50 to 1000), workstations (5 to 40) and workpiece lengths (10 to 40 length units (*lu*)). We set the accessibility windows of the workstations to 10 *lu*. To test the proposed MILP model, we used CPLEX 12.2 solver with a time limit of 1 hour.

Computational results show a significant improvement in the percentage of instances solved. More specifically, the number of optimal solutions increased from 40% in the earlier model to 70% in the enhanced MILP model.

For the remaining 30% of instances that could not be solved exactly we applied the aforementioned heuristic approach. The evaluation of the heuristic quality showed that 60% of the heuristic solutions coincided with the optimal solutions obtained by our MILP models, being the average gap for not coincident solutions less than 4%.

By using the presented MILP model, all instances with workpiece lengths up to 15 *lu* are solved. Instances with workpiece lengths up to 25 *lu* could also be solved almost entirely (98%). Finally, 73%, 40% and 22% of the instances corresponding to 30, 35 and 40 workpiece lengths, respectively, were exactly solved.

4. Conclusions and perspectives

In this paper, we propose an enhanced MILP model to solve the AWALBP that leads to a significant increase in the number of instances exactly solved. In order to improve the solution of larger instances that are still out of reach for the model, we will shift our focus of research to metaheuristics and matheuristics.

We first intend to develop metaheuristics to generate neighbor solutions in the space of the movement schemes, since we have developed a MILP model that enables an efficient task assignment. Classical metaheuristics such as Simulated Annealing, Tabu Search or Variable Neighborhood Search may perform well in our problem.

We also consider developing a matheuristic approach made by the interoperation of a metaheuristic and the presented MILP model.

References

[1] M. Muller-Hannemann and K. Weihe. Moving policies in cyclic assembly line scheduling. Theoretical Computer Science, 351; 425-436, 2006.

[2] G. Calleja, A. Corominas, A. García-Villoria and R. Pastor. Exact and Heuristic Approaches for the Visibility Windows Assembly Line Balancing Problem (VWALBP). 12th Annual Congress on the French National Society of Operations Research and Decision Science, (ROADEF, Saint-Étienne, France, from 2nd to 4th March 2011); Nº 386 p. II-583.

[3] G. Calleja, A. Corominas, A. García-Villoria and R. Pastor. Heurísticas para el Visibility Windows Assembly Line Balancing Problem (VWALBP). V International Conference on Industrial Engineering and Industrial Management / XV Congreso de Ingeniería de Organización (CIO, Cartagena, Spain, from 7th to 9th September 2011); p. 201 – 205.

Modelo de PLEM mejorado Accessibility Windows Assembly Line Balancing Problem (AWALBP)

In Proceedings of the 6th International Conference on Industrial Engineering and Industrial Management: XVI Congreso de Ingeniería de Organización (CIO 2012), 879-886, Vigo, Spain, 18-20 July, 2012.

Modelo de PLEM mejorado para el Accessibility Windows Assembly Line Balancing Problem (AWALBP)

Calleja Sanz G[2], Corominas Subias A, García Villoria A, Pastor Moreno R

Abstract (English) The Accessibility Windows Assembly Line Balancing Problem (AWALBP) occurs in those assembly lines where the length of the workpieces is large relative to the width of the workstations. As a result, each workstation can only access to the limited portion of workpiece(s) that is inside its accessibility window. In previous works we proposed a mixed-integer linear programming (MILP) model and a heuristic decomposition approach to solve AWALBP. Computational results revealed the size limits of the instances that could be solved. In this work, we provide an enhanced MILP model using reformulations and additional bound contraints, which significantly improves the percentage of the instances optimally solved.

Resumen (Castellano) El problema denominado Accessibility Windows Assembly Line Balancing Problem (AWALBP) ocurre en aquellas líneas de montaje donde la longitud de las piezas es mayor que el ancho de las estaciones que las procesan. Como resultado, cada estación solamente tiene acceso a la porción limitada de las piezas que están dentro de su ventana de accesibilidad. En trabajos anteriores se presentó un modelo de programación lineal entera mixta (PLEM) y una heurística basada en la descomposición del problema para resolver el AWALBP. En este trabajo se presenta un modelo de PLEM mejorado mediante reformulaciones y adición de cotas, el cual permite aumentar significativamente el porcentaje de los ejemplares resueltos óptimamente.

