Algorithms and Software for Decision Support in Design of Assembly and Transfer Lines

Sergey Malyutin

N°d'ordre NNT : 2016LYSEM020

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Ecole des Mines de Saint-Etienne

Ecole Doctorale N° 488
Sciences, Ingénierie, Santé

Spécialité de doctorat : génie industriel

Soutenue publiquement le 24/10/2016, par :
Sergey Malyutin

Algorithms and Software for Decision Support in Design of Assembly and Transfer Lines

---

Devant le jury composé de :

Prof. Farouk Yalaoui, Université de technologie de Troyes	Rapporteur
Prof. Lyès Benyoucef, Aix-Marseille Université	Rapporteur
Prof. Alain Quilliot, ISIMA, Clermont-Ferrand	Membre de jury
Prof. Alexandre Dolgui, École des Mines de Nantes	Directeur de thèse
Dr. Xavier Delorme, École des Mines de Saint-Étienne	Co-directeur de thèse
Prof. Mikhail Kovalyov, Académie des Sciences de Biélorussie	Co-directeur de thèse

Spécialités doctorales	Responsables :	Spécialités doctorales	Responsables
SCIENCES ET GENIE DES MATERIAUX	K. Wolski Directeur de recherche	MATHEMATIQUES APPLIQUEES	O. Roustant, Maître-assistant
MECANIQUE ET INGENIERIE	S. Drapier, professeur	INFORMATIQUE	O. Boissier, Professeur
GENIE DES PROCEDES	F. Gruy, Maître de recherche	IMAGE, VISION, SIGNAL	JC. Pinoli, Professeur
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche	GENIE INDUSTRIEL	X. Delorme, Maître assistant
SCIENCES ET GENIE DE L'ENVIRONNEMENT	D. Graillot, Directeur de recherche	MICROELECTRONIQUE	Ph. Lalevée, Professeur

EMSE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'Etat ou d'une HDR)

ABSI	Nabil	CR	Génie industriel	CMP
AUGUSTO	Vincent	CR	Image, Vision, Signal	CIS
AVRIL	Stéphane	PR2	Mécanique et ingénierie	CIS
BADEL	Pierre	MA(MDC)	Mécanique et ingénierie	CIS
BALBO	Flavien	PR2	Informatique	FAYOL
BASSEREAU	Jean-François	PR	Sciences et génie des matériaux	SMS
BATTON-HUBERT	Mireille	PR2	Sciences et génie de l'environnement	FAYOL
BEIGBEDER	Michel	MA(MDC)	Informatique	FAYOL
BLAYAC	Sylvain	MA(MDC)	Microélectronique	CMP
BOISSIER	Olivier	PR1	Informatique	FAYOL
BONNEFOY	Olivier	MA(MDC)	Génie des Procédés	SPIN
BORBELY	Andras	MR(DR2)	Sciences et génie des matériaux	SMS
BOUCHER	Xavier	PR2	Génie Industriel	FAYOL
BRODHAG	Christian	DR	Sciences et génie de l'environnement	FAYOL
BRUCHON	Julien	MA(MDC)	Mécanique et ingénierie	SMS
BURLAT	Patrick	PR1	Génie Industriel	FAYOL
CHRISTIEN	Frédéric	PR	Science et génie des matériaux	SMS
DAUZERE-PERES	Stéphane	PR1	Génie Industriel	CMP
DEBAYLE	Johan	CR	Image Vision Signal	CIS
DELAFOSSE	David	PR0	Sciences et génie des matériaux	SMS
DELORME	Xavier	MA(MDC)	Génie industriel	FAYOL
DESRAYAUD	Christophe	PR1	Mécanique et ingénierie	SMS
DJENIZIAN	Thierry	PR	Science et génie des matériaux	CMP
DOUCE	Sandrine	PR2	Sciences de gestion	FAYOL
DRAPIER	Sylvain	PR1	Mécanique et ingénierie	SMS
FAVERGEON	Loïc	CR	Génie des Procédés	SPIN
FEILLET	Dominique	PR1	Génie Industriel	CMP
FOREST	Valérie	MA(MDC)	Génie des Procédés	CIS
FOURNIER	Jacques	Ingénieur chercheur CEA	Microélectronique	CMP
FRACZKIEWICZ	Anna	DR	Sciences et génie des matériaux	SMS
GARCIA	Daniel	MR(DR2)	Génie des Procédés	SPIN
GAVET	Yann	MA(MDC)	Image Vision Signal	CIS
GERINGER	Jean	MA(MDC)	Sciences et génie des matériaux	CIS
GOEURIOT	Dominique	DR	Sciences et génie des matériaux	SMS
GONDRAN	Natacha	MA(MDC)	Sciences et génie de l'environnement	FAYOL
GRAILLOT	Didier	DR	Sciences et génie de l'environnement	SPIN
GROSSEAU	Philippe	DR	Génie des Procédés	SPIN
GRUY	Frédéric	PR1	Génie des Procédés	SPIN
GUY	Bernard	DR	Sciences de la Terre	SPIN
HAN	Woo-Suck	MR	Mécanique et ingénierie	SMS
HERRI	Jean Michel	PR1	Génie des Procédés	SPIN
KERMOUCHE	Guillaume	PR2	Mécanique et Ingénierie	SMS
KLOCKER	Helmut	DR	Sciences et génie des matériaux	SMS
LAFOREST	Valérie	MR(DR2)	Sciences et génie de l'environnement	FAYOL
LERICHE	Rodolphe	CR	Mécanique et ingénierie	FAYOL
MALLIARAS	Georges	PR1	Microélectronique	CMP
MOLIMARD	Jérôme	PR2	Mécanique et ingénierie	CIS
MOUTTE	Jacques	CR	Génie des Procédés	SPIN
NIKOLOVSKI	Jean-Pierre	Ingénieur de recherche	Mécanique et ingénierie	CMP
NORTIER	Patrice	PR1		SPIN
OWENS	Rosin	MA(MDC)	Microélectronique	CMP
PERES	Véronique	MR	Génie des Procédés	SPIN
PICARD	Gauthier	MA(MDC)	Informatique	FAYOL
PIJOLAT	Christophe	PR0	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR1	Génie des Procédés	SPIN
PINOLI	Jean Charles	PR0	Image Vision Signal	CIS
POURCHEZ	Jérémy	MR	Génie des Procédés	CIS
ROBISSON	Bruno	Ingénieur de recherche	Microélectronique	CMP
ROUSSY	Agnès	MA(MDC)	Génie industriel	CMP
ROUSTANT	Olivier	MA(MDC)	Mathématiques appliquées	FAYOL
STOLARZ	Jacques	CR	Sciences et génie des matériaux	SMS
TRIA	Assia	Ingénieur de recherche	Microélectronique	CMP
VALDIVIESO	François	PR2	Sciences et génie des matériaux	SMS
VIRICELLE	Jean Paul	DR	Génie des Procédés	SPIN
WOLSKI	Krzystof	DR	Sciences et génie des matériaux	SMS
XIE	Xiaolan	PR1	Génie industriel	CIS
YUGMA	Gallian	CR	Génie industriel	CMP

Mise à jour : 01/02/2016

"Ideas are of themselves extraordinarily valuable, but an idea is just an idea. Almost any can think up an idea. The thing that counts is developing it into a practical product."

Henry Ford, My life and work, 1923, NY

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Alexandre Dolgui for the continuous support of my Ph.D study. His guidance helped me to solve both research problems and problems of everyday life. I could not have imagined having a better advisor and mentor for my Ph.D study.

I place on record, my sincere thank you to my co-supervisor Dr. Xavier Delorme, for the useful comments, remarks and engagement through the learning process of this Ph.D thesis.

My sincere thanks also goes to my co-supervisor Prof. Mikhail Y. Kovalyov, who supported me during my stay in Belarus. I am extremely thankful and indebted to him for sharing expertise, and sincere and valuable guidance.

Besides my advisors, I would like to thank the rest of my thesis committee: Prof. Farouk Yalaoui, Prof. Lyés Benyoucef, Prof. Alain Quilliot.

Furthermore, I would like to thank all the members of amePLM project for providing me an opportunity to bring my insights to the project.

I thank my fellow labmates in for the stimulating discussions and all the time we spent together.

Last but not the least, I would like to thank all my family. Special thanks to my dad for the inspiration to start a thesis. And of course to my wife for providing me with unfailing support throughout my years of study and through the process of researching and writing this thesis.

Contents

List of Figures

List of Tables

General Introduction

In recent years, many organisations consider automation as a tool that can help to reduce costs due to the optimisation of production process. But before starting automation they need to revise their existing processes, optimise them, make them more effective and after that begin automation. This approach brought a success to Henry Ford when he designed his first assembly line. He did not operate the term of assembly line and he did not think about that. But he wanted to make assembly of the Model T car more efficient and he found the way how to achieve it. His contributions were: the use of interchangeable parts where every part can, without any modification, be moved from any car to another; the use the conveyor, where the machines are set up in the proper order for manufacturing. He standardised the car. The idea for the Model T cars was that they are all the same.

Nowadays, problems of automation, design and optimisation of production systems become even more important and popular as a result of the world wide competition and due to the rapid progress of manufacturing technologies. Their efficient solutions are demanded. Many new characteristics and properties of production systems are appearing. Therefore, the industries require design and re-design of production lines and, hence, new methods of mathematical modelling are needed.

These problems attract more and more attention from academic researchers. Huge amount of journal papers and monographs exist in the considered field. Numerous new articles are published each year.

In Chapter 1, an overview of assembly lines, transfer lines and flow lines is given, and differences and similarities between them are outlined. Many results are obtained in the considered field, however, many scientific and practical problems are still unsolved. The aim of this thesis is to solve two new important problems in this field, which were not studied before. A broad survey of problems of flow line design, including those related to the problems solved in this thesis, is given in Chapter 1.

In Chapter 2, a workforce minimisation problem for a semi-automated multi-product sequential assembly line is studied. In this problem, several

identical workers can be assigned to the same station. A chain precedence graph and task times, inversely proportional to the number of workers, are considered. Algorithms and their computer implementations in modern programming language are developed.

In Chapter 3, a problem, more general then the problem in Chapter 2, with arbitrary acyclic precedence constrains and task times, non-decreasing in the number of assigned workers is studied.

In Chapter 4, methods, algorithms and their computer implementation for a multi-objective transfer line balancing problem, appearing in a single product serial production line, are described.

In Chapter 5, an application of the results obtained in Chapter 2 to solving real production problems related to the european project amePLM is described. Objectives and results of the project amePLM, with a special attention to the workforce optimisation problem are presented.

Chapter 1

Flow lines and related problems and solution methods

1.1 Introduction

Manufacturers from different industries (like automotive production, electronics, etc.) are increasingly interested in the optimisation of their production systems in order to remain competitive. The optimisation objectives can be different. There can be a single criteria optimisation of total investment cost, production space, throughput, changes frequency or number of workers. Multi-objective optimisation with several criteria of production needs satisfaction are also relevant and useful.

Assembly line process is a manufacturing process in which some parts or semi-products are combined to form a final product. Each half-finished product moves in the same direction through sequentially arranged workstations. The stations are connected by a transporting system. The whole system is a conveyor type. As soon as a workstation is released a new half-finished product can enter it. The production cycle includes operations on all workstations related to the same product. If all half-finished products move from the current workstation to the next one with the same time step, then the production line is called paced. Otherwise, it is called unpaced. The length of the mentioned time step for the paced line is called cycle time, see Dolgui and Proth [34].

Historically, the first assembly lines were human operated. Lately, steam and electric engined conveyors were employed to move parts from one workstation to another. Nowadays, assembly lines are mainly automated or semi-automated, however, human operated lines still exist.

The design of assembly lines was improved by Henry Ford by introducing moving platforms to the conveyor system that produced Ford Model T, starting from December 1, 1913 [50]. In this system, the chassis of

the vehicle was towed by a rope that moved it from one station to another where workers were assembling car parts. By using this technology, each car unit was produced every 93 minutes, with a total amount of around two million units in a year.

During the next historical period assembly lines were evolving a lot. In 1950s-1960s, robots were invented and intensively introduced into the industrial process. Robots are still in use in modern assembly lines.

The Digital Revolution that started from the late 1950s to the late 1970s changed mechanical and analogue electronic technology to digital electronics technology and had a positive effect on manufacturing processes. An important event was the introduction of Computer Numeric Control (CNC), see Dolgui and Proth [35]. CNC machining implies the use of computers to control machine tools and operates them by precisely programmed commands encoded on a storage medium (computer command module, usually located on the device), which is opposed to the manual and mechanical control and operation. Numerical control allowed execution of advanced and precise operations, including parallel operations, and production of larger quantities of complex products, see Dolgui et al. [40].

There is a terminological difference between the notions of the assembly line and transfer line. While the assembly line assumes assembly operations as a major ones, transfer line can be intended for various operations including, for example, assembly operations, drilling holes, milling of shapes, tapping, boring, painting, cleaning, etc. Transfer lines are usually paced and serial. They consist of a sequence of stations linked by an automated material handling device. Each station is equipped with a special tools which perform blocks of operations. All operations of a block are usually executed simultaneously. When machining at the current station is finished (all blocks of operations assigned to this machine have been executed) the part is moved to the next station.

There are 3 principal types of automated transfer machining lines: dedicated transfer line (DTL), flexible transfer line (FTL) and reconfigurable transfer line (RTL). Each of these lines has its own characteristics. Dedicated transfer lines are used for the production of a single type product in huge amounts (large quantity of identical products are manufactured with the same sequence of operations). Therefore, DTL has high productivity (see Figure 1.1). Flexible transfer line can produce several types of products, that have comparable dimensions and geometric characteristics.

Frequent changes are specific for these lines (see Figure 1.1). In this thesis, we focus on reconfigurable transfer lines used in reconfigurable manufacturing systems (RMS). The latter notion is introduced by Koren et al. [64]. RMS allow easy changes of their physical configuration to cope with market needs in both volume and product type. RTL combines positive features of DTL and FTL by providing shorter cycle time and on the other hand allows for multiple product types (see Figure 1.1).

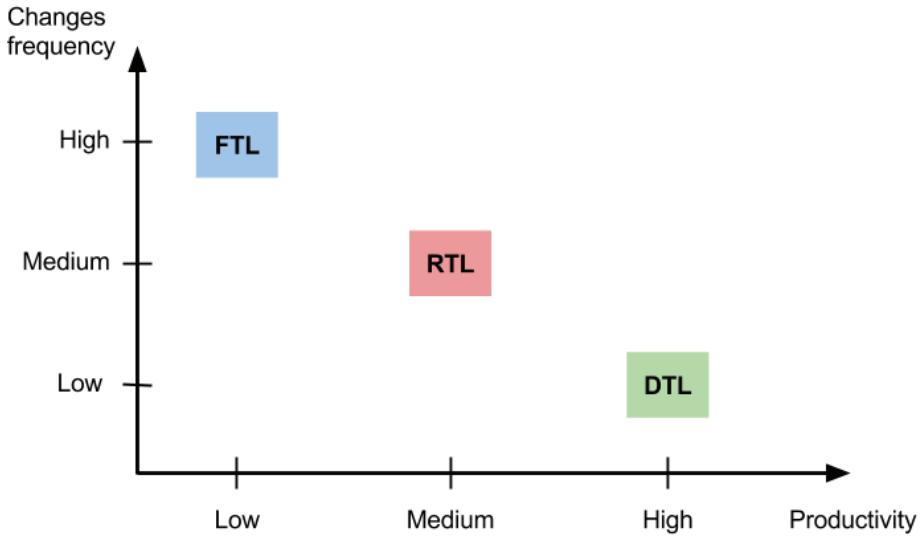


FIGURE 1.1: Main characteristics of different types of transfer lines

In the US literature, large series assembly lines and transfer lines are sometimes considered as synonyms. Also, the flow line term is sometimes used to generalise both assembly line and transfer lines terms. In this thesis, assembly lines are considered as the lines intended for assembly operations. And transfer lines are considered as the lines where various machining operation are executed.

In section 1.2, general problems of the design of flow lines are discussed. In section 1.3, specific problems of flow line design are discussed an their solution methods are reviewed.

1.2 General problems of flow line design

Designing a flow line is a complex decision problem, which requires many important decisions affecting the manufacturing productivity and the cost of the product. The real life flow line design problems can not be solved by unique optimisation procedure or a single person. The design process is usually split into several stages which are interrelated with their input and output data, see Dolgui and Proth [35].

Some of the general flow line design problems are:

- product analysis and design (the result is basic operations required to manufacture the final product);
- process planning (the result is a single production process selected from a set of all alternative processes. The latter set can be represented by a collection of relations on the set of basic operations and their physical, temporal and value characteristics);
- line configuration (the result is configuration type of the line, which can be serial line, parallel lines, U-line, etc.);
- line balancing (the result is the assignment of operation to the workstations and their sequences);
- workforce assignment (the result is assignment of workers to the operations that require workforce);
- part sequencing for multi-model lines (the result is a product sequence);
- equipment selection (the result is a single set of required equipment from the set of alternative equipment);
- equipment dimensioning (the result is the capacities of workstations and aggregated tools).

Some of the described problems are solved based on manufacturing experience. On the other hand models, methods and software exist to support solving these problems in specific environments. Since there exist many different production environments and product requirement is it practically not possible to provide a mathematical model, method and software for any specific problem of the flow line design. This thesis is

devoted to solving two new problems of line balancing and workforce assignment. An overview of line balancing and workforce assignment problems for which there exist mathematical algorithms and software tools is given in section 1.3. The relation between these early studied problems and problems in this thesis are outlined. Problems and solution methods which are the topic of this thesis are presented in chapters 2, 3 and 4.