Keywords: accessibility windows, line balancing, mixed-integer linear programming

Palabras clave: ventanas de accesibilidad, equilibrado de líneas, programación lineal entera mixta

[2] Gema Calleja Sanz (✉)
Instituto de Organización y Control (IOC). Universitat Politècnica de Catalunya (UPC) . Avda Diagonal, 647, 11th floor, 08028 Barcelona, Spain. e-mail: gema.calleja@upc.edu

1. Introducción

En los problemas más extendidos de equilibrado de líneas se suele asumir que cada estación tiene acceso a toda una pieza entera al mismo tiempo, y que cada pieza solamente puede ser procesada por una única estación al mismo tiempo. Sin embargo, el problema denominado Accessibility Windows Assembly Line Balancing Problem (AWALBP) presenta la particularidad de que la longitud de las piezas es mayor que el ancho de las estaciones que las procesan (Fig. 1). Como consecuencia, una misma pieza puede ser procesada por varias estaciones a la vez, y una misma estación puede procesar partes de dos piezas consecutivas a la vez.

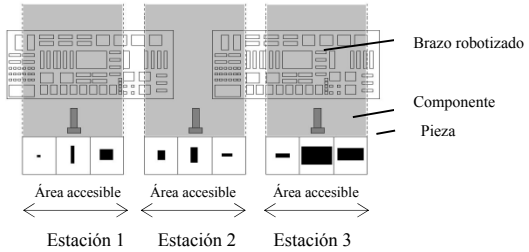

Fig. 1 Ejemplo de una línea de montaje con ventanas de accesibilidad

El AWALBP es una variante del problema generalizado de equilibrado de líneas (GALBP) que ocurre en varios entornos de producción automatizados, por ejemplo, en el montaje automatizado de placas de circuitos impresos (Tazari et al, 2006). Este problema fue descrito por Müller-Hannemann y Weihe (2006) y formalizado con un modelo de programación lineal entera mixta (PLEM) por Corominas y Pastor (2009). Dicho modelo fue implementado por Calleja et al (2011a) y se resolvieron de forma óptima ejemplares hasta cierto límite. Posteriormente, se desarrolló una heurística basada en la descomposición del problema (Calleja et al 2011b).

En este trabajo se presenta un modelo de PLEM mejorado, cuyas modificaciones respecto al modelo inicial consisten en i) reformulación de variables y restricciones y ii) adición de cotas. La estrategia de resolución propuesta consiste en la combinación de una heurística y el modelo propuesto. Los resultados de la experiencia computacional realizada revelan un incremento significativo del número de ejemplares resueltos. El resto de este trabajo se estructura como sigue. En la sección 2 se describe el problema. En la sección 3 se presenta la estrategia de resolución y el modelo de PLEM mejorado. En la sección 4 se presentan los resultados computacionales. Por último, la sección 5 contiene las conclusiones y las futuras líneas de investigación.

2. Descripción del problema

Un número (potencialmente infinito) de piezas idénticas deben ser procesadas en la línea de montaje. Las piezas se colocan sobre la línea con una separación constante entre sí y avanzan mediante una cinta transportadora a través de varias estaciones en serie. El proceso de montaje es cíclico y consta de un número S de *etapas estacionarias* separadas entre sí por un *paso de avance*. En la etapa estacionaria s (s=1,…,S), la línea está inmóvil, y las estaciones realizan tareas de montaje sobre las piezas. Una tarea consiste en tomar un componente de un alimentador y colocarlo en una posición predefinida de la pieza. Una vez las estaciones han realizado todas las tareas correspondientes a dicha etapa estacionaria, se produce un paso de avance δ_s, mediante el cual las piezas son transportadas hacia delante una distancia determinada ($\Delta \cdot \delta_s$, donde Δ es una longitud denominada paso elemental, que depende de la tecnología de la línea).

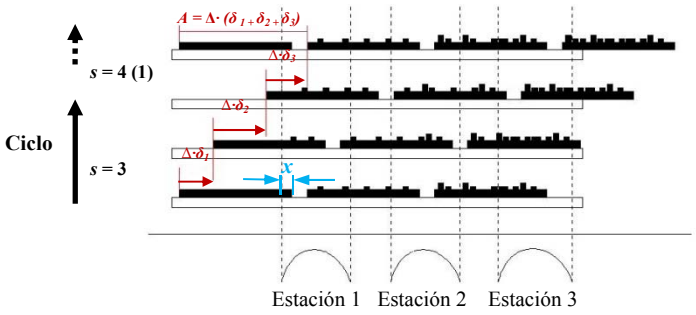

Fig. 2 Esquema de avance en un ciclo de montaje con tres etapas estacionarias

El movimiento de las piezas en la línea se repite cíclicamente según un *esquema de avance*, que determina la posición exacta de las piezas en cada una de las etapas estacionarias del ciclo. Debido a la accesibilidad limitada de cada estación, habitualmente ocurre que una tarea solamente puede ser realizada en un subconjunto de etapas estacionarias, específicamente en aquellas etapas en las que la tarea es accesible desde su estación correspondiente. La Fig.2 muestra cuatro instantáneas de la posición de las piezas en un ciclo de montaje con tres etapas estacionarias (S=3). Inicialmente las piezas se sitúan en la línea tomando como referencia la posición del borde derecho de la primera pieza, situada con un desplazamiento x respecto al límite izquierdo de la primera estación. Las flechas al inicio de cada etapa estacionaria indican los movimientos de la línea desde la etapa actual hacia la etapa siguiente. En la siguiente etapa después de haberse completado un ciclo, una nueva pieza entra en la línea. Al mismo tiempo, una pieza totalmente terminada abandona la línea. Una vez completado el ciclo, las piezas vuelven a ocupar la misma posición que en la primera etapa del ciclo anterior, y se han desplazado una distancia A, que corresponde a la suma de las distancias recorridas en los pasos de avance. De este modo, el esquema de avance queda definido por:

- El desplazamiento inicial x de las piezas en la primera etapa del ciclo.
- El número S de etapas estacionarias (igual al número de pasos de avance).
- La secuencia de las distancias recorridas en cada uno de los pasos de avance, $\Delta \cdot \delta_s$ (s = 1,…, S).

El problema de optimización consiste en determinar:

i) un esquema de avance y
ii) para cada tarea, la asignación a una etapa estacionaria en la que su posición sea accesible desde la estación que la debe procesar.

El conjunto de estaciones que pueden realizar una tarea es parte del input. Debido a la tecnología de la línea, puede ocurrir que dicho conjunto esté formado por una única estación. En nuestro problema, inspirado en el caso real descrito por Muller-Hannemann y Weihe (2006), dicho conjunto está formado por una sola estación.

El objetivo del problema es la minimización del tiempo de ciclo, que consiste en minimizar la suma de i) el tiempo de desplazamiento en las etapas de avance (correspondiente al tiempo para transportar las piezas a velocidad máxima en la línea más un tiempo adicional T para acelerar/desacelerar la línea entre dos etapas estacionarias consecutivas) y ii) el tiempo de proceso en las S etapas estacionarias. Dado que el tiempo para transportar las piezas en la línea a velocidad máxima es constante, no se tiene en cuenta en la función objetivo:

$$[MIN] z = T \cdot S + \sum_{s=1}^{S} C_s \qquad (1)$$

donde C_s corresponde a las duraciones de cada una de las etapas estacionarias que constituyen un ciclo.

3. Estrategia de resolución

La estrategia de resolución propuesta consiste en la combinación de dos elementos: i) la adición de cotas de la función objetivo y del número de etapas estacionarias, obtenidas mediante la heurística descrita en Calleja et al (2009) y ii) un modelo de PLEM mejorado con relación al propuesto por Corominas y Pastor (2009), como se explica a continuación.

La heurística mencionada anteriormente proporciona una solución factible con el menor número de etapas estacionarias para un desplazamiento inicial x dado. Dicha heurística se aplica iterativamente para cada uno de los valores posibles de desplazamiento inicial x y, entre las soluciones generadas, se selecciona la solución con el menor valor de la función objetivo, lo que proporciona una cota superior del valor óptimo de la función objetivo, UB_{CT}. De las soluciones generadas, la de menor número de etapas estacionarias proporciona una cota inferior del número de etapas estacionarias, LB_S.