1.3 Line balancing and workforce assignment problems and their solution methods

Optimisation for different types of production systems is considered in this work. We start with the line balancing problem (assignment of operations to workstations), which is an important problem in the design of flow lines. The optimisation of production systems is an important stage for manufacturers to minimise costs and remain competitive. There are different types of line balancing problems. The line balancing problem was first studied for assembly lines. The name of Assembly Line Balancing problem (ALBP) and its first scientific research is attributed to Salveson [80] in 1955. ALBP was also considered by Baybars [11]. Later on the name of this problem was changed to the simple line balancing problem (SALBP). The goal of SALBP is the minimisation of the number of workstations for a given cycle time (SALBP-1) or the minimisation of cycle time for a given number of workstations (SALBP-2). Various approximate and exact methods were developed for this problem. One of the first proposed heuristics was called Ranked Positional Weight (RPW). It was proposed by Helgeson and Birnie [52] in 1961. The main idea of this heuristic is to give priority for execution to the operations from the longest chains in the precedence graph. Several years later, a Computer Method for Sequencing Operations on the Assembly Lines (COMSOAL) was proposed by Arcus [5] in 1966. The idea behind it is to generate a large set of feasible solutions, and then to choose the best solution from them.

There are many extensions of SALBP. The Sequence-Dependent Assembly Line Balancing Problem (SDALBP) has a significant relevance to practical to the real asembly lines and extends the basic problem by considering sequence-dependent operation times (see Scholl et al [82]). Another extension is the Parallel Assembly Line Balancing Problem (PALBP), which considers a production system consisting of a number of parallel

assembly lines. A given product is manufactured on each line and a common cycle time is observed. This problem has been considered by Gökçen et al. [56].

Another generalisation of SALBP is denoted as SALBP-G. It is obtained by minimising the sum of idle times, subject to varying production rates and numbers of stations (Scholl [81]). The U-Assembly Line Balancing Problem (UALBP) considers the case of U-shaped assembly lines for a single product. There, workers are assigned either side of the U, i.e., they perform early and late operations in the production process Gökçen and Ağpak [57]. There exist several extensive surveys of the assembly line balansing problems, see for example Boysen et al. [23] [22]. The paper of Rekiek et al. [78] is a review of the assembly line design problems, with a special attention paid to the line balancing and resource planning. A survey of exact methods, heuristics and metaheuristics to solve these problems is given.

Another type of line balancing problems is associated with the Transfer Line Balancing Problem (TLBP). In most works that consider TLBP, the line is composed of sequentially arranged workstations and a transport system which ensures a constant flow of parts along the workstations. Each workstation consists of several identical machines (machining centres). All machines have a set of machine-tools and can execute different processing operations. Each operation is characterised by an operational time. Besides the operation times, some of the problems consider unproductive times between the operations (set-up times), Essafi et al. [46]. These set-up times are needed to displace or change tools or to displace or rotate the part. In such a line, a repeatable set of operations is executed in each production cycle. The set of operations to be performed on the transfer line is determined by the technological process for which the line is designed. These machining lines produce large series of identical or similar items.

The solution of the TLBP answers the following questions:

- Which machining units are to be chosen to execute the required operations?
- How many workstations are necessary?
- How should the machining units be assigned to the stations?

These questions are answered so the total equipment and production costs are minimised for a given production cycle time.

In comparison with ALBP, the TLBP has a number of additional characteristics such as parameterised operation times, parallel operation execution at the same workstations, and so on. The specificity of the ALBP is that there are no restrictions other than precedence constraints. The TLBP is considers different types of constraints on the set of operations such as:

- Precedence constraints define a partial order between the operations.
- Inclusion constraints define sets of operations such that all operations of the same set must be executed on the same workstation.
- Exclusion constraints define sets of operations such that all operations of the same set can not be executed on the same workstation, but any proper subset of the same set can be executed on the same worstation.
- Accessibility constraints specify that some sides of the part are only accessible in certain positions of the part.

The additional characteristics of the TLBP do not allow the use of the optimisation methods developed for the ALBP directly. The TLBP was defined and studied in Dolgui et al. [38], where several exact, approximate and heuristic methods for solving TLBP were proposed. The most significant methods for an exact resolution of TLBP are linear programming (Belmokhtar et al [16]), dynamic programming (Dolgui et al [38]), mixed integer approach (Dolgui et al [37], Essafi et al [47]) and branch and bound procedures (Ihnatsenka et al [36]). Exact methods are useful to better understand the problem, however, for large scale problems they often require an unreasonable computation time. Contrarily, approximate methods provide fast solution but they do not guarantee solution optimality. Additionally, heuristic algorithms are often easier to implement than the optimal procedures. Several approximate methods were developed for large scale problems: Priority Rules heuristics (Finel et al. [49]), a Heuristic Multi-Start Decomposition approach (Guschinskaya et al. [58]) and Ant Colony Optimisation algorithm (Essafi et al [48]), metaheuristics such as GRASP method and a Genetic Algorithm (Eremeev et al [41]). Essafi et al. [46] proposed heuristic based on the GRASP method combined with path relinking and MIP approach to select sequences of operations on workstations. In the paper of Borisovsky et al. [20] suggested a Genetic Algorithm based on the permutation representation of solutions. An additional local improvement step is introduced, which uses a special MIP

model to determine an optimal composition of workstations if the order of operations is fixed.

Workforce assignment is another important problem arising in flow line design. In this problem, there is a flow line that produces various products and consists of several assembly stations. Each product or semi-product requires a specific set of operations to be performed by workers. The assignment of operations to the stations is known. Operations assigned to different stations can be performed in any order and those assigned to the same station should be performed according to the given precedence relations. No worker can perform more than one operation at a time. It is assumed that workers can switch between the stations at zero time. The problem is to assign workers to operations so that the total number of workers is minimised, provided that a given cycle time is satisfied.

Literature related to the optimal workforce assignment can be partitioned into several categories. The first category includes publications on the classic assignment problems of combinatorial optimisation, in which there is a cost of assigning worker i to operation j and the objective is to match workers with the operations so that the total cost is minimised. The most important results of this category are described in the monograph of Burkard et al. [24]. The second category can be characterised by the keywords *timetabling, rostering* and *shift scheduling*. Example studies of this type can be found in Burke and Trick [25] and Jiang et al. [51]. The third category is the *resource constrained project scheduling*. A good source of information on this category is the book of Artigues et al. [6].

There exists a vast body of the literature on workforce assignment to operations of a production line. Vidic [90] studies the effect of the assignment of a fully cross-trained workforce on the throughput of a serial production line. She focuses on dynamic work sharing and fixed workforce assignment and suggests two models. One model assumes that workers' performance is determined by their steady-state productivity rate and the other model assumes that workers' productivity rates depend of their learning and forgetting characteristics. Heuristic methods are developed.

A recent review and classification of the literature related to workforce planning problems with skills is presented by De Bruecker et al. [31]. Corominas et al. [30] study a problem in which skilled permanent and unskilled temporary workers have to be assigned to an assembly line. Skilled workers require less time to finish an operation (task) than unskilled ones,

and unskilled workers must work alongside at least one skilled worker. The criterion is to minimise the number of unskilled workers. The authors suggest a binary linear programming formulation for this problem.

In the paper of Blum and Miralles [18] a generalisation of SALBP-2 which is called the Assembly Line Worker Assignment and Balancing Problem (ALWABP-2) with the objective of minimising the cycle time is considered. The difference with the classic SALBP-2 is that operations must be assigned to workers,and workers to workstations.Task processing times are worker specific, and workers might even be incompatible with certain tasks. Authors employ beam search heuristics for that problem.

Araujo et al. [4] consider Assembly Line Worker Assignment and Balancing Problem (ALWABP) with non-identical workers. Two cases of this problem are studied: parallel workstations (one worker is assigned to the same station) and collaborative workstations (several workers are assigned to the same station).

Borba and Ritt [19] propose a MIP model for ALWABP-2, a heuristic algorithm based on a beam search, and a task oriented branch-and-bound procedure for the same problem with the non-identical workers.

An extension of the ALWABP to minimise the expected cycle time under uncertain worker availability is proposed by Ritt et al. [77].

Moreira and Costa [73] studies ALWABP problem for non-identical workers and propose hybrid algorithm that uses a Mixed Integer Programming (MIP) to select appropriate schedules from a pool of heuristically constructed solutions. A local search based on MIP neighbourhoods is used as a post-optimisation method.

Iterative Genetic Algorithm (IGA) for ALWABP-2 with the cycle time minimisation criterion was developed by Mutlu et al. [74]. In the IGA, three search approaches are adopted in order to obtain search diversity and efficiency: a modified bisection search, a Genetic Algorithm and Iterated Local Search.

Vilá and Pereira [91] propose an exact enumeration procedure for the same problem. The new lower bounds allowed the authors to improve results for a benchmark set of instances.

A multi-product semi-automated assembly line from automotive industry with temporary workers and with non-neglected setup-times was considered by Giard and Jeune [55]. Multi-objective problem of optimisation workers cost and setup cost simultaneously was solved by employing exact methods that allow obtain optimal solutions for instances with up to

15 items and satisfactory feasible solutions for some of the larger problems within a reasonable time limit.

A new assembly line problem with qualification requirements of operations and qualification levels of workers is introduced by Sungur and Yavuz [85]. In the hierarchical workforce structure, a lower qualified worker can be substituted by a higher qualified one (but not vice versa) with a certain cost. The objective is to minimise the total cost, while respecting a given cycle time. An ILP model is used to solve this problem.

The part supply scheduling problem in line-integrated supermarkets with the workforce minimisation criterion and no-stockouts constraint is studied by Boysen and Emde [21]. The authors use a heuristic decomposition approach to solve this problem and draw important conclusions for managers.

A scheduling problem with the criterion of workforce minimisation is treated by Camm et al. [26]. The authors deal with a paced line composed of identical parallel workstations. The number of workers needed at a workstation depends on the job processed and it is fixed. A two-stage approach is applied. The first stage is a MILP model that determines starting times. The second stage is a polynomial time procedure which assigns jobs to specific assembly workstations.

There is a stream of publications, in which workforce requirements of the operations are assumed to be fixed, there is one operation for any product on each station of the transfer line, and the problem is to find a cyclic sequence of products such that the maximum number of workers needed at any time is minimised. The relevant results can be found in Akagi et al. [3], Wilson [94], Lutz and Davis [70], Lee and Vairaktarakis [67], Vairaktarakis and Winch [89], Kouvelis and Karabati [65], Vairaktarakis et al. [88], Vairaktarakis and Cai [87] and Kovalyov et al. [63].

In this thesis, we used the ideas of the GRASP approaches for single objective TLBP, which are given provided in Eremeev et al. [41] Essafi et al. [46] Borisovsky et al.[20]. The novelty of the results in this thesis is the application of the earlier methods for the multi-objective TLBP with the goal to minimise the total investment cost and the cycle time. Our approaches and their combination in a metaheuristic are described in Chapter 4.

Problems with non-identical workers have been widely studied in the literature. This thesis considers problems with identical workers and an assumption that operation time depends on the number of workers assigned to this operation. Two different problems are studied in Chapters 2

and 3. The problem in Chapter 2 assumes that the precedence contain have the form of chains and that the operation time is inversely proportional to the number of workers. The problem in Chapter 3 is more general. ,Is considered arbitrary precedence constrains and arbitrary non decreasing function of operation times depending on the number of workers. Several exact and approximate methods are proposed to solve these problems. They will be described in more details in Chapters 2 and 3.

1.4 The issues of computer implementation and their resolution

Nowadays, solutions of real production problems can not be obtained without computers. A proper efficient computer implementation of mathematical models and algorithms is an important issue, because, solution quality and the required computer resources (running time and memory) are the characteristics that are the most important for the customers. Implementation includes selection and realisation of computer science subroutines such as sorting, searching in data structures, basic graph algorithms and so on, which are represented as blocks of commands in a particular programming language. The efficiency of the implementation depends on the efficiency of all the sub-routings. Which is also confirmed by Cormen et al. [29]. There can be an inefficient implementation of a theoretically efficient algorithm and an efficient implementation of a theoretically inefficient algorithm for the same problem, which can led to the incorrect interpretations when comparing experimental results the two mentioned algorithms.

In the paper of Segal and Morris [83], a difference between scientific and commercial software development is outlined. They stress that, due to the insufficient scientific knowledge of the average software engineers, the scientific researchers often implement their algorithms themselves.

Paper of Hannay et al. [59] presents the results of a an online survey conducted in October-December 2008, which received almost 2000 responses. Main conclusions of their paper are that 1) the knowledge required to develop scientific software is primarily acquired from peers and thought self-study, rather than from formal education and training; 2) while many scientists believe that software testing is important, a rare

number of them know the principals and organisation of adequate computer testing.

Most of mathematical models and algorithms are published and pass through a review process. However, their computer implementations are usually not reviewed, because they are usually not a part of the article. Thus the readers can only rely on the statements drawn by authors. Lessons learned during implementations are usually not published as the results, and therefore, they are often not shared, see Carver and Epperly [27].

Ku and Beck [66] compare performance of several MIP models implemented by commercial solvers CPLEX and GUROBI and non-commercial solver SCIP for a scheduling problem. In their article, they provide performance results for configurations with 1 and 8 processor threads and various solver parameters. Even though they show that parameters tuning may give performance improvements, they do not describe which parameters were tuned and how.

Almost all scientific researchers make computer implementations of their algorithms in order to solve benchmark or real life problems. Results of those implementations are usually included in the papers, however, computer programming techniques used to make the implementations efficient are usually omitted from the papers and not available to other researchers. In the filed of line balancing and workforce assignment we found no publication in which a technic to improve computer implementation of a solution algorithm is described.

A set of best practices for scientific software development is described in the paper of Wilson et al. [95]. Best practices served to improve research productivity and the reliability and maintainability of the research software. Some of the best practices are: turning bugs into the test cases; using version control system; making incremental changes; using peer code reviews and tracking tools; re-using a verified code segment instead of rewriting it; using build tools to automate workflows; writing human readable code; making names consistent, distinctive and meaningful; making code style and formatting consistent.

Merelo et al [71] is an example of a papers where computer implementation of Evolutionary Algorithms is improved due to the application of best programming practices and usage of profiler program to determine bottlenecks.

In this thesis, an attention is paid not only to the development of the

algorithms, but also to their proper and efficient computer implementations. Our software use best know practices, some of which are described by Wilson et al. [95].

1.5 Conclusions

This chapter gives basic definitions and discusses the terms used in this thesis. Different types of production lines are described in Section 1.1.

Section 1.2 provides an overview of flow line design problems.

Section 1.3 surveys existing methods for the classic line balancing problem SALBP. Extensions of the classical problem by introducing additional characteristics and constraints are reviewed. A special attention is paid to the transfer line balancing problem and workforce assignment problem. A survey of the solution methods for these problems is provided. The problems studied in this thesis are introduced and similarity and difference of these problems with early studied problems are remarked.

The issues of computer implementation of mathematical models and algorithms are discussed in Section 1.4. The lack of descriptions of programming techniques to improve computer implementations of scientific algorithms in the filed of line balancing and workforce assignment is outlined. General best practices for scientific computer program development are mentioned.

Chapter 2

Workforce minimisation for a multi-product assembly line with chain precedence relations

2.1 Introduction

The problem for a serial multi product assembly line with the objective of minimising the number of workers is studied. This problem appeared in a real life industrial situation. There is a *paced unidirectional assembly line* that manufactures different types of products, which are three automobile engine models V12, V16 and V20. A set of *assembly stations* and a set of *sub-assembly stations* of this line are given, at which required operations on the engines are performed in the predetermined sequences. The stations are connected by a unidirectional conveyor. Products enter the line in a given sequence one after another, and each product visits assembly stations in the same order. At the same station, the same operations are performed for any engine.

In a given production *cycle*, operations of different stations are performed in parallel, and operations of the same station are performed sequentially.

Each sub-assembly station is associated with a unique assembly station. An assembly station may have several sub-assembly stations associated with it. In each production cycle, operations of an assembly station are performed on the product that is assembled at this station in this cycle, while operations of sub-assembly stations in this cycle are linked with the product that will arrive to the corresponding assembly station in the next cycle. In one production cycle all operations of sub-assembly stations linked to a certain assembly station must be performed. Thus, if there are several sub-assembly stations associated with an assembly station they

can be considered as a single sub-assembly station. The sequence of the assembly stations to be visited by the semi-finished engines, the sequences of operations to be performed at the assembly and sub-assembly stations are the same for each engine model. However, the operation processing times depend on the engine model. For the same number of workers assigned to an operation, some operations for V20 engine take longer time than those for V16 engine, and some operations for V16 engine take more time than those for V12 engine, while some operations have the same processing times for all engine models if the number of assigned workers is the same.

The assembly line is paced. Therefore, the sum of operation processing times at a station in each cycle must not exceed a given cycle time. The cycle time is determined as the ratio of the total production time available in a year to the planned annual productivity.

The information on the available production time is given in Table 2.1.

TABLE 2.1: Available production time

Days per year	230
Shifts per day	2
Shift duration, hours	8

The annual line productivity and shares of demand for each engine model are determined based on the customers demand analysis. They are given in Table 2.2.