La cota inferior de la función objetivo, LB_{CT}, se obtiene mediante la suma de las cotas inferiores de los dos términos que la componen: i) $T \cdot LB_S$ más ii) una cota inferior del tiempo de proceso $\sum_{s=1}^{S} C_s$, denominada W_{max}, correspondiente al tiempo de proceso de la estación más cargada.

Finalmente, se tiene que el número de etapas estacionarias de la solución óptima no puede ser mayor que la diferencia entre UB_{CT} y W_{max} dividida por el tiempo T, lo cual proporciona el valor de la cota superior del número de etapas estacionarias, UB_S.

El modelo de PLEM mejorado es el que se propone a continuación.

Datos

N número de tareas

m número de estaciones

$[L_i, R_i]$ ventana de accesibilidad de la estación i ($i = 1,\ldots,m$), donde $L_1 = 0$,

$L_i < R_i$ ($i = 1,\ldots,m$), $R_i < L_{i+1}$ ($i = 1,..,m-1$)

A_0 longitud de la pieza

A distancia entre los bordes derechos de dos piezas consecutivas en la línea

T tiempo para acelerar / desacelerar la línea entre dos etapas estacionarias consecutivas

Δ longitud de un paso elemental. Sin pérdida de generalidad, todas las magnitudes de longitud son múltiplos de Δ. En nuestro trabajo, Δ es un valor entero (Δ=1), y en consecuencia, todas las magnitudes de longitud son también enteras

p_j tiempo de proceso de la tarea j (j = 1,…, N)

a_j ($0 \le a_j \le A_0$), distancia respecto al borde derecho de la pieza correspondiente a la tarea j (j = 1,…, N)

J_0 conjunto de tareas ($|J_0| = N$)

J_i conjunto de tareas a realizar en la estación i (i = 1,…, m), donde $\bigcup_{i=1,..,m} J_i = J_0$ y $J_i \cap J_k = \varnothing \ \ \forall i,k$

$\hat{S}$ cota superior del número de etapas estacionarias en un ciclo ($\hat{S} \le A/\Delta$)

Variables

$\delta_s \in Z^+$ número de pasos elementales del paso de avance s (s=1,…, $\hat{S}$)

$\eta_s \in \{0,1\}$ $\eta_s = 1$ sii el paso de avance *s* existe

C_s tiempo de proceso, para toda la línea, correspondiente a la etapa estacionaria siguiente al paso de avance $s-1$ $(j = 1,...,N; s = 1,...,\hat{S})$

$y_{js} \in \{0,1\}$ $y_{js} = 1$ sii la tarea *j* se realiza durante la etapa estacionaria siguiente al paso de avance $s-1$ $(j = 1,...,N; s = 1,...,\hat{S})$

$x \geq 0$ desplazamiento inicial del borde derecho de la primera pieza en la línea con respecto al límite izquierdo de la estación 1 (donde L_1=0): $0 \leq x \leq \min(R_1 + a_1^{\min}, A - \Delta)$ y $a_1^{\min} = \min_{j \in J_i} a_j \; k_j \in Z$ número de piezas en la línea que preceden una pieza cuando la tarea *j* está siendo realizada en la misma (j=1,…,N; $kmin_j \leq k_j \leq kmax_j$), donde, $(i \,|\, j \in J_i)$:

$$kmin_j = \left\lceil \frac{L_i + a_j - \min(R_1 + a_1^{\min}, A - \Delta) - A + \Delta}{A} \right\rceil \text{ y } kmax_j = \left\lfloor \frac{R_i + a_j}{A} \right\rfloor$$

Restricciones

$$[MIN] z = T \cdot \sum_{s=1}^{\hat{S}} \eta_s + \sum_{s=1}^{\hat{S}} C_s \tag{1}$$

$$\sum_{s=1}^{\hat{S}} \delta_s = \frac{A}{\Delta} \tag{2}$$

$$\delta_s \leq \frac{A}{\Delta} \cdot \eta_s \qquad s = 1,...,\hat{S} \tag{3}$$

$$\eta_s \leq \delta_s \qquad s = 1,...,\hat{S} \tag{4}$$

$$\eta_{s+1} \leq \eta_s \qquad s = 1,...,\hat{S}-1 \tag{5}$$

$$A \cdot k_j - a_j + x + \Delta \cdot \sum_{l=1}^{s-1} \delta_l \geq L_i - M_{js} \cdot (1 - y_{js}) \qquad s = 1,...,\hat{S}; j = 1,...,N; i \,|\, j \in J_i \tag{6}$$

$$A \cdot k_j - a_j + x + \Delta \cdot \sum_{l=1}^{s-1} \delta_l \leq R_i + M'_{js} \cdot (1 - y_{js}) \qquad s = 1,...,\hat{S}; j = 1,...,N; i \,|\, j \in J_i \tag{7}$$

$$\sum_{s=1}^{\hat{S}} y_{js} = 1 \qquad j = 1,...,N \tag{8}$$

$$\sum_{j \in J_i} p_j \cdot y_{js} \leq C_s \qquad i = 1,...,m; s = 1,...,\hat{S}; \tag{9}$$

$$\sum_{j \in J_i} y_{js} \leq |J_i| \cdot \eta_s \qquad i = 1,...,m; s = 1,...,\hat{S}; \tag{10}$$

donde $(i \,|\, j \in J_i)$:

$M_{js} = L_i - A \cdot kmin_j + a_j - \Delta \cdot (s-1)$

$M'_{js} = A \cdot (kmax_j + 1) + \min(R_1 + a_1^{\min}, A - \Delta) - R_i - a_j$

El objetivo (1) es la minimización del tiempo de ciclo. La restricción (2) impone que el número de pasos elementales en un ciclo corresponde a la distancia entre los bordes derechos de dos piezas consecutivas; (3) aseguran que el paso *s* existe si tiene un número positivo de pasos elementales; (4) evitan la existencia de pasos de avance de distancia nula de forma que, si el paso de avance *s* existe, entonces tiene un número positivo de pasos elementales; (5) eliminan simetrías, asegurando que el paso de avance s existe sólo si $s-1$ existe; (6) y (7) garantizan, para cada tarea, que es accesible, desde su estación correspondiente, durante la etapa estacionaria en la que la tarea debe realizarse; (8) imponen que cada tarea sea asignada a una única etapa estacionaria; (9), que el tiempo de proceso correspondiente a las etapas estacionarias no es menor que el tiempo de proceso en cualquiera de las estaciones; finalmente, (10) imponen la existencia de una etapa estacionaria cuando al menos una tarea ha sido asignada a la misma.

Las modificaciones incorporadas al modelo son las siguientes. En la nueva formulación, se incorporaron al modelo las cotas superiores e inferiores del tiempo de ciclo y del número de etapas estacionarias descritas anteriormente. Por otro lado, se eliminó la variable entera k_j, que representa el número de piezas en la línea que preceden una pieza cuando la tarea j está siendo realizada en la misma. En su lugar, se introdujo la variable binaria b_{jsk}, que es igual a 1 sii la tarea j se realiza en la etapa estacionaria s y hay k piezas en la línea que preceden la pieza cuando la tarea j está siendo realiza en la misma (j=1,...,N, s=1,…, $\hat{S}$; k=$kmin_j$,…,$kmax_j$). A consecuencia de la introducción de esta variable, las restricciones (6) y (7) se transforman en las restricciones (6') y (7'):

$$A \cdot k - a_j + x + \Delta \cdot \sum_{l=1}^{s-1} \delta_l \geq L_i - M_{js} \cdot (1 - b_{jsk}) \qquad s = 1,...,\hat{S}; j = 1,...,N; k = kmin_j,...,kmax_j \qquad (6')$$

$$A \cdot k - a_j + x + \Delta \cdot \sum_{l=1}^{s-1} \delta_l \leq R_i - M'_{js} \cdot (1 - b_{jsk}) \qquad s = 1,...,\hat{S}; j = 1,...,N; k = kmin_j,...,kmax_j \qquad (7')$$