TABLE 2.2: Annual line productivity with model shares

Annual productivity (engines/year)	1450
V12 share	75%
V16 share	20%
V20 share	5%

Thus, the cycle time in hours is calculated as follows:

$$Cycle\ time = \left\lfloor \frac{230\ [days/year] \times 2\ [shifts/day] \times 8\ [hours/shift]}{1450\ [engines/year]} \right\rfloor =$$

$$= 2.5\ [hours/engine]$$

The manufacturer wants to supply the consequent stages of the automobile production with the engines of different types evenly. Due to this requirement, the sequence of model types to enter the line is designed, which counts for the fractions of engine models in the total annual production. These fractions are

$$Fraction\ V12 = 75\% = \frac{3}{4} = \frac{15}{20}$$

$$Fraction\ V16 = 20\% = \frac{1}{5} = \frac{4}{20}$$

$$Fraction\ V20 = 5\% = \frac{1}{20}$$

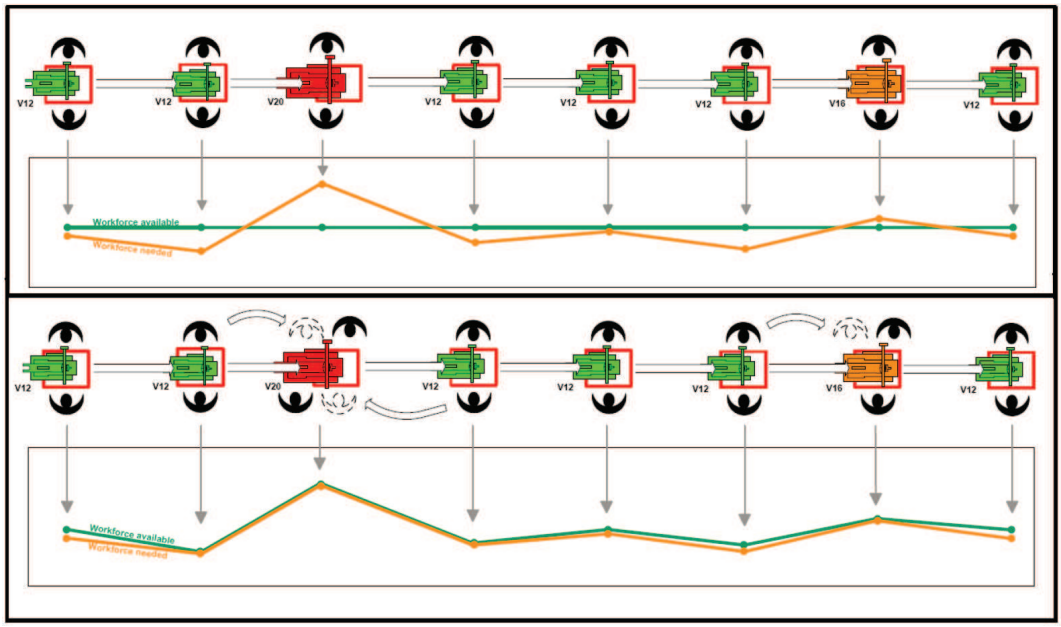
The sequence of evenly distributed engine models adopted by the manufacturer is a repetition of the sub-sequence of 20 models given in Figure 2.1.

FIGURE 2.1: Repeated sub-sequence of evenly distributed engine models

Operations are performed by *identical* workers of a given set. Any worker can perform any operation, but their number assigned to the same operation is upper bounded. If a worker is assigned to an operation, he/she is fully occupied by this operation from its start time till completion. The workers can switch between operations. The switching time is assumed to be zero. If a lack of workforce occurs at some station, then workers from the stations with a surplus of the workforce can move there to help completing operations in time. An example of such a situation is illustrated in Figure 2.2.

The processing time of an operation depends on the number of workers assigned to this operation and the product type. The values of processing times are given for each operation, each possible number of workers and each product type. For the studied industrial case, the processing time of an operation in a given production cycle is inversely proportional to the number of workers assigned to it. Let operation i require p time units if performed by one worker. In a given cycle, denote by p_{im} the processing time of operation i if it is performed by m workers. Then $p_{im} = \frac{p}{m}$. No

FIGURE 2.2: Movement of workers between stations



more than four workers can be assigned to the same operation in the real life situation.

The line *cycle time* is assumed to be given such that the total processing time of all operations at any station in any cycle should not exceed this time.

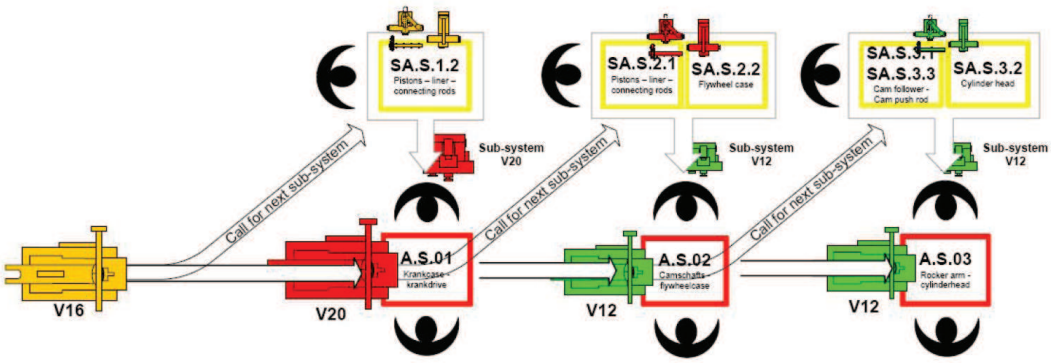
One production cycle is considered. For this cycle, the assignment of products to the assembly and sub-assembly stations is assumed to be given. Therefore, operation processing time may be assumed not to depend on the product type. The problem is to determine an assignment of workers to operations such that the total number of workers is minimised and the line cycle time is not exceeded. We denote this problem as P.

The distance between any two consequent stations is at most 11 meters, and any worker can travel between them in less than 10 seconds, which is only 0.09% of the cycle time. Therefore, we simplify the problem by assuming that any worker can move from one operation to another at zero time.

The sub-assembly stations operate in a *Just-In-Sequence* mode, according to which operations of a sub-assembly station must be completed just before the arrival of a specific engine model to the corresponding assembly station. In order to do this, a *sub-system*, which is a part of the engine

prepared at the sub-assembly stations, must be finished in a cycle that immediately precedes the cycle, in which the engine model will be treated at the corresponding assembly station. In Figure 2.3, engine V20 arrives to the assembly station A.S.01 and, at the same time, the sub-assembly stations SA.S.2.1 and SA.S.2.2 start to prepare a sub-system for engine V20, so that it will be mounted on the engine in the next cycle at the assembly station A.S.2.

FIGURE 2.3: Relation between sub-assembly and assembly stations



There are 170 operations in total to be executed at 11 assembly and 17 sub-assembly stations.

Since the sequence of engine models entering the line is a repetition of a sub-sequence that consists of 20 models, there are 20 distinct cycles which differ by the assignment of engine models to the assembly stations. For each of these 20 cycles, the problem is to find an assignment of workers to the 170 operations such that the total number of workers is minimised and the cycle time of 2.5 hours is satisfied.

2.2 Product Sequencing Problem

Taking into account the input data, we can assume without loss of generality that if the variant V20 is placed first at the line, we only need to find the place for 4 products of type V16 and 15 products of type V12. As it is

known from combinatorics, the total amount of such arrangements can be computed as follows:

$$\binom{4}{19} = \binom{15}{19} = \frac{19\,!}{4\,! * 15\,!} = 3876. \tag{2.1}$$

One can see that is not large amount of possible sequences, which means that we can apply an exhaustive search to find the best sequence of variants.

While constructing all possible sequences, those with the minimal theoretical number of workers are selected. This number N_{min} is calculated by dividing the maximal total working time for the sequence T_{total} by cycle time.

$$N_{min} = \left\lceil \frac{T_{total}}{T_{cycle}} \right\rceil = \left\lceil \frac{T_{total}}{150min} \right\rceil. \tag{2.2}$$

After applying the exhaustive search the following optimal sequence shown on Figure 2.4 was obtained:

FIGURE 2.4: Best product sequence

With such a sequence the most demanding configuration requires 59 h 42 min, which means that the minimal bound for the number of workers is

$$N_{min} = \left\lceil \frac{T_{total}}{T_{cycle}} \right\rceil = \left\lceil \frac{59h42min}{150min} \right\rceil = 24. \tag{2.3}$$

However, one can see that the sequences with workforce demand for all configurations not exceeding 60 hours will also provide the same minimal bound of 24 workers. Therefore, all sequences (Figure 2.5) providing the same minimal bound are processed in next step.

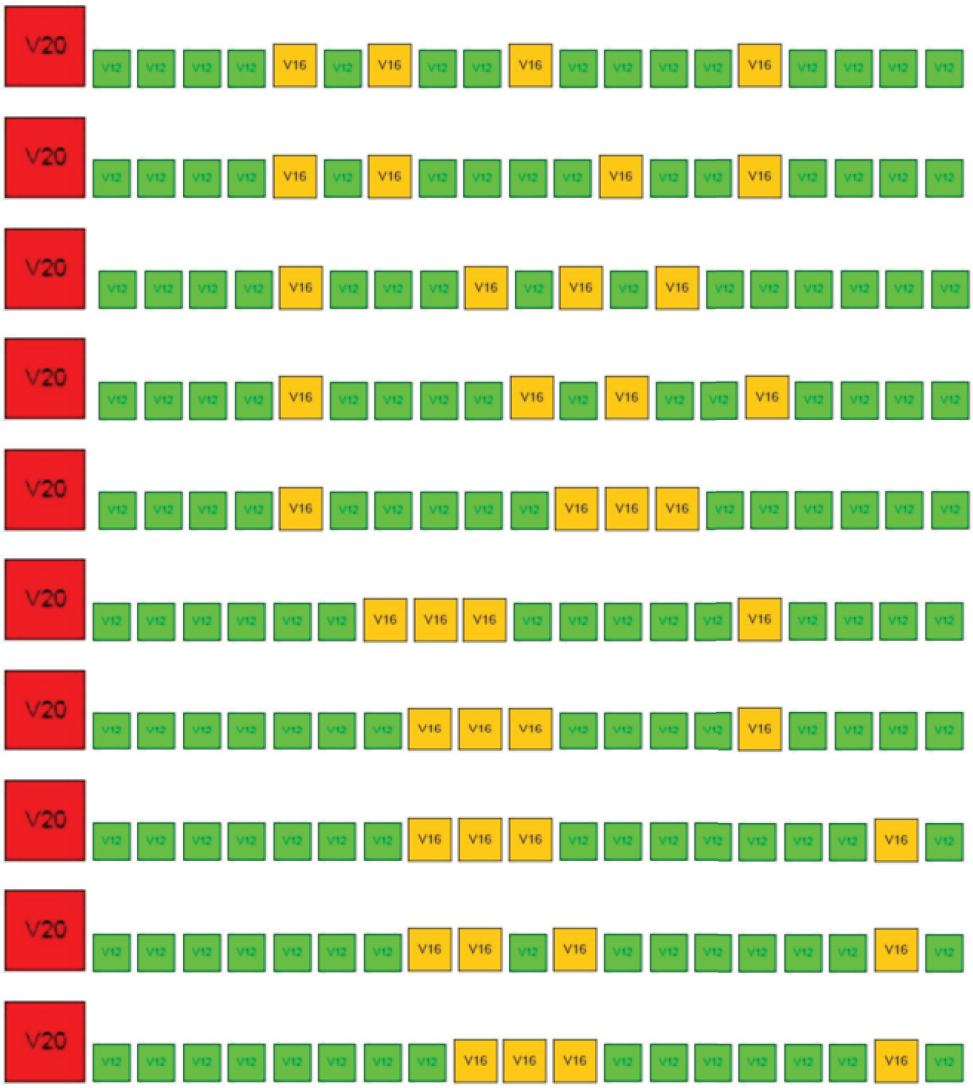


FIGURE 2.5: All product sequence with minimal bound of 24 workers

Because of discrete tasks times, any of these 10 sequences may provide the optimal line configuration with the minimal number of workers. However, if configuration with 24 workers is not feasible, all the sequences providing the lower bound of 25 workers can be examined during the process of searching optimal workforce assignment.

Note that even a slight increase in the number of product types or changing the number of each product type results in a dramatically increase of the number of possible product sequences, which leads to the

impossibility of applying an exhaustive search procedure.

2.3 MILP formulation

A *Mixed-Integer Linear Programming (MILP)* model was developed for the workers assignment problem. In the industrial case, the model is the same for each of the 20 cycles determined by the entering sequence of evenly distributed engine types, but the input data can be different for different cycles. The following notation is used.

GIVEN SETS:

- W - the set of available workers;
- I - the set of operations;
- L - the set of last operations at all stations;
- O - the set of ordered pairs of operations (i, j) such that operation i precedes operation j on their station, not necessarily immediately;
- O' - the set of ordered pairs of operations (i, j) such that operation i immediately precedes operation j on their station;
- D - the set of unordered pairs of operations $\{i, j\}$ such that they are assigned to different stations.

INDICES:

- i or j - an operation;
- w - a worker;
- m - the number of workers assigned to an operation.

GIVEN PARAMETERS:

- c - the cycle time;
- w_{max} - the cardinality of the set W of available workers, $w_{max} = |W|$;
- m_{max} - an upper bound on the number of workers assigned to the same operation, $m_{max} \leq w_{max}$;

- p_{im} - the processing time of operation i, if it is performed by m workers, $i \in I$, $m = 1, \ldots, m_{max}$.

DECISION VARIABLES:

- S_i - the start time of operation i;
- x_{iw}= {1, if i is the first operation performed by worker w, 0 otherwise};
- x_{ijw} = {1, if worker w performs operations i and j, and no other operation between them, 0 otherwise};
- u_{ij} = {1, if operations i and j are performed by the same any worker, and this worker performs no other operation between them, 0 otherwise};
- y_{im} = {1, if operation i is performed by m workers, 0 otherwise};
- z_w = {1, if worker w is used, 0 otherwise}.

Let M be a sufficiently large positive number. Our MILP model is the following.

$$\min \sum_{w=1}^{w_{max}} z_w,$$

subject to

$$S_i + \sum_{m=1}^{m_{max}} p_{im} y_{im} \leq C, \; i \in L, \tag{2.4}$$

$$S_j - \sum_{m=1}^{m_{max}} p_{im} y_{im} \geq S_i, \; (i,j) \in O', \tag{2.5}$$

$$u_{ij} + u_{ji} \leq 1, \; i,j \in I, \; i \neq j, \tag{2.6}$$

$$x_{ijw} + x_{jiw} \leq 1, \; i,j \in I, \; i \neq j, \; w \in W, \tag{2.7}$$

$$\sum_{i \in I} x_{ijw} \leq 1, \; j \in I, \; w \in W, \tag{2.8}$$

$$\sum_{i \in I} x_{jiw} \leq 1, \; j \in I, \; w \in W, \tag{2.9}$$

$$u_{ji} = 0, \; (i,j) \in O, \tag{2.10}$$

$$S_j - S_i \geq \sum_{m=1}^{m_{max}} p_{im} y_{im} - M(1 - u_{ij}), \; \{i,j\} \in D, \tag{2.11}$$

$$\sum_{w=1}^{w_{max}} x_{ijw} \geq u_{ij}, \; \{i,j\} \in D, \tag{2.12}$$

$$\sum_{w=1}^{w_{max}} x_{ijw} \leq w_{max} u_{ij}, \; \{i,j\} \in D, \tag{2.13}$$

$$\sum_{m=1}^{m_{max}} y_{im} = 1, \; i \in I, \tag{2.14}$$

$$\sum_{i \in I} x_{iw} = z_w, \; w \in W, \tag{2.15}$$

$$x_{iw} + \sum_{j \in I, j \neq i} x_{jiw} \geq \sum_{j \in I, j \neq i} x_{ijw}, \; i \in I, \; w \in W, \tag{2.16}$$

$$\sum_{m=1}^{m_{max}} m y_{im} = \sum_{w=1}^{w_{max}} x_{iw} + \sum_{j \in I, j \neq i} \sum_{w=1}^{w_{max}} x_{jiw}, \; i \in I, \tag{2.17}$$

$$z_w \geq x_{ijw}, \; w \in W, \; i,j \in I, \tag{2.18}$$

$$z_w \geq z_{w+1}, \; w \in W, \; w \neq w_{max}, \tag{2.19}$$

$$x_{iw}, x_{ijw}, u_{ij}, y_{im}, y_w \in \{0,1\}, \; i,j \in I, \; w \in W, \; m = 1, \ldots, m_{max}, \tag{2.20}$$

$$S_i \geq 0, \; i \in I. \tag{2.21}$$

The constraints (2.4) ensure that the cycle time limit is not exceeded. These constraints can be extended by considering $i \in I$ if it will accelerate the solver.

The constraints (2.5) guarantee that if operation i immediately precedes

operation j on a station, then it completes before or at the start of operation j. These constraints can be extended by considering $i \in O$ if it will accelerate the solver.

The constraints (2.6) exclude the case that $u_{ij} = 1$ and $u_{ji} = 1$, and the constraints (2.7) exclude the case that $x_{ijw} = 1$ and $x_{jiw} = 1$ for any operations i and j and any worker w.

By the constraints (2.8) and (2.9), any worker can switch to an operation from at most one other operation, and he can switch from an operation to at most one operation.

The constraints (2.10) state that no worker can perform operation j before operation i, if i precedes j on a station.

The constraints (2.11)-(2.13) consider operations i and j that are on different stations. The constraints (2.11) assure that if i and j are decided to be performed by the same worker in the order (i, j) with no other operation between them, i.e., $u_{ij} = 1$, then the start time of operation i plus its processing time does not exceed the start time of operation j.

The constraints (2.12) state that, if i and j are decided to be performed by the same worker in the order (i, j) with no other operation between them, i.e., $u_{ij} = 1$, then at least one worker should be this worker.

The constraints (2.13) require that, if i and j are decided not to be performed by the same worker in the order (i, j) with no other operation between them, i.e., $u_{ij} = 0$, then no worker can perform these operations in this order with no operation between them.

The constraints (2.14) ensure that the number of workers assigned to an operation is unique.

The constraints (2.15) state that, if a worker is chosen, then he should be assigned his first operation, and, if he is not chosen, then no first operation should be assigned to him.

The constraints (2.16) require that if some worker performs operations i and j in the order (i, j) with no other operation between them ($\sum_{j \in I, j \neq i} x_{ijw} = 1$ for a worker w), then either i is its first operation ($x_{iw} = 1$) or he performs some other operation immediately before operation i ($\sum_{j \in I, j \neq i} x_{jiw} = 1$).

On the left hand side of the constraints (2.17), there is the number of workers assigned to operation i. On the right hand side, the first sum is the number of workers for whom i is the first operation. The double sum counts for the number of workers that switch to i immediately from some other operations. It is required that the values in both sides are equal.

The constraints (2.18) state that if worker w is not used, i.e., $z_w = 0$, then there is no operations i and j such that this worker switches to j immediately after i.

The constraints (2.19) break the symmetry by requiring that only consecutively numbered workers are used.

2.4 Heuristic methods

In order to solve problem P described in section 2.1, we developed several heuristics. The first heuristic is simple. It determines the number of workers for each station. They perform every task of their station and do not move to another station. This heuristic gives the same solution as the solution proposed by our industrial partner.

Let s be the number of stations and let I_q be the given set of tasks of station q, $q = 1, \ldots, s$. Let m_q denote the number of workers assigned to station q, which has to be determined, $m_q = 1, \ldots, m_{max}$.

2.4.1 Heuristic Same-Station

- **Step 1** Calculate $m_q = \min\{m | \sum_{j \in I_q} p_j(m) \leq C\}, q = 1, \ldots, s$. Note that, if $\sum_{j \in N} p_j(m_{max}) \leq C\}$, then the bisection search can be employed to find the above minimum in polynomial time for any non-increasing function $p_j(m)$. In each of $O(\log_2 m_{max})$ iterations of the bisection search, the relation $\sum_{j \in I} p_j(m) \leq C$ needs to be verified for a trial value $m \in \{1, \ldots, m_{max}\}$.

- **Step 2** Output a solution, in which m_q consecutively indexed workers perform tasks required for station q and these tasks only, $q = 1, \ldots, s$. If $m_q < m_{max}$ for a $q, 1 \leq q \leq s$, then the solution found is unfeasible. Otherwise, it is feasible with the total number of workers $W^{(1)} = \sum_{q=1}^{s} m_q$.

This algorithm can be applied with any functions $p_j(m)$. For the case $p_j(m) = \frac{p_j}{m}$, with this algorithm we obtain m_q equal to $\lceil \frac{\sum_{j \in I_q} p_j}{c} \rceil$.

2.4.2 Heuristic Sequential-Stations

Our second heuristic, denoted as Sequential-Stations, is also applied for $m_q \leq m_{max}, q = 1, 2, \ldots, s$. The heuristic considers stations in a certain

sequence. Let the sequence be $1, 2, \ldots, s$. The heuristic proposes to assign the same arbitrary m_{max} workers to the stations $1, 2, \ldots, q_{max}$, where $q_max \leq s - 1$, so that the workers serve these stations in the indicated order from time zero until the total processing time of the last task of station q_{max} does not exceed the takt time C. Then the assigned workers and the stations $1, \ldots, q_max$ are removed from the problem input, and the process is repeated. If $q_{max} = s$, then $m_0 < m_{max}$ workers can be assigned to the last stations.

- **Step 1** Calculate $m_q = \min\left\{m | \sum_{j \in I_q} p_j(m) \leq C\right\}$, $1 = 1, 2, \ldots s$. If $m_q > m_{max}$ for a $q, 1 \leq q \leq s$, then stop: no feasible solution exists.
- **Step 2** Re-number stations $1, 2, \ldots, s$ in a certain order, for example, such that the sub-assembly stations of station 1 go first, then station 1, then the sub-assembly stations of station 2, station 2, and so on. Set $a = 1$. Initialize the total number of assigned workers $W^{(2)} = 0$.
- **Step 3** Determine $q_{max} = \max_{a \leq q \leq s}\left\{q | \sum_{h=a}^{q} \sum_{j \in I_h} p_j(m_{max}) \leq C\right\}$. If $q_{max} = s$, then go to Step 4. If $q_{max} \leq s - 1$, then perform the following computations. Assign arbitrary m_{max} workers from the set W of available workers to each task of the stations $a, a + 1, \ldots, q_{max}$ so that they serve these stations in the indicated order. Remove the assigned workers from the set W. Re-set $a := q_{max} + 1$ and $W^{(2)} := W^{(2)} + m_{max}$. Repeat Step 3.
- **Step 4** Determine $m_0 = \min_{1 \leq m \leq m_{max}}\left\{m | \sum_{h=a}^{s} \sum_{j \in I_h} p_j(m) \leq C\right\}$ Assign arbitrary m_0 workers to each task of the stations $a, a+1, \ldots, s$ so that they serve these stations in the indicated order. Re-set $W^{(2)} := W^{(2)} + m_0$. Output the final solution and its value $W^{(2)}$

2.4.3 Heuristic Sequential-Stations-Random

The third heuristic, denoted as Sequential-Stations-Random, differs from the heuristic Sequential-Stations in that the sequence of stations is generated at random

2.4.4 Heuristic Sequential-One-Traveling-Worker

Let s be the number of stations and let I_q be the given set of tasks required at station $q, q = 1, \ldots, s$. Let m_q denote the number of workers assigned

to station q, which has to be determined, $q = 1, \ldots, m_{max}$, where i or j is a task, $i, j \in I$.

If the number of workers m_q, performing tasks of station q with a given cycle time C, is constant, then we can evaluate the minimum number of workers m_q required to perform the given set of tasks I_q at station q for a given tact time C:

$$m_q = \min \{m | \sum_{j \in I_q} p_j(m) \leq C\}, q = 1, \ldots, s \tag{2.22}$$

Now we consider the case when a part of the tasks of set I_q is performed by m_q workers and a part of the tasks of set I_q is performed by $m_q - 1$ workers.

Let $I^-_{(q,k)}$ be first k tasks of station q and $I^+_{(q,k)} = I_q \backslash I^-_{(q,k)}$.

We can calculate:

$$k_q = \min \left\{ k | \sum_{j \in I^-_{q,k}} p_j(m_q) + \sum_{j \in I^+_{q,k}} p_j(m_q - 1) \leq C \right\} \tag{2.23}$$

Note that $I^-_{q,k} \neq \varnothing$ and $I^+_{q,k}$ can be empty set.

If $I^+_{q,k} \neq \varnothing$, then tasks of set I^-_{q,k_q} are performed by m_q workers and tasks of set I^+_{q,k_q} are performed by $m_q - 1$ workers. In this situation, one worker at station q is free starting from the time:

$$S_q = \sum_{j \in I^-_{q,k}} p_j(m_q) \tag{2.24}$$

This worker can perform tasks of set $I_{q+1,r}$ at station $q + 1$. We assume that all tasks assigned to a particular station are denoted sequentially and first task performed at a station has index 1, so:

$$I_{q+1,r} = \{j \in I_{q+1}, j \geq r\} \tag{2.25}$$

If all the workers initially assigned to the station $q + 1$ remain at this station until the end of takt time, we can calculate r using the following formula:

$$r = \min \left\{ r | \sum_{j \in I_{q+1}, j < r} p_j(m_{q+1}) \geq S_q \right\} \tag{2.26}$$

However we should note that this formula does not take into account that, due to the set $I^{+}_{q+1,k}$, one of the workers at station $q+1$ does not necessarily perform all tasks from set $I^{*} = \{j|j \in I_{q+1}, j < r\}$, because of his/her movement to station $q+2$. We will provide the exact value for r after clarifying the partition of all tasks of station $q+1$ into subsets that are performed by different numbers of workers.

Now we need to calculate the minimum number of workers m_{q+1} required to perform a given set of tasks $I_{q+1}I_q$ at station $q+1$, taking into account that a worker from station q performs tasks $I_{q+1,r}$. To make these calculations, we consider the following equalities:

$$I_q = I^{-}_{q,k} \cup I^{+}_{q,k} \tag{2.27}$$

$$I_q = (I_q \backslash I_{q,r}) \cup I_{q,r} \tag{2.28}$$

$$I_q \backslash I_{q,r} = (I^{-}_{q,k} \backslash I_{q,r}) \cup (I^{+}_{q,k} \backslash I_{q,r}) \tag{2.29}$$

$$I_{q,r} = (I_{q,r} \backslash I^{-}_{q,k}) \cup (I_{q,r} \backslash I^{+}_{q,k}) \tag{2.30}$$

Therefore

$$I_q = (I^{-}_{q,k} \backslash I_{q,r}) \cup (I^{+}_{q,k} \backslash I_{q,r}) \cup (I_{q,r} \backslash I^{-}_{q,k}) \cup (I_{q,r} \backslash I^{+}_{q,k}) \tag{2.31}$$

The number of workers for each subset is shown in Table 2.3.

TABLE 2.3: Number of workers for each subset of I_q

$I^{-}_{q,k} \backslash I_{q,r}$	m
$I^{+}_{q,k} \backslash I_{q,r}$	$m-1$
$I_{q,r} \backslash I^{-}_{q,k}$	m
$I_{q,r} \backslash I^{+}_{q,k}$	$m+1$

Figures 2.6 and 2.7 demonstrates the subsets defined above.

Since we have already defined how many workers are performing tasks for each subset (see Table 2.3), we can provide an exact value for r:

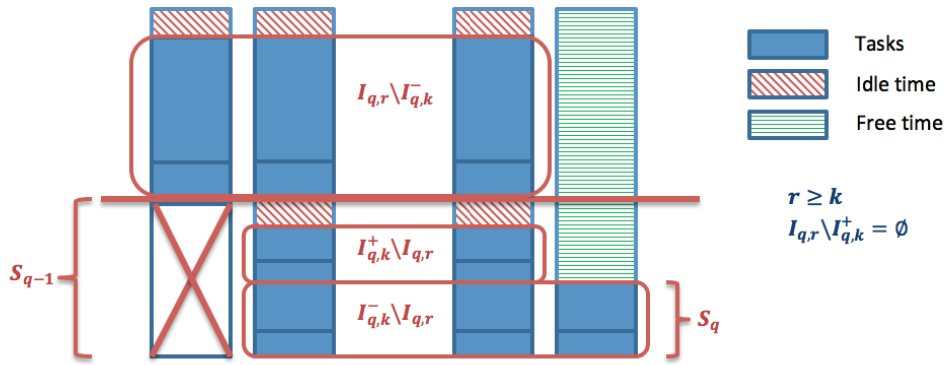


FIGURE 2.6: Graphic representation of I_q subsets in case if $r \geq k$ and $I_{(q,r)} \backslash I^+_{(q,k)} = \varnothing$

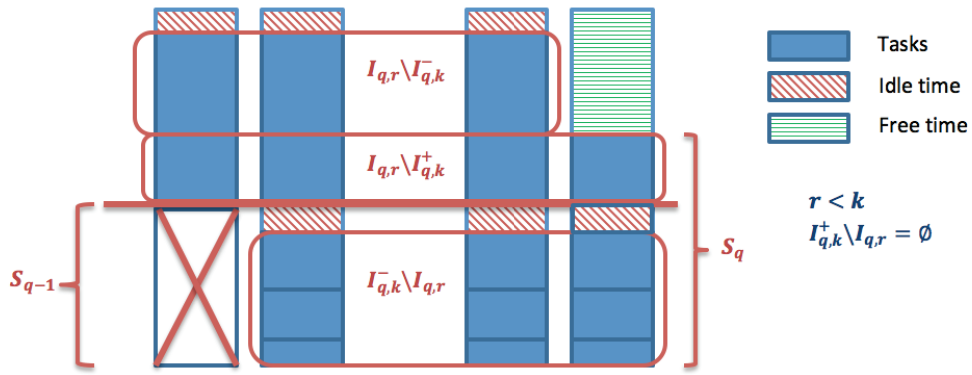


FIGURE 2.7: Graphic representation of I_q subsets in case if $r < k$ and $I^+_{(q,r)} \backslash I_{(q,r)} = \varnothing$

$$r = \min \left\{ r \middle| \sum_{j \in I^-_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1}) + \right.$$

$$\left. + \sum_{j \in I^+_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1} - 1) \geq S_q \right\} \quad (2.32)$$

Note that set $I_{q,r}$ defined by formula (2.32) contains the set $I_{q,r}$ defined by the formula (2.25).

One can think that we can use the following formula to determine m_{q+1}:

$$m_{q+1} = \min\{m|k_{q+1} = \\ = \min\left\{k|\sum_{j\in I^-_{q+1,k}\backslash I_{q+1,r}} p_j(m_{q+1}) + \sum_{j\in I^+_{q+1,k}\backslash I_{q+1,r}} p_j(m_{q+1}-1) \right. \\ \left. + \sum_{j\in I_{q+1,r}\backslash I^-_{q+1,k}} p_j(m_{q+1}) + \sum_{j\in I_{q+1,r}\backslash I^+_{q+1,k}} p_j(m_{q+1}+1) \le C\right\}\right\} \quad (2.33)$$

However, this formula does not guarantee that there will be no overflow for one worker and underflow for another one (see an explanation below). We need to add constraints for each worker and then combine them. As the result we will obtain the following:

$$m_{q+1} = \min\{m|k_{q+1} = \\ = \min\left\{k|\sum_{j\in I^-_{q+1,k}\backslash I_{q+1,r}} p_j(m_{q+1}) + \sum_{j\in I^+_{q+1,k}\backslash I_{q+1,r}} p_j(m_{q+1}-1) \le S_q, \right. \\ \left. \sum_{j\in I_{q+1,r}\backslash I^-_{q+1,k}} p_j(m_{q+1}) + \sum_{j\in I_{q+1,r}\backslash I^+_{q+1,k}} p_j(m_{q+1}+1) \le C - S_q\right\}\right\} \quad (2.34)$$

Thus, one worker at station $q+1$ does not perform tasks of set $(I^+_{q+1,k}\backslash I_{q+1,r}) \cup I_{q+1,r}\backslash I^-_{q+1,k} = I^+_{q+1,k}$. This worker can perform tasks of set I_{q+2r} at station $q+2$ starting from the time S_{q+1}:

$$S_{q+1} = \sum_{j\in I^-_{q+1,k}\backslash I_{q+1,r}} p_j(m_{q+1}) + \sum_{j\in I_{q+1,r}\backslash I^+_{q+1,k}} p_j(m_{q+1}+1) + \Delta_{q+1} \quad (2.35)$$

$$\Delta_{q+1} = \begin{cases} 0 & r \geq k \\ S_q - \sum_{j \in I^-_{q+1,k} \setminus I_{q+1,r}} p_j(m_{q+1}) + \sum_{j \in I^+_{q+1,k} \setminus I_{q+1,r}} p_j(m_{q+1} - 1) & r < k \end{cases}$$

Note that $S_{q+1} \geq \sum_{j \in I^{-1}_{q+1,k}} p_j(m_{q+1})$, due to possible idle time between tasks.

The term Δ_{q+1} appears in equality (2.35) because the time S_q is evaluated for tasks of station q but the sets $I^+_{q+1,k}, I^-_{q+1,k}$ are defined for tasks of station $q+1$. Therefore, the idle time shown in Figure 5 appears between tasks of set $I^-_{q+1,k} \setminus I_{q+1,r}$ and tasks for set $I_{q+1,r} \setminus I^+_{q+1,k}$. For this reason, in order to calculate m_{q+1} we use formula (2.34) instead of formula (2.33).

Note that we can define S_{q+1} in another way:

$$S_{q+1} = \begin{cases} \sum_{j \in I^-_{q+1,k} \setminus I_{q+1,r}} p_j(m_{q+1}) & r \geq k \\ S_q + \sum_{j \in I_{q+1,r} \setminus I^+_{q+1,k}} p_j(m_{q+1} + 1) & r < k \end{cases}$$

2.4.5 Heuristic Min-Idle-One-Traveling-Worker

In the previous heuristic we used sequentially arranged stations. However, we can improve this heuristic by choosing the next station for a worker to move, it will be the station which will have the minimal total idle time after adding this supplementary worker. Note that we should only take into account idle time but not the free time (the rest of the time at the start of moving, i.e. the difference between the end of cycle and the moment when the move starts) of the worker who is going to move to the next station.

2.4.6 Heuristic Random-One-Traveling-Worker

This heuristic is similar to algorithm described in Sections 2.4.4 and 2.4.5. The difference of this methods is that stations are chosen at random.

2.4.7 Heuristic Additional-Workers

This heuristic is as follows (we assume that workers performing tasks can move from one station to another).