La estrategia de resolución propuesta combina la heurística y el modelo de PLEM mejorado como se explica a continuación. Dada una solución factible inicial que proporciona una cota superior del tiempo de ciclo, UB_{CT}, se evalúa su optimalidad por comparación con la cota inferior del tiempo de ciclo, LB_{CT}. Si ambos valores coinciden, se concluye que la solución inicial es óptima. En caso contrario, se lanza el modelo de PLEM mejorado, el cual proporciona uno de los siguientes cuatro resultados: i) no hay soluciones factibles, lo cual certifica que la solución inicial es óptima, ii) una solución óptima, iii) una solución factible, con un tiempo de ciclo menor que el de la solución inicial, y iv) no consigue hallar una solución factible en el límite de una hora de tiempo computacional. En los dos primeros casos, el modelo proporciona una solución óptima, mientras que en los dos casos restantes se obtiene una solución factible (en el caso iv) corresponde a la solución proporcionada por la heurística).

4. Experiencia computacional

Con el fin de identificar el porcentaje de ejemplares que es posible resolver mediante la estrategia propuesta, se realizó un estudio computacional que se describe a continuación. Se generó un juego de 1.200 ejemplares con un número creciente de tareas (de 50 a 1000), estaciones (de 5 a 40) y longitudes de piezas (11 a 40 unidades de longitud (*ul*)). El ancho de las estaciones se fijó a 10 ul y el tiempo T de aceleración/desaceleración a 200 unidades de tiempo.

Se utilizó el solver IBM ILOG CPLEX 12.2, con un límite de tiempo computacional de una hora y un gap absoluto de 0,999999. Los resultados muestran un aumento significativo en el porcentaje de las instancias resueltas de forma óptima, del 39,50% en nuestro anterior modelo hasta el 73,08% en el modelo de PLEM propuesto en este trabajo. Análogamente, se consiguió aumentar el porcentaje de instancias resueltas de forma óptima para todas las longitudes de placa. Específicamente, se logró resolver el 99.5, 97.5, 74.5, 42 y 25% de los ejemplares correspondientes a piezas de hasta 20, 25, 30, 35 y 40 ul, respectivamente.

5. Conclusiones y futuras líneas de investigación

En este trabajo se considera el problema denominado Accessibility Windows Assembly Line Balancing Problem (AWALBP), en el que las estaciones solamente tienen acceso a la porción limitada de las piezas que son visibles dentro de su ventana de accesibilidad.

Para resolver el problema se propone una estrategia que combina una heurística con un modelo de PLEM mejorado. La experiencia computacional realizada reveló un aumento significativo del porcentaje de ejemplares resueltos respecto al modelo anterior.

Las futuras líneas de investigación incluyen la utilización de metaheurísticas y mateheurísticas para tratar de aumentar el número de ejemplares resueltos.

Referencias

Calleja G, Corominas A, García-Villoria A, Pastor R (2011a) Exact and heuristic approaches for the Visibility Windows Assembly Line Balancing Problem (VWALBP). *Proceedings of the 12th Congrès Annuel de la Societé Française de Recherche Opérationelle et d'Aide à la Décision (ROADEF), Saint-Étienne:* II, 583.

Calleja G, Corominas A, García-Villoria A, Pastor R (2011b) Heurísticas para el Visibility Windows Assembly Line Balancing Problem (VWALBP). XV Congreso de Ingeniería de Organización CIO, Cartagena (321-328).

Corominas A, Pastor R (2009) *A MILP model for the Visibility Windows Assembly Line Balancing Problem (VWALBP): the case of Müller-Hannemann & Weihe problem*. Working paper. Retrieved February 22, 2012, from the website: http://upcommons.upc.edu/e-prints/bitstream/2117/7047/1/IOC-DT-P-2009-09.pdf.

Müller-Hannemann M, Weihe K (2006) Moving policies in cyclic assembly line scheduling. *Theoretical Computer Science*, 351, (425-436).

Tazari S, Müller-Hannemann M, Weihe K (2006) Workload balancing in multi-stage production processes. In Lecture Notes in Computer Science. *Proceedings of the 5th International Workshop on Experimental Algorithms, WEA 2006: 4007* (49-60).