Step 1: For every station $q, q = 1, \ldots, s$, we evaluate the number of workers m_q so that

$$\sum_{j \in I_q} p_j(m_q + 1) < C \leq \sum_{j \in I_q} p_j(m_q), \qquad q = 1, \ldots, s \tag{2.36}$$

and the remainder time (slack time):

$$r_q = \sum_{j \in I_q} p_j(m_q) - C \tag{2.37}$$

and we assign m_q workers to each station q.

Step 2: Now, the problem is to find a minimal set of additional workers $\{w_d, 1 \leq d \leq D\}$ to perform the set of tasks I . This problem is equivalent to the problem of placement of intervals with length $r_q, q = 1, \ldots, s$, without intersections in the minimal number of intervals with length C.

For station $q = 1$, let b_{q-1} be equal to 0 and let the index of additional worker d be equal to 1. Find minimal set of tasks $I_{q,r}$ for station q such that they are performed after time b_{q-1} and their runtime is greater than or equal to r_q. In order to do this, we calculate:

$$a_q = \min \left\{ a | \sum_{j \in I_q, j \leq a} p_j(m_q) \geq b_{q-1} \right\}, \tag{2.38}$$

$$b_q = \min \left\{ b | \sum_{j \in I_q, a_q \leq j < b} p_j(m_q + 1) \geq r_q \right\}. \tag{2.39}$$

Note that we can also use the following formula for b_q:

$$b_q = \min \left\{ b | \sum_{j \in I_q, j < a_q} p_j(m_q) + \sum_{j \in I_q, a_q \leq j < b} p_j(m_q + 1) + \sum_{j \in I_q, j \geq b} p_j(m_q) < C \right\}. \tag{2.40}$$

Then, $I_{q,r} = \{j \in I_q, a_q \leq j < b_q\}$. If $I_{q,r}$ exists, we add the set $I_{q,r}$ to tasks that are performed by the worker with number d, then go to next station $q + 1$ and find a minimal set of tasks $I_{q+1,r}$. If the set $I_{q,r}$ does not exist, we take a new additional worker w_{d+1}, assign $b_{q-1} = 0$ and find a minimal set of tasks $I_{q,r}$ for station q.

2.4.8 Heuristic Additional-Workers-Improved

We can also consider the following modification of the heuristic proposed above. For every station $q, q = 1, \ldots, s$, we evaluate

$$b_q = \min \left\{ b | \sum_{j \in I_q, a_q \leq J < b} p_j(m_q + 1) \geq r_q \right\}. \tag{2.41}$$

where a_q does not depend on b_{q-1}; here a_q is selected randomly or so that $b_q - a_q$ is minimized. We get intervals $[a_q, b_1], q = 1, \ldots, s$. Then, we solve the problem of placement of intervals $[a_q, b_q]$ without intersections in the minimal number of intervals $[0, C]$.

2.5 Computer experiments

In order to compare proposed heuristics, they were implemented using C# programming language. Figure 2.8 demonstrates a screenshot of the developed software application.

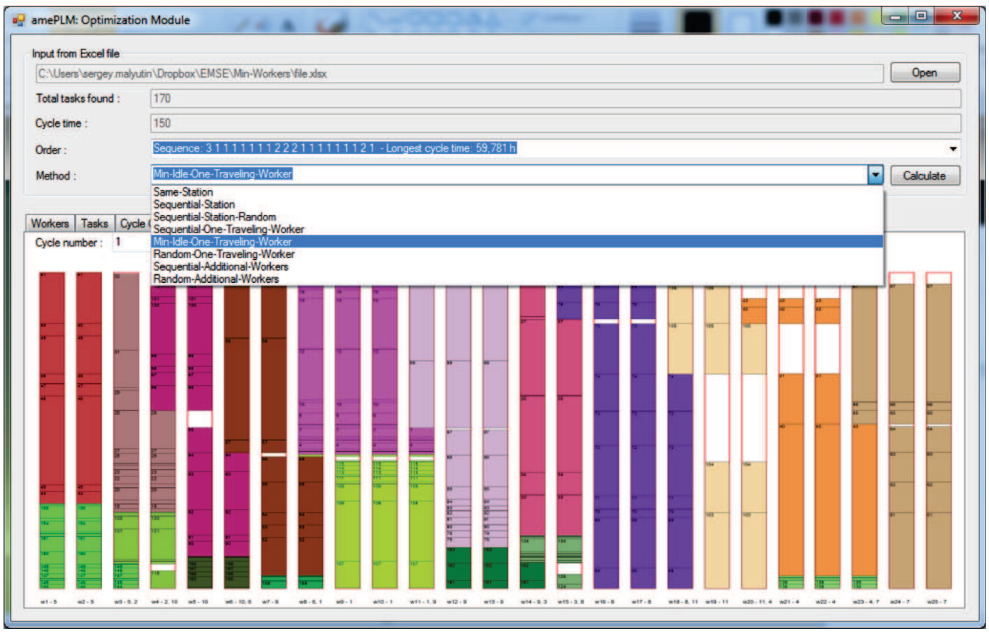


FIGURE 2.8: Screenshot of the created application

The experiments were run on a PC with Intel Core i7-3520M CPU with 4Gb of RAM under MS Windows 7, 64bit. All the heuristics take a few seconds to provide a solution, which is very fast. The time limit for solving

the MILP problem related to each of the 20 cycles was set to 30 minutes. However, we were unable to solve the initial MILP to find the exact solution, because of the exponentially increasing time needed to find a solution.

In order to visualise the performance of the created heuristics in different cycles, we have made a chart (Figure 2.9) which demonstrates number of workers obtained by all heuristics. The vertical axis shows the number of workers needed and the horizontal axis shows the number of a cycle. Due to the independence of cycles, we can select for each cycle the best solution of different heuristics. For example, we can choose the solution obtained by heuristic Random-One-Traveling-Worker for all cycles except 6, 11 and 16. For cycles 6, 11, 16 we choose solution obtained by Random-Additional-Workers heuristic. Detailed results are given in Appendix B.

The workload distribution is unbalanced - some workers have a higher percentage of idle time than others. In order to smooth the workload, we propose to rank workers in cycle q (which is just finished) in the non-increasing order of their accumulated idle times. For the next cycle $q + 1$, the workers are ranked in the non-increasing order of their workload. Then, for the cycle $q + 1$, the worker with the highest workload will be replaced (to perform his/her tasks) by the worker with the highest accumulated idle time in the cycle q. For example, let worker w_1 have the highest accumulated idle time up to and including the current cycle q, and let worker w_2 have the highest workload in the next cycle q+1 according to the obtained solution. Then, in the cycle $q + 1$, worker w_1 will perform the tasks of worker w_2.

2.6 Conclusions

Preliminary results of this chapter were described in Battaïa et al. [8] [9] [10].

Previously the manufacturer used a solution, in which workers do not move between stations. For the industrial case, such an approach finds solutions with 28 workers for 15 cycles, and 29 workers for the remaining 5 cycles. Since the set of workers should not change from one cycle to another, 29 was the number of workers used by the industrial partner. The problem of best product sequence was solved for the real case from the industrial partner. The workforce minimisation problem for a semi-automatic multi-product sequential assembly line and identical workers

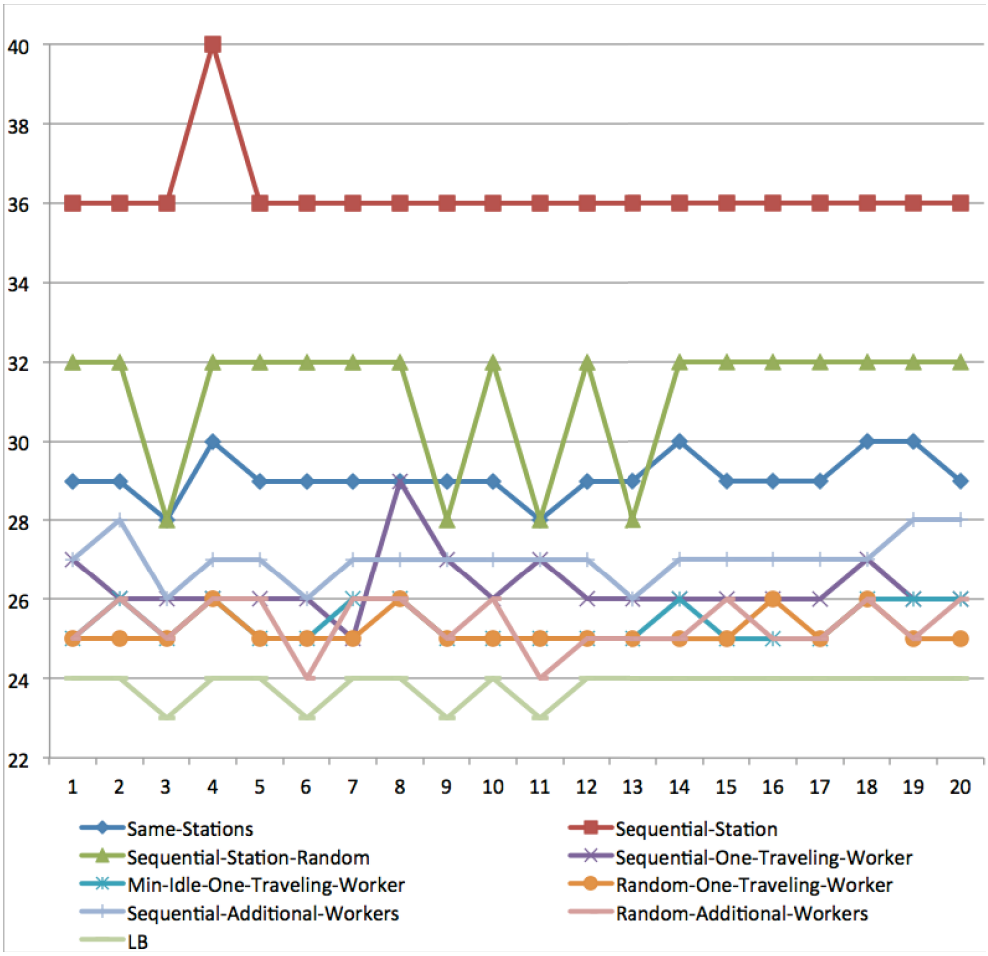


FIGURE 2.9: Comparison of the heuristics' performance (number of workers for each cycle)

was solved. After the results of two solved problems (best product sequence problem and workforce minimisation problem) have been applied, the number of workers to 26. This result was obtained by a combination of all the proposed heuristics.

An exact method provided no solution. Therefore heuristic methods were developed. Some of these heuristics are based on the idea of limiting workers movements. In particular, each worker is allowed to make only one move from station to another and if there are many workers assigned to a station, only one from them can move to another station. As a perspective for the future research, another MILP model can be developed which addresses constraint that restricts the number of worker movements. This approach is a heuristic for the the studied problem.

Chapter 3

Workforce minimisation with an acyclic precedence graph

3.1 Introduction

The problem studied in this chapter generalises the problem studied in chapter 2 so that chain precedence constraints are replaced with arbitrary acyclic precedence constraints and operation times inversely proportional to the number of workers are replaced with operation processing times being arbitrary non-decreasing functions of the numbers of workers. We consider a *paced unidirectional assembly line* consisting of m stations and manufacturing different products. Every station switches from processing a current product to the following one simultaneously and with the same time step. The time interval that spans between two successive switches is called *cycle* and its length is called *cycle time*. Motivated by a real life production situation, we study a workforce assignment problem for one cycle of such a line. Without loss of generality, we assume that the cycle starts at time zero. In a given cycle, workers on station k execute a given set N_k of operations, $k = 1, \ldots, m$. Operations of different stations can be performed in parallel. The order of operations of the same station should follow a given technological process characterized by *precedence relations* between the operations. If operation i is followed by operation j, then i must be completed before the start time of i. If operations i and j have no precedence relation, then they are called *independent* and can be performed in any order. Operations with no predecessor can start at time zero. The set of precedence relations of operations on station k is represented by a *directed acyclic graph* $G_k = (N_k, U_k)$, where N_k is the set of *vertices* (operations) and U_k, $U_k \subset N_k \times N_k$, is the set of *arcs* (oriented pairs of operations) (i, j) such that $(i, j) \in U_k$ if and only if operation i is followed by operation j. Let $N = \cup_{k=1}^{m} N_k$, $n = |N|$, and $U = \cup_{k=1}^{m} U_k$.

Operations are executed by $r_{\max}$ *identical* workers. *Processing time* $p_j(r)$ of an operation j is an integer valued positive non-increasing function of the number of workers r assigned to it. If a worker is assigned to an operation, he or she performs this operation from its start until completion. No worker can perform more than one operation at any time. The time of the worker's movement from one station to another is negligibly small comparing to the time of any operation. Therefore, we assume that any worker can move from one operation to another in zero time. The decision to be made is a *schedule*, in which the operations' start times and workforce assignment are specified. Given a schedule, the number r_j of workers assigned to operation j, the operation *start time* S_j and the operation *completion time* C_j can be calculated for each operation j such that $C_j = S_j + p_j(r_j)$, $j = 1, \ldots, n$. The *makespan* of a schedule is defined as $C_{\max} = \max_{j \in N}\{C_j\}$. This value is equal to the line cycle time.

The following constraints must be satisfied in a *feasible* schedule:

- $a_j \leq r_j \leq b_j$, where a_j and b_j are given positive integer numbers, $j = 1, \ldots, n$, and
- $C_{\max} \leq d$, $j = 1, \ldots, n$, where d is a given upper bound on the line cycle time.

The problem that we denote as $MinNumber$ consists in finding a feasible schedule which minimizes the maximum number of workers employed simultaneously,

$$W_{\max} = \max_{0 \leq t \leq d} \Big\{ \sum_{j \in N(t)} r_j \Big\},$$

where $N(t)$ is the set of operations executed at time t. Let $W^*_{\max}$ denote the minimum $W_{\max}$ value. Assume without loss of generality that the number of available workers is such that $r_{\max} \leq \sum_{j \in N} b_j$, because otherwise we can re-set $r_{\max} = \sum_{j \in N} b_j$, and that $\sum_{j \in N_k} p_j(b_j) \leq d$ for $k = 1, \ldots, m$ and $\max\Big\{\max_{j \in N}\{a_j\}, \lceil \sum_{j \in N} p_j(b_j)/d \rceil\Big\} \leq r_{\max}$, because otherwise the problem $MinNumber$ has no solution.

For the sake of clarity, consider an example in which the assembly line consists of two stations, eight operations and four available workers. Precedence relations are given by the graph in Figure 3.1.

Processing times of operations depending on the number of workers are given in Table 3.1.

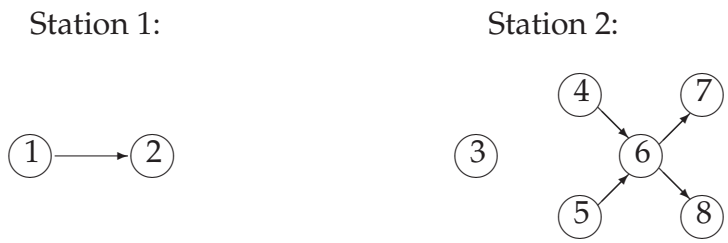


FIGURE 3.1: Precedence graph

Operations\ Number of workers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	50	10	10	9	11	24	20	
2	30		6	5	7	12	12	9
3						8	7	6

TABLE 3.1: Processing times of operations

An empty entry for a given number of workers and operation j means that this number of workers is either less than a_j or greater than b_j. Note that all four workers can be used on the line, but only one, two or three of them can be used to perform the same operation. A Gantt chart illustrating a feasible schedule with an upper bound on line cycle time $d = 44$ and maximum number of workers $r_{\max} = 4$ is given in Figure 3.2.

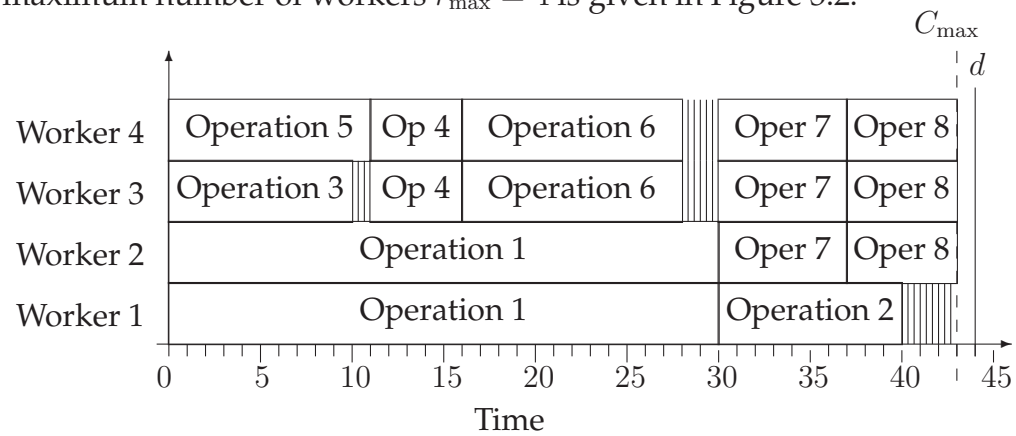


FIGURE 3.2: A feasible schedule. Dashed rectangles represent idle times of workers.

3.2 Heuristics

We suggest three parametrized constructive heuristics for the problem $MinNumber$: two conventional ones and one randomized. All heuristics use a numerical parameter α, $0 \leq \alpha \leq 1$, which affects the heuristics behavior in such a way that a higher value of α tends to assign workers with a higher *ready time* value to an operation. In the beginning, all three heuristics determine a hypothetical minimal total number workers, W, to be employed, set initial numbers of workers for each operation: $r_j = a_j$, $j \in N$, and build a schedule by assigning operations to W workers over time in an order which is topologically feasible with respect to the precedence graph G. The topological feasibility means that in the constructed schedule precedence constraints are not violated. Since there can exist numerous topologically feasible orders, we have to determine the operations selection procedure. Thus, from the set of operations having no predecessor, the conventional heuristic $TopLong(\alpha)$ selects the longest operation first, the conventional heuristic $TopLongPath(\alpha)$ selects an operation of the longest path first and the randomized heuristic $TopRandom(\alpha)$ selects an operation to be assigned first at random.

Conventional heuristics $TopLong(\alpha)$ and $TopLongPath(\alpha)$

Step 1 Determine an initial hypothetical minimal total number of workers, W. Obviously, $\max_{j \in N}\{a_j\} \leq W \leq r_{\max}$. The specific value of W can be defined by an expert. Alternatively, we propose to set $W = \max\left\{\max_{j \in N}\{a_j\}, \lceil \sum_{j \in N} p_j(b_j)/d \rceil\right\}$.

Set $r_j = a_j$ and calculate $p_j(r_j)$ for $j = 1, \ldots, n$. Initiate *ready times* T_i *of workers*: $T_i = 0$, $i = 1, \ldots, W$, and *ready times* t_j *of operations*: $t_j := 0$, $j = 1, \ldots, n$.

Step 2 In graph G, identify the set N^+ of operations having no predecessor. In heuristic $TopLongPath(\alpha)$, calculate the *longest path* P^* connecting a vertex of N^+ with any other vertex of the graph G. Path length is the total *weight* of its vertices, and the weight of vertex j is $p_j(r_j)$.

Step 3 Select $j^* \in N^+$ with the largest value $p_j(r_j)$ in heuristic $TopLong(\alpha)$ and select j^* as the first vertex of the longest path P^* in heuristic $TopLongPath(\alpha)$.

Let the workers be ordered such that $T_{i_1} \leq \cdots \leq T_{i_W}$. Determine sets of r_{j^*} workers $X_h := \{i_{h-r_{j^*}+1}, i_{h-r_{j^*}+2}, \ldots, i_h\}$, $h = r_{j^*}, r_{j^*}+1, \ldots, k$,

where k satisfies $T_{i_k} \leq \alpha T_{i_W} + (1-\alpha)T_{i_{r_{j^*}}}$ and $T_{i_{k+1}} > \alpha T_{i_W} + (1-\alpha)T_{i_{r_{j^*}}}$, $T_{i_{W+1}} := +\infty$. For example, $X_k = \{i_1, \ldots, i_{r_{j^*}}\}$ if $\alpha = 0$ and $X_k = \{i_{W-r_{j^*}+1}, \ldots, i_W\}$ if $\alpha = 1$. Select index h which minimizes $\max\{t_{j^*}, T_{i_h}\} - T_{h-r_{j^*}+1}$ for $r_{j^*} \leq h \leq k$. Let it be index h^*.

Assign workers of the set X_{h^*} to operation j^* so that all of them start performing this operation at the same time $t^* := \max\{t_{j^*}, T_{i_{h^*}}\}$. Note that index h^* is selected such that the maximum *idle time* of workers in the sets X_h, $h = r_{j^*}, r_{j^*}+1, \ldots, k$, just before they start processing operation j^*, is minimized.

Update ready times of workers so that $T_h := t^* + p_{j^*}(r_{j^*})$, $h \in X_{h^*}$. Update ready times of *immediate successors* of vertex j^* so that $t_j := \max\{t_j, t^*\} + p_{j^*}(r_{j^*})$, $j \in A(j^*)$, where $A(j^*)$ is the set of immediate successors of j^* in the graph G. Update graph G by removing vertex j^* from it.

If G is empty, then a complete schedule is constructed and the following computations are performed.

- Calculate the makespan $C_{\max} = \max\{T_i \mid i = 1, \ldots, W\}$.
- If $C_{\max} \leq d$, then return the corresponding schedule with W workers and stop.
- Suppose that $C_{\max} > d$. If $W = r_{\max}$, then return the infeasible solution with $r_{\max}$ workers and stop. If $W \leq r_{\max} - 1$, then determine operations of a *critical path*, which do not intersect in time and whose processing times sum up to $C_{\max}$. Re-set $r_j := \min\{r_j+1, b_j, W+1\}$ for every such operation, re-set $W := W+1$, restore original graph G, and go to Step 2.

If G is not empty, then perform Step 2.

Heuristic $TopLong(\alpha)$ or $TopLongPath(\alpha)$ is run until a feasible solution is constructed, or an infeasible solution with $r_{\max}$ workers is obtained, or a given solution time limit is exceeded, in which case its output is an infeasible solution found last. If the solution time limit permits, then heuristics $TopLong(\alpha)$ and $TopLongPath(\alpha)$ can be run for several values of α, $0 \leq \alpha \leq 1$.

Our **randomized heuristic** $TopRandom(\alpha)$ differs from heuristics $TopLong(\alpha)$ and $TopLongPath(\alpha)$ in that in Step 3 operation $j^* \in N^+$ is selected at random. If the solution time limit permits, then heuristic

$TopRandom(\alpha)$ can be run several times and for several values of α. Computational experiments with the heuristics are described in Section 3.5.

3.3 Reduction to a series of feasibility problems

Consider a problem, which is to determine if there exists a feasible solution of the problem $MinNumber$ with a given number Q of workers. We denote this problem as $Feasible(Q)$.

Problem $MinNumber$ can be solved by the following *bisection search* procedure over the range of possible values of Q. Let LB and UB be lower bound and upper bound, respectively, on the value of Q for which there is a feasible solution of the problem $MinNumber$.

Step 1 Apply heuristics $TopLong(\alpha)$, $TopLongPath(\alpha)$ and $TopRandom(\alpha)$ for certain values of α, $0 \leq \alpha \leq 1$.

- If the heuristics found no feasible solution, then solve the problem $Feasible(r_{\max})$. If no feasible solution of this problem is found, then the problem $MinNumber$ has no feasible solution, stop. If a feasible solution is found, then set $UB = r_{\max}$.
- If the heuristics found a feasible solution, then set UB to be the minimum feasible solution value.
- Set $LB = \max\left\{\max_{j\in N}\{a_j\}, \lceil \sum_{j\in N} p_j(b_j)/d \rceil\right\}$ and solve the problem $Feasible(LB)$. If a feasible solution of this problem is found, then it is an optimal solution of the problem $MinNumber$, stop. If no feasible solution is found, then the *generic iteration* of the bisection search is applied. It can be described as follows.

Step 2 (Generic iteration of bisection search)

- If $UB = LB + 1$, then a feasible schedule for the problem $Feasible(UB)$ is an optimal schedule for the problem $MinNumber$.
- If $UB \geq LB + 2$, then solve the problem $Feasible(Q)$ for $Q = \lceil \frac{UB+LB}{2} \rceil$. If no feasible solution is found, then re-set $LB := Q$ and repeat the generic iteration. If a feasible solution is found, then re-set $UB := Q$ and repeat the generic iteration.

The number of iterations of the bisection search is $O(\log_2(UB - LB))$, and in each iteration the problem $Feasible(Q)$ is solved for a given Q, $Q \in \{LB, LB+1, \ldots, UB\}$.

We now describe a MILP model for the problem $Feasible(Q)$. It is convenient to introduce the following notation:

- N^-: subset of vertices from N, which do not have successors;
- I: set of vertex pairs (i, j) that are independent with respect to the precedence constraints, $I = \{(i, j) \mid i \in N, j \in N, i \neq j, (i, j) \notin U, (j, i) \notin U\}$;
- $I_k = \{(i, j) \mid (i, j) \in I, i \in N_k,\ j \in N_k\}$, $k = 1, \ldots, m$;
- $I_{gk} = \{(i, j) \mid (i, j) \in I, i \in N_g,\ j \in N_k\}$, $g = 1, \ldots, m$, $k = 1, \ldots, m$, $g \neq k$.

Let us introduce the following variables:

- $x_{ir} \in \{0, 1\}$, $i \in N$, $r = 1, \ldots, Q$: $x_{ir} = 1$ if operation i is assigned to worker r, and $x_{ir} = 0$, otherwise;
- $y_{ij} \in \{0, 1\}$, $(i, j) \in I$: $y_{ij} = 1$ if operation i completes before or at the start of operation j, and $y_{ij} = 0$, otherwise. Note that $y_{ij} = 0$ implies that either j completes before or at the start of i, or some parts of i and j are performed in parallel. In the latter case, i and j must be performed by different workers in a feasible solution;
- $z_{ir} \in \{0, 1\}$, $i \in N$, $r = 1, \ldots, Q$: $z_{ir} = 1$ if operation i is performed by r workers, and $z_{ir} = 0$, otherwise;
- $S_{ir} \geq 0$, $i \in N$, $r = 1, \ldots, Q$: start time of operation i if it is processed by worker r, and any non-negative value otherwise.

Below we give a Mixed Integer Linear Programming (MILP) formulation of the problem $Feasible(Q)$, for which we keep the same notation $Feasible(Q)$.

Problem $Feasible(Q)$:

$$S_{ir} + \sum_{q=1}^{Q} p_i(q) z_{iq} \leq d, \; i \in N^-, \; r = 1, \ldots, Q, \tag{3.1}$$

$$\sum_{i=1}^{n} \sum_{q=1}^{Q} p_i(q) z_{iq} \leq Qd, \tag{3.2}$$

$$\sum_{r=1}^{Q} z_{ir} = 1, \;\; i \in N, \tag{3.3}$$

$$\sum_{r=1}^{Q} x_{ir} = \sum_{r=1}^{Q} r z_{ir}, \;\; i \in N, \tag{3.4}$$

$$S_{jh} - S_{iq} \geq \sum_{r=1}^{Q} p_i(r) z_{ir}, \; (i,j) \in U, \; h, q = 1, \ldots, Q, \tag{3.5}$$

$$S_{jh} - S_{iq} \geq \sum_{r=1}^{Q} p_i(r) z_{ir} - (d+1)(3 - y_{ij} - x_{ih} - x_{jq}), \; (i,j) \in I_k, \tag{3.6}$$
$$k = 1, \ldots, m, \; h, q = 1, \ldots, Q,$$

$$S_{jh} - S_{ih} \geq \sum_{r=1}^{Q} p_i(r) z_{ir} - (d+1)(3 - y_{ij} - x_{ih} - x_{jh}), \; (i,j) \in I_{gk}, \tag{3.7}$$
$$g, k = 1, \ldots, m, \; g \neq k, \; h = 1, \ldots, Q,$$

$$S_{ih} - S_{iq} \leq (d+1)(2 - x_{ih} - x_{iq}), \;\; i \in N, \; h = 1, \ldots, Q, \tag{3.8}$$
$$q = h+1, \ldots, Q,$$

$$S_{iq} - S_{ih} \leq (d+1)(2 - x_{ih} - x_{iq}), \;\; i \in N, \; h = 1, \ldots, Q, \tag{3.9}$$
$$q = h+1, \ldots, Q,$$

$$y_{ij} + y_{ji} \leq 1, \;\; (i,j) \in I, \tag{3.10}$$

$$x_{ir} + x_{jr} - 1 \leq y_{ij} + y_{ji}, \; (i,j) \in I, \; r = 1, \ldots, Q, \tag{3.11}$$

$$\sum_{r=1}^{Q} r z_{ir} \leq b_i, \;\; i \in N, \tag{3.12}$$

$$a_i \leq \sum_{r=1}^{Q} r z_{ir}, \;\; i \in N, \tag{3.13}$$

$$x_{ir}, y_{ij}, z_{ir} \in \{0, 1\}, \;\; i, j \in N, \; r = 1, \ldots, Q, \tag{3.14}$$

$$S_{ir} \geq 0, \;\; i \in N, \; r = 1, \ldots, Q. \tag{3.15}$$

The constraints (3.1) address the cycle time d on each station by considering completion times of operations that have no successors. The constraints (3.2) guarantee that all the operation time slots fit the rectangle

with dimensions Q and d. The constraints (3.3) state that exactly one number of workers in the range $1, \ldots, Q$ must be employed for the execution of every operation. The constraints (3.4) ensure that if operation i is assigned some number of workers, say v, and, hence, $z_{iv} = 1$, then exactly v variables x_{ir} take value 1. The constraints (3.5) require that the starting times of operations i and j performed by any workers are $p_i(r)$ time units away from each other if i precedes j. The constraints (3.6) enforce worker q to complete operation i before or at the time when worker h starts operation j if i and j are on the same station (from the same set I_k), it is decided that operation i completes before operation j starts, worker q performs operation i and worker h performs j, i.e., if $y_{ij} = 1$, $x_{iq} = 1$ and $x_{jh} = 1$. The constraints (3.7) do the same as (3.6), but in the case when operations i and j are on different stations and they are performed by the same worker. The constraints (3.8) and (3.9) force all workers assigned to the same operation to start at the same time. The constraints (3.10) ensure that, for independent operations i and j, the case when both $y_{ij} = 1$ and $y_{ji} = 1$ cannot happen. The constraints (3.11) guarantee that if two independent operations are assigned to the same worker, then one of these operations must be completed before or at the start time of the other, i.e., either $y_{ij} = 1$ or $y_{ji} = 1$ must hold. Assume that operations i and j are independent. Consider values of the pair (x_{ir}, x_{jr}). If $(x_{ir}, x_{jr}) = (1, 1)$, i.e., worker r is assigned to i and j, then (3.10) and (3.11) guarantee that either $y_{ij} = 1$ or $y_{ji} = 1$ must take place, which, together with (3.5) for $h = q = r$, guarantees that worker r will not process parts of i and j simultaneously. If $(x_{ir}, x_{jr}) = (1, 0)$, i.e., worker r is assigned to i and not to j, then $y_{ij} = 0$ and $y_{ji} = 0$ can happen, that is, processing of i and j can be done independently in time. Implication of the cases $(x_{ir}, x_{jr}) = (0, 1)$ and $(x_{ir}, x_{jr}) = (0, 0)$ is the same as for $(x_{ir}, x_{jr}) = (1, 0)$. The constraints (3.12) and (3.13) address the limits on the number of workers for each operation.

Note that the solution of the MILP problem $Feasible(Q)$ specifies the start times of the operations, worforce assignment and the operation durations. This information is sufficient to construct the corresponding schedule. The MILP model is verified in the computational study in Section 3.5.

3.4 Relation to multi-mode project scheduling and multiprocessor scheduling. Computational complexity

The problem $Feasible(Q)$ can be viewed as a multi-mode project scheduling problem, in which an activity (operation) is assigned a mode (set of workers) and has the mode dependent duration, see Kolisch et al. [62], Tseng and Chen [86] and Artigues et al. [6] for the definitions of the latter problem. The known mathematical programming formulations of such problems include variables with indices whose number is equal to the number of modes, see Kolisch and Sprecher [61]. Since solution of the problem $Feasible(Q)$ should specify not only the number of workers to perform an operation, but it should also identify these workers, there can be $2^Q - 1$ different modes and the same number of variables in the modeling approach based on modes. The most popular solution techniques for multi-mode project scheduling problems include time-indexed and event-indexed MILP formulations, branch and bound schemes and meta-heuristics, see, for example, Monma et al. [72], Demeulemeester et al. [33], Salewski et al. [79], Ranjbar and Kianfar [76], Li and Womer [68], Besikci et al. [17], and Ghoddousi et al. [54].

Problem $Feasible(Q)$ is related to the multiprocessor *moldable* task scheduling problem, in which the processing time of a computer task depends on the number of identical processors allocated to it, and this number cannot change during the task execution. The processors play the role of the workers. Box constraints on the number of processors allocated to the same task are not considered. According to the survey of multiprocessor task scheduling problems of Drozdowski [42], the makespan minimization counterpart of the problem $Feasible(Q)$, in which a single operation is assigned to each station, is denoted as $P|spdp - lin - \delta_j|C_{\max}$ if the task processing times are inversely proportional to the number of assigned processors, $p_j(r) = p_j/r$, and as $P|spdp - any|C_{\max}$ if processing times are arbitrary functions of the number of processors. For multiprocessor task scheduling problems, off-line and on-line heuristics with performance guarantees are often developed, see, for example, Wang and Cheng [92, 93], Choundhary et al. [28], Srinivasa Prasanna and Musicus [84], Blazewicz et al. [15], Blazewicz et al. [13], Dutot et al. [43] and Hunold [60].