Using tabu search and MILP for the Accessibility Windows Assembly Line Balancing Problem (AWALBP)

In Proceedings of the XXXIV Congreso Nacional de Estadística e Investigación Operativa (SEIO 2013), 117, Castellón, Spain, 11-13 September, 2013.

Using tabu search and MILP for the Accessibility Windows Assembly Line Balancing Problem (AWALBP)

Gema Calleja (IOC Research Institute, UPC, Barcelona, Spain),
Albert Corominas (IOC Research Institute, UPC, Barcelona, Spain),
Alberto García-Villoria (IOC Research Institute, UPC, Barcelona, Spain),
Rafael Pastor Moreno (IOC Research Institute, UPC, Barcelona, Spain)

The AWALBP arises in those assembly lines where, in contrast to standard ones, the length of the workpieces is larger than the accessibility windows of the workstations. Because of this, only a limited portion of one or two consecutive workpieces can be reached from each station at any moment. In our problem, the cycle decomposes into stationary stages separated between them by forward steps, according to a cyclic movement scheme. Several procedures were previously proposed to solve the problem to optimality and instances up to a certain size limit were solved. In this study, we propose a tabu search (TS) and a combination procedure using TS and a mixed integer linear programming (MILP) model in order to solve larger instances. The neighborhood search is performed in the space of the movement schemes. Results show that a better solution is obtained in most of the cases that could not be previously solved optimally.

Using simulated annealing and MILP for the Accessibility Windows Assembly Line Balancing Problem (AWALBP)

In Proceedings of the XXVI EURO-INFORMS Joint International Conference, 26th European Conference on Operational Research (EURO 2013), 38, Rome, Italy, 1-4 July, 2013.

Using simulated annealing and MILP for the Accessibility Windows Assembly Line Balancing Problem (AWALBP)

Gema Calleja, IOC-DOE, UPC, Av. Diagonal, 647, 11th floor, 08028 Barcelona, Spain, gema.calleja@upc.edu, *Albert Corominas*, *Alberto García-Villoria*, *Rafael Pastor*

The AWALBP is an assembly line balancing problem where the length of the workpieces is larger than the width of the workstations. A procedure using a matheuristic and a mixed integer linear programming (MILP) model was previously tested to solve the AWALBP and it succeeded in finding optimal solutions to instances up to a certain size. We propose simulated annealing (SA) and a hybrid procedure using SA and MILP in order to find good quality solutions for larger instances. Results show that a better solution is obtained in most of the cases that could not be previously solved optimally.

The Accessibility Windows Assembly Line Balancing Problem (AWALBP): A review of advances and trends

In Proceedings of the 20th Conference of the International Federation of Operational Research Societies (IFORS 2014), 191. Barcelona, 13-18 July, 2014.

The Accessibility Windows Assembly Line Balancing Problem (AWALBP): A review of advances and trends

Gema Calleja, Albert Corominas, Alberto García-Villoria, Rafael Pastor

We investigate the Accessibility Windows Assembly Line Balancing Problem (AWALBP), where, in sharp contrast to traditional assembly line problems, only a portion of the workpieces can be reached from each workstation. The literature distinguishes different variants of the problem, and several formulations and solution approaches have been proposed. This talk gives an overview on recent advances in the methods used to solve AWALBP, including exact, heuristic and hybrid methods. An extensive set of computational experiments, along with some guidelines for further lines of research are reported.

MILP-based Tabu Search using Corridor Method for an assembly line balancing problem with accessibility windows

In Proceedings of the 20th Conference of the International Federation of Operational Research Societies (IFORS 2014), 191-192, Barcelona, 13-18 July, 2014.

MILP-based Tabu Search using Corridor Method for an assembly line balancing problem with accessibility windows

Albert Corominas, Gema Calleja, Alberto García-Villoria, Rafael Pastor

In this work, we present an MILP-TS matheuristic for an assembly line balancing problem with accessibility windows. The proposed matheuristic uses an MILP model embedded in a tabu search (TS) algorithm to iteratively solve reduced portions of the original solution space. We use the paradigm of the corridor method to impose exogenous constraints of the original mathematical formulation and, subsequently, we apply an MILP solver to optimally solve the constrained problem. Computational results show the effectiveness of the proposed matheuristic.