The relation with multiprocessor task scheduling allows establishing computational complexity of the following special cases of the problem $Feasible(Q)$, in which a single operation is assigned to each station: 1) $p_j(r) = 1$ for any r, $a_j = b_j$, $j \in N$; 2) $p_j(r) = 1$ for any r, $a_j = b_j$, $b_j \in \{1, \ldots, \Delta\}$, $j \in N$, Δ is a given constant; 3) $Q = 5$, $a_j = b_j$, $j \in N$; 4) $Q \in \{2, 3\}$, $a_j = b_j$, $j \in N$; 5) $a_j = 0$, $b_j = Q$, $p_j(r) = p_j/g_j(r)$, $g_j(r)$ is a convex increasing function; 6) $a_j = b_j = 1$, $j \in N$. Case 1) is strongly NP-hard, because, in the notation of Drozdowski [42], problem $P|size_j, p_j = 1|C_{\max}$ is strongly NP-hard due to Lloyd [69]. Case 2) is solvable in $O(n)$ time, because problem $P|size_j \in \{1, \ldots, \Delta\}, p_j = 1|C_{\max}$ is solvable in $O(n)$ time due to Blazewicz et al. [14]. Case 3) is strongly NP-hard and case 4) is pseudo-polynomially solvable, because problem $P5|size_j|C_{\max}$ is strongly NP-hard and problem $Pm|size_j|C_{\max}$, $m \in \{2, 3\}$, is pseudo-polynomially solvable due to Du and Leung [44]. Case 5) is solvable in $O(n)$ time, because it reduces to the malleable task scheduling problem studied by Blazewicz et al. [12] and Barketau et al. [7]. In an optimal solution of the latter problem, each task is assigned Q processors. Case 6) is strongly NP-hard, because it is equivalent to the decision version of the classic scheduling problem $P||C_{\max}$, which is NP-hard in the strong sense due to Garey and Johnson [53].

We now prove that the problem $MinNumber$ is NP-hard in the strong sense if a single operation is assigned to each station, values a_j and b_j differ by one unit, and operation processing times are inversely proportional to the number of assigned workers, as they are in the industrial case of the project amePLM [1], which motivates our studies.

Theorem 1 *Problem $MinNumber$ is NP-hard in the strong sense if a single operation is assigned to each station, $a_j = b_j - 1$ and $p_j(r) = p_j/r$, $j \in N$.*

We will use a reduction from the NP-complete problem 3-PARTITION, see Garey and Johnson [53].

3-PARTITION: Given $3v + 1$ positive integer numbers $h_1, \ldots, h_{3v}$ and H satisfying $\sum_{j=1}^{3v} h_j = vH$, does there exist a partition of the set $\{1, \ldots, 3v\}$ into subsets $Y_1, \ldots, Y_v$ such that $\sum_{j \in Y_t} h_j = H$ for $t = 1, \ldots, v$? Assume without loss of generality that $h_j \geq v + 1$, $j = 1, \ldots, 3v$. If it is not the case, all numbers $h_1, \ldots, h_{3v}$ and H can be multiplied by $v + 1$ with no change of the problem complexity.

Given an instance of 3-PARTITION, we construct an instance of the problem $MinNumber$, in which there are $n = 3v$ operations and the same

number of stations, operation j is assigned to station j, $p_j(r) = h_j/r$, $a_j = h_j - 1$, $b_j = h_j$, $j = 1, \ldots, n$, and $d = v$. Thus, the processing time of operation j can take one of the two values: 1 or $1 + \frac{1}{h_j - 1}$, where $1 \leq 1 + \frac{1}{h_j-1} \leq 1 + \frac{1}{v}$, $j = 1, \ldots, n$. Note that any operations can be performed in parallel if they are assigned to different workers. We will show that there exists a feasible solution for this instance with value $W^*_{\max} \leq H$ if and only if the original instance of 3-PARTITION has a solution.

Assume that there exists a feasible solution of the problem $MinNumber$ with r_j workers assigned to operation j, $j = 1, \ldots, n$, and the maximum number of used workers $W^*_{\max} \leq H$. For this solution, let us represent an assignment of a worker i to operation j as a *small rectangle* of height 1 and length $p_j(r_j)$ located in line i of a *large rectangle* of height H and length v. Then the solution can be viewed as a collection of small rectangles inscribed into the large rectangle so that no two small rectangles have a common point other than on their border. Figure 3.2 can be used for an illustration. Observe that every operation j contributes h_j to the total area of the large rectangle irrespectively of the assignment of workers, because $rp_j(r) = h_j$ for any r. Since $\sum_{j=1}^{n} h_j = vH$, the union of the small non-intersecting rectangles must give the large rectangle.

We will now show that every operation j is assigned h_j workers. Assume the contrary that at least one operation q is assigned $h_q - 1$ workers. Consider one of these workers. Let it be used to fulfill w operations, and let J be a subset of these operations, each operation j of which is assigned $h_j - 1$ workers. We have $w \leq v - 1$, because if $w = v$, then, since the processing time of every operation is at least one time unit and the processing time of at least one operation is greater than one time unit, the total occupation time of this worker is greater than v, that is, the cycle time constraint is violated. The total occupation time of this worker is equal to

$$w + \sum_{j \in J} \frac{1}{h_j - 1} \leq v - 1 + \frac{|J|}{v} \leq v - 1 + \frac{w}{v} \leq v - 1 + \frac{v-1}{v} < v.$$

This strict inequality implies that there is a space in the large rectangle not occupied by any small rectangle, which is a contradiction.

We have shown that every operation j must be assigned h_j workers. Hence, the processing time of any operation is equal to 1. Denote the set of operations performed in parallel in the time interval $[t-1, t]$ as Y_t. Since the union of small rectangles gives the large rectangle, equality $\sum_{j \in Y_t} h_j = H$ holds for $t = 1, \ldots, v$, as required for the proof of the part "only if".

Part "if" is proved easily: if $Y_1, \ldots, Y_v$ is a solution of the problem 3-PARTITION, then assign h_j workers to operation j, $j = 1, \ldots, n$, and perform operations of the set Y_t in parallel in the time interval $[t-1, t]$, $t = 1, \ldots, v$.

Computational complexity results are summarized in Table 3.2.

TABLE 3.2: Complexity of special cases of $MinNumber$, in which a single operation is assigned to each station

Problem characteristics	Complexity
$p_j(r) = 1$, $a_j = b_j$	strongly NP-hard
$p_j(r) = 1$, $a_j = b_j$, $b_j \in \{1, \ldots, \Delta\}$, Δ is a given constant	$O(n \log r_{\max})$
$r_{\max} = 5$, $a_j = b_j$	strongly NP-hard
$r_{\max} \in \{2, 3\}$, $a_j = b_j$	pseudo-polynomially solvable
$p_j(r) = p_j/g_j(r)$, $g_j(r)$ is convex increasing, $a_j = 0$, $b_j = r_{\max}$	$O(n \log r_{\max})$
$a_j = b_j = 1$	strongly NP-hard
$p_j(r) = h_j/r$, $a_j = h_j - 1$, $b_j = h_j$	strongly NP-hard

3.5 Computational study

We performed two series of computational experiments with the problem $MinNumber$. The goal of the first series is to establish the maximum problem size in terms of the number of operations that can be solved to optimality in a reasonable time based on the bisection search and the MILP formulation. The goal of the second series is to compare the quality of the heuristics. We used the data generator for the simple assembly line balancing problem described in Otto et al. [75], extending it to match the specificity of the problem $MinNumber$. This generator constructs instances with the number of operations $n = 20$ and $n = 50$. We do the same. The number of stations, the assignment of operations to the stations and the box constraints are selected to conform with the real life problem of our industrial partner, see Chapter 2. Processing times $p_j(1) = p_j$, where p_j are generated as in [75], and $p_j(r) = p_j/r$ for $r \geq 2$. We assumed that each connected component of a precedence graph, generated as in [75], corresponds to a station. Values $a_j \in \{1, 2\}$ were generated with probability 2/3 for $a_j = 1$ and probability 1/3 for $a_j = 2$, and values $b_j = 4$

for all j, as in the industrial application. The source code of the computer implementations is available on request at GitHub [2].

3.5.1 Exact solution: maximum number of operations

The MILP model $Feasible(Q)$ was handled by the solver IBM ILOG CPLEX Optimization Studio 12.6.2, which was run on a workstation Intel Xeon CPU E5-2673 v3 @ 2.40 GHz 8 GB RAM with MS Windows Server 2008 R2 Datacenter, 64bit on 2 different hardware configurations, with 4 threads and 16 threads. The time limit for solving each instance was set to 1 hour.

For $n = 50$, no instance of $MinNumber$ was solved to optimality within one hour in either configuration. For $n = 20$, 250 instances were generated. Since functions $p_j(r) = p_j/r$ are convex, the relaxed problem with $a_j = 1$, $b_j = W^*_{\max}$, no precedence constraints and the possibility to perform any operations in parallel if they are assigned to different workers, is the special case 5) in Section 3.4, which is solved by assigning all $W^*_{\max}$ workers to each operation. Therefore, $W^*_{\max} \geq LB = \lceil \sum_{i \in N} p_i/d \rceil$ in this case, and we used the latter lower bound LB in the experiments. We also calculated the upper bound UB on the number of workers as the sum of the minimal numbers of workers required for each station independently: $UB = \sum_{k=1}^{m} \max \left\{ \max_{j \in N_k}\{a_j\}, \lceil \sum_{j \in N_k} p_j/d \rceil \right\}$.

Recall that the bisection search procedure includes solving the problem $Feasible(Q)$ for $Q = LB$, where LB depends on the problem instance as it is written in the previous paragraph. Fig. 3.3 shows the numbers of solved and unsolved instances of the problem $Feasible(LB)$ for $n = 20$. The horizontal axis represents the values of LB that were calculated for all the generated instances. The height of the color column corresponding to a given LB represents the number of instances with this LB. One can see that the number of solved instances decreases as the number of workers increases.

We observed that the solver finds a feasible solution of the problem $Feasible(Q)$ quite fast if it exists, but it takes a lot of time to detect infeasibility. This observation can be used to solve the problem $Feasible(Q)$ heuristically: if a feasible solution is not found within a certain time limit, then it is decided that the problem $Feasible(Q)$ has no solution. This observation is confirmed by the fact that the ratio of the number of solved

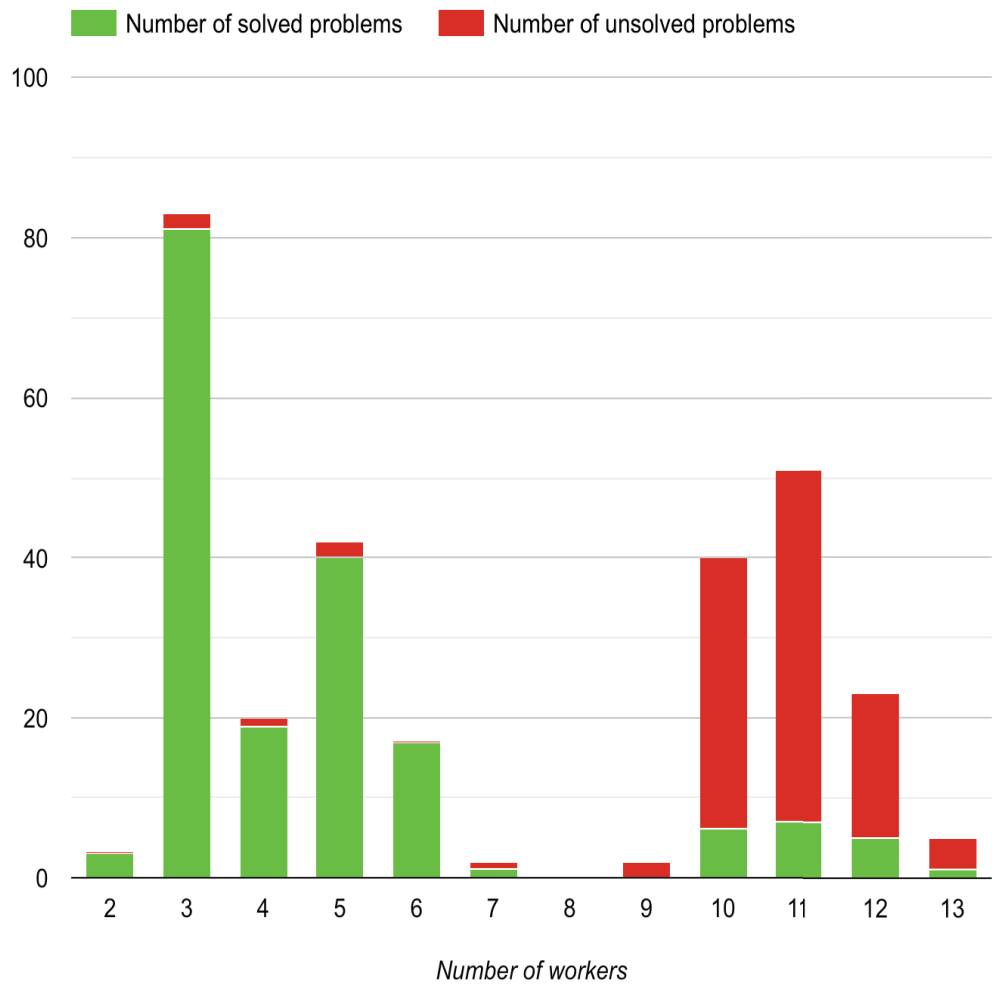


FIGURE 3.3: Numbers of solved and unsolved instances of $Feasible(LB)$ in 1 hour for $n = 20$

instances to the number of unsolved instances increases as the value of Q increases from LB to UB.

We also noticed that the idle time parameter calculated as $Idle = \frac{Qd - \sum j \in N p_j}{Qd}$ impacts the performance, see Fig. 3.4) for an example. One can see that higher value of $Idle$ implies faster solution of the problem $Feasible(3)$.

We compared the impact of the number of available processor threads

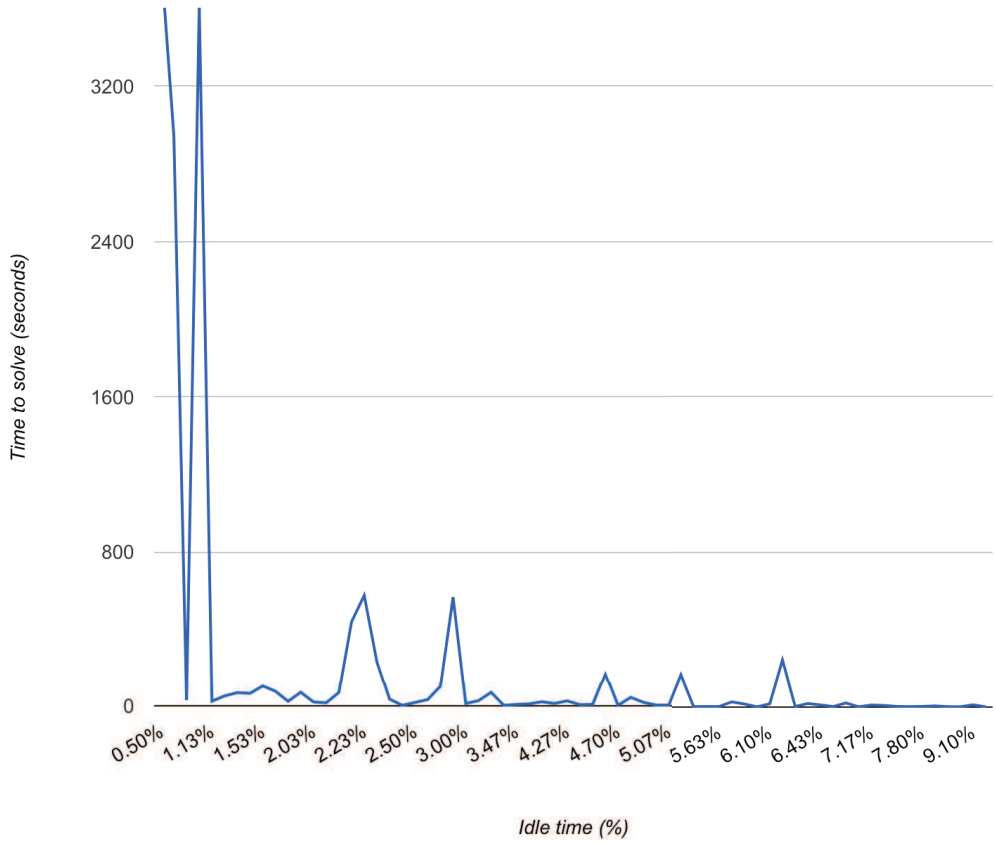


FIGURE 3.4: Impact of parameter $Idle$ on solution time of the problem $Feasible(3)$

on the first 100 generated instances. Table 3.3 shows that using 16 processor threads instead of 4 leads to solving 7% more instances within the 1h time limit.

Configuration	4 threads	16 threads
Percent of solved instances	67%	75% (+7%)
Average solving time of solved instances	225 s	271 s (+46 s)

TABLE 3.3: Scalability of the MILP model

However, the average instance solution time for instances solved to optimality slightly increases for the more powerful configuration due to the fact that the instances unsolved for 4 threads also need long solution

time for 16 threads. A general observation is that realization of the MILP model on parallel threads does not improve the performance a lot.

We have also tried to solve the real-life problem from the project amePLM [1] with 170 operations. In this problem, there are 20 production cycles, which differ by the assignment of the same set of operations to the stations. The problem is to minimize the maximum number of workers needed for each cycle. For this problem, the straightforward solution of the MILP model was not possible because of the memory limitation, which was mainly caused by the constraints (3.6) and (3.7). We noted that the matrix of these constraints contains many zero entries and decided to modify the model such that it is populated by non-zero entries only. This modification removed the memory problem, but the required CPU time increased dramatically such that 24 hours were not enough to populate the model. In order to find a compromise between the time and memory requirements, we have tried to balance the model population with with non-zero row entries only. However, we failed in all our attempts.

Based on the advice of a referee, we have applied a heuristic approach which is to aggregate operations on the same station, whose intervals $[a_j, b_j]$ intersect, into one operation, say J, with the interval $[a_J, b_J]$ being the intersection of the original intervals, and solve the problem with the aggregated operations. By doing this, we reduced the number of operations from 170 to 28. A feasible solution to the problem $MinNumber$ with 25 workers was found in 1 day by solving the problem $MinNumber$ for each of the 20 production cycles. In order to better demonstrate the performance, we performed experiments with the problems $Feasible(Q)$ for all values of Q from the interval $[LB, UB]$, see Fig. 3.5. Note that, though $LB = 23$ for several production cycles, the entire line can not operate with less than 24 workers. Therefore, we did not solve solve $Feasible(Q)$ for $Q = 23$. We also did not solve the problem $Feasible(Q)$ for $Q > UB$. The corresponding entries are blank.

Solution with 25 workers for the problem with the 28 aggregated operations was converted into a feasible solution for the original real-life problem with 170 operations by addressing the precedence constraints. This result is better than the previously obtained result of 26 workers obtained in Chapter 2 for the same problem.

Production cycle	Number of workers								
	LB	UB	24	25	26	27	28	29	30
0	24	29	3600	129	40	50	41	52	
1	23	28	626	55	39	44	49		
2	24	29	3600	280	36	30	75	41	
3	24	29	908	49	38	28	54	43	
4	23	28	516	130	40	151	43		
5	24	30	3600	605	49	32	50	23	56
6	24	29	3600	540	50	40	33	62	
7	24	29	1195	322	59	43	22	49	
8	24	28	600	298	42	42	41		
9	23	28	38	36	44	44	26		
10	24	29	628	180	42	43	45	44	
11	24	30	3600	186	70	133	53	41	56
12	24	29	1965	38	188	37	126	59	
13	24	30	2299	42	73	42	48	51	55
14	24	29	3600	693	51	55	51	35	
15	24	29	3600	41	50	79	34	48	
16	24	29	3600	57	35	53	53	55	
17	24	30	3600	522	47	193	37	45	37
18	24	30	3600	856	38	38	28	42	33
19	24	30	3600	557	70	53	48	46	39

Legend:	Feasible solution found	No feasible solution found in 1 hour

FIGURE 3.5: Solution time of the problem $Feasible(Q)$ for the real-life instance with $n = 28$ aggregated operations

3.5.2 Quality of heuristics

We compared the solution quality of the heuristics $TopLong(\alpha)$, $TopLongPath(\alpha)$ and $TopRandom(\alpha)$. They were tested on the real-life instance with 170 operations. We made experiments for $\alpha \in \{0, 0.1, 0.2, \ldots, 1\}$. The heuristic TopRandom(α) was applied in two scenarios, with 100 and 1000 runs for each α. The obtained results are demonstrated in the Fig. 3.6. 1000 runs of TopRandom(α) for a given α take about 30 minutes and provide better result than 100 runs of the same heuristic. Heuristic $TopLong(\alpha)$ is not mentioned in Fig. 3.6 because it finds unreasonably high numbers of workers for all values of α.

We also applied heuristics to the problem with the 28 aggregated operations. TopRandom(α) was run 100000 times for each value of α, and

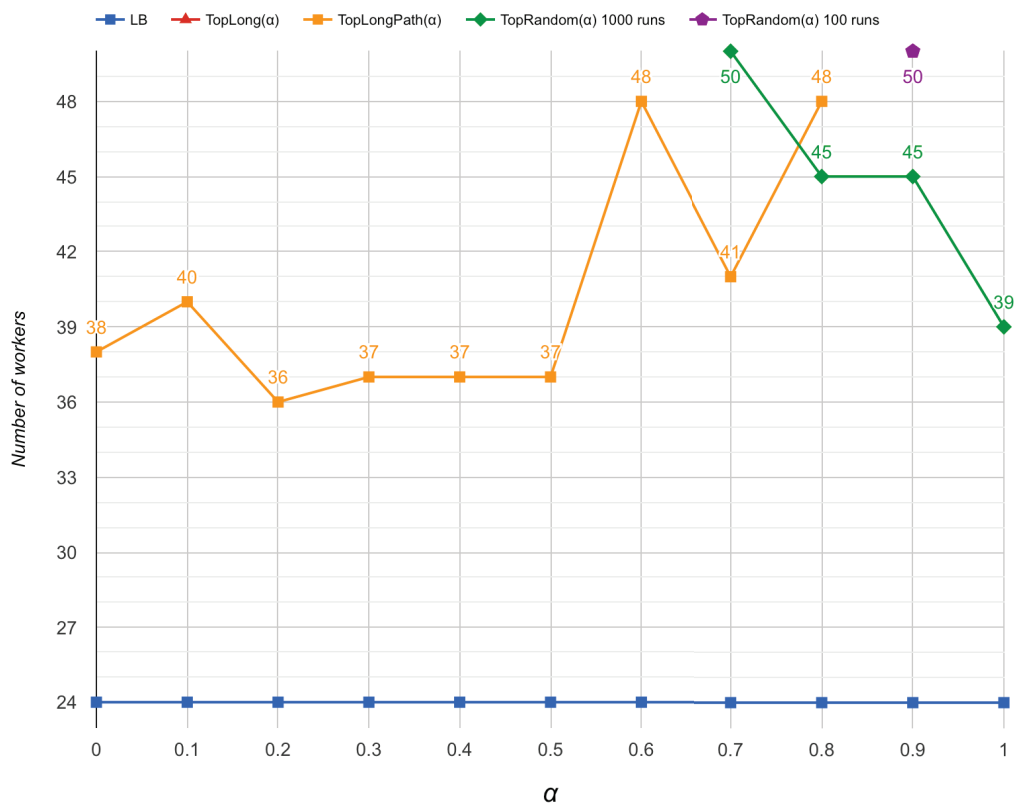


FIGURE 3.6: Heuristic solutions for the real-life instance with $n = 170$

all these runs required less than 30 minutes. The results are given in Figure 3.7.

We noticed that the heuristics were able to find a solution 10000 times faster than the MILP solver for the same problem $Feasible(Q)$. The minimal number of workers delivered by all the heuristics can be used as an upper UB in the bisection search procedure for the exact solution of the problem $MinNumber$.

3.6 Conclusions

Preliminary results of this chapter were described in Dolgui et al. [39].

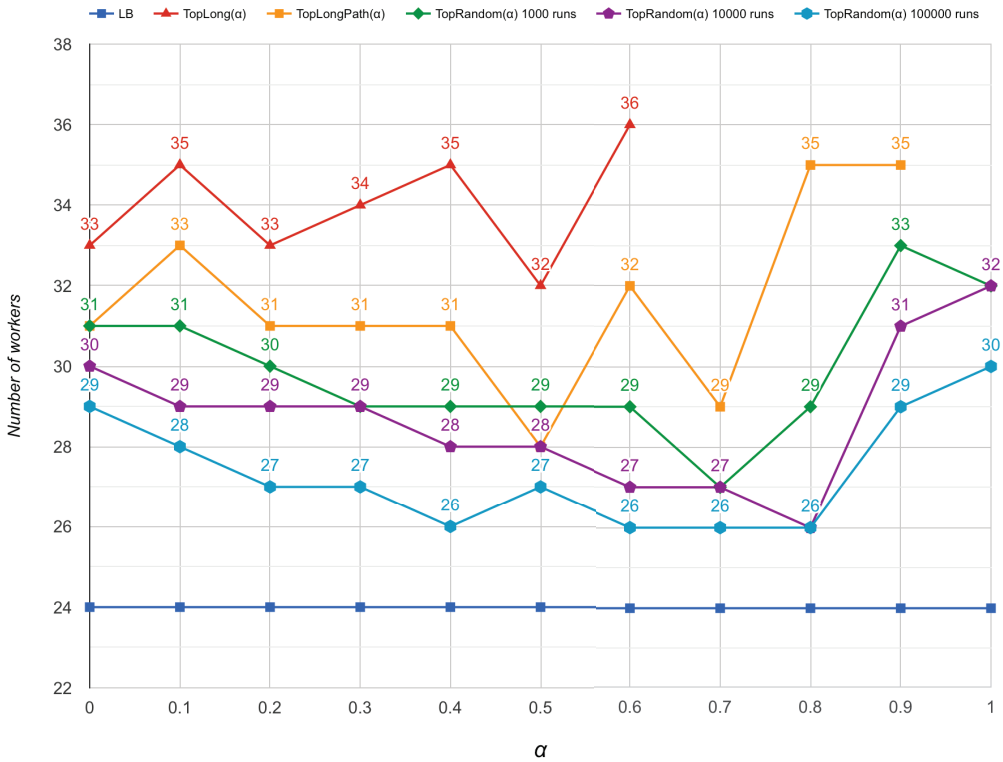


FIGURE 3.7: Heuristic solutions for the real-life instance with $n = 28$ aggregated operations

The following results are obtained for the problem $MinNumber$: two conventional and one randomized heuristics, a bisection search reduction to a series of feasibility problems $Feasible(Q)$, a MILP model for the feasibility problem, a relation of the feasibility problem to the multi-mode project scheduling problems and multiprocessor moldable task scheduling problems, which is used to establish computational complexity of several special cases of the problem $MinNumber$, and computer experiments with the suggested exact and heuristic solution approaches.

Computational experiments demonstrated that the instances with up to 20 operations and 10 workers can be solved to optimality in a reasonable time on a standard computer. They also showed that the quality of the heuristic solutions for the industrial problem with 170 operations is sufficiently good. The obtained results can also be useful for modeling and solving relevant multi-mode project scheduling and moldable task scheduling problems.

Developing meta-heuristic and matheuristic approaches, which are

able to find near optimal solutions of the problem $MinNumber$, is interesting from the practical point of view, and investigating computational complexity of special cases of this problem is interesting from the theoretical point of view. For example, what is the complexity of the problem $MinNumber$, in which $p_j(r) = p/r$ and $r_j \in \{a, b\}$, $j \in N$, for given p, a and b?

Chapter 4

Bi-criteria Transfer Line Balancing Problem

4.1 Introduction

Multi objective transfer line balancing problem for a serial single product production line is studied in this chapter. We consider paced and serial production line, which mean that each part moves from one station to the next one using transport system between those stations composed of conveyors and robots used for part loading and unloading. Stations are equipped with CNC machines. All machines are identical. The goal for machining lines is to assign set of operations to workstations equipped with a set of machines tools satisfying usual assembly lines constraints: processing time of operations and their precedence graph. However for machining lines additional time is required to pass from one operation to the next due to tool changes, displacements and rotations of the part, which is called setup time. Some sides and elements of the part may not be accessible for machining when part is fixed on a station. Therefore, part position should be also considered in the optimization procedure. We consider, that part position can be only changed during transportation from one station to another.

The full list of input data can be described as follows:

- Processing times for all operations (as in usual ALB problem);
- Precedence constraints (used to contract feasible sequences of operations);
- Sequence-dependent setup times (usually presented as a matrix with time values for all combinations of 2 operations);

- Inclusion constraints (some operations must be executed on the sane stations);
- Exclusion constraints (some operations cannot be executed on the same workstations);
- Part fixing constraints (normally a subset from a set of possible part positions is associated with each operation).

Solution (which refers to line configuration) for the studied problem contains information about the following 3 decisions:

- Assignment of the set of operations to workstations and number of those workstations (balancing problem);
- Sequencing of the operations for each station (scheduling problem);
- Choice of the number of CNC machines (equipment problem).

4.2 Aproaches to construct a solution

In this section we will present 3 sub approaches and their combination in a metaheuristic used to solve the described problem.

4.2.1 Construction of a solution for a given sequence

Let's consider a fixed sequence of the operations of set N satisfying precedence constraints. The remaining decisions to be made are:

- Number of workstations and assignment of operations to workstations, which is equivalent to finding the places in the sequence of operations where there is a change of workstation;
- Number of CNC machines and part fixing position for each workstation.

Notations associated with the problem:

- N: set of operations;
- P_i: set of all predecessors of operation with number i;
- ES: set of subsets of set N of operations which must be assigned to the same station (inclusion constraints);

- $\overline{ES}$: set of operation pairs (i, j) which cannot be assigned to the same station (exclusion constraints);
- A: set of all possible part fixing positions;
- A_i: set of part fixing positions for operation i (subset of A);
- t_i: time needed to execute operation i;
- $t_{i,j}$: setup time for switching from operation i to operation j (which means that full time for executing both operation is $t_i + t_{i,j} + t_j$);
- $TMAX$: upper bound for line cycle time;
- CoS: cost of one CNC machine;
- $CoMAX$: upper bound on total investment cost;
- $M = \{1, \ldots, MaMAX\}$: set of possible numbers of machines on a station;

The multi-objective optimisation problem can be formulated with the following MIP:

$$\min \sum_{i=1}^{n} (c_S.\sigma_i + c_M.\mu_i) \tag{4.1}$$

s.c.

$$\sigma_i \geq Tc - TMAX * (1 - \sum_{c=1}^{c_0} x_{i,c}), i = 1, \ldots, n \tag{4.2}$$

$$\mu_i \geq c * (Tc - TMAX * (1 - \sum_{c'=c}^{c_0} x_{i,c'})), i = 1, \ldots, n, c = 1, \ldots, c_0 \tag{4.3}$$

$$c_S * \sum_{i=1}^{n} \sum_{c=1}^{c_0} x_{i,c} + c_M * \sum_{i=1}^{n} \sum_{c=1}^{c_0} c * x_{i,c} \leq CoMAX \tag{4.4}$$

$$Tc \leq TMAX \tag{4.5}$$

$$\sum_{i=1}^{n} \sum_{c=1}^{c_0} x_{i,c} \leq w_0, \tag{4.6}$$

$$\sum_{c=1}^{c_0} x_{i,c} \leq 1, i = 1, \ldots, n-1 \tag{4.7}$$

$$\sum_{c=1}^{c_0} x_{n,c} = 1 \tag{4.8}$$

$$\sum_{i=a}^{b-1} \sum_{c=1}^{c_0} x_{i,c} \geq 1, (a, b) \in \overline{\mathcal{ES}} \tag{4.9}$$

$$\sum_{i=\min_{j \in \mathcal{I}} \{j\}}^{\max_{j \in \mathcal{I}} \{j\}} \sum_{c=1}^{c_0} x_{i,c} = 0, \mathcal{I} \in \mathcal{ES} \tag{4.10}$$

$$\tau_i \geq \tau_{i-1} + t_i + t_{i-1,i} - M. \sum_{c=1}^{c_0} x_{i-1,c}, i = 2, \ldots, n \tag{4.11}$$

$$\tau_i \leq c.Tc + M(1 - \sum_{c'=1}^{c} x_{i,c'}), i = 1, \ldots, n, c = 1, \ldots, c_0 \tag{4.12}$$

$$\sum_{j=i}^{i+n_0-1} \sum_{c=1}^{c_0} x_{j,c} \geq 1, i = 1, \ldots, n - n_0 + 1 \tag{4.13}$$

$$\sum_{j=i}^{\max\{l \in \{i,\ldots,n\} | \cap_{r \in [i,l]} \mathcal{A}_r \neq \emptyset\}} \sum_{c=1}^{c_0} x_{j,c} \geq 1, i = 1, \ldots, n \tag{4.14}$$

$$x_{i,c} \in \{0, 1\}, i = 1, \ldots, n, c = 1, \ldots, c_0 \tag{4.15}$$

$$\tau_i \geq t_i, i = 1, \ldots, n \tag{4.16}$$

$$Tc \geq 0 \tag{4.17}$$

In the described model the decision variables $x_{i,k}$ are equal to 1 if operation i is the last operation assigned to a workstation equipped with k CNC machines, τ_i is the workload time accumulated on the current workstation up to operation i, and T corresponds to the cycle time of the line.

σ_i is equal to the cycle time of the line if operation i is the last operation signed to a workstation and 0 otherwise, μ_i is equal to the cycle time of the line multiplied by the number of CNC machines which equip the current workstation if operation i is the last operation assigned to a workstation and 0 otherwise. However solution of the model does not have information about part fixing position, the constrain 4.14 ensures that there is feasible solution in the meaning of part position for each workstation.

4.2.2 Local search on the number of machines

Let's consider an assignment of all operations to workstation is known and fixed. The remaining decisions in this case are:

- Sequence of operations on each workstation;
- Number of CNC machines used on each workstation;
- Part fixing position on each workstation (which is trivial due to known assignment of operations).

The first decision corresponds to a single machine scheduling problem with sequence-dependent setup time and precedence constraints which has to be solved for each workstation. As shown by (Bigras et al., 2008), this problem is equivalent to a time-dependent traveling salesman problem for which various MIP formulations and algorithms have been proposed.

The second decision can be obtained with the following algorithm:

- Generate S_1 with 1 CNC machine per workstation
- Determine the workstation w_1 with the largest workload
- Set i to 1
- Until w_i has the maximal number of CNC machine repeat
  - Generate S_{i+1} by adding 1 CNC machine to w_i
  - Determine the workstation w_{i+1} with the largest local cycle time
  - Increment i
- End repeat

4.2.3 Combining lines

Let's consider a solution corresponding to a feasible RML denoted X. The cost and the cycle time of this solution are denoted $C(X)$ and $T(X)$, respectively. In case we are searching for a solution with lower cycle time than cycle time of solution X, we can a production system $X^{(2)}$ composed of two identical production lines X working in parallel. The cost of the obtained line will be twice more than the cost of X, but at the same time the cycle time of $X^{(2)}$ will be just a half of X.

Similar to the above, we can consider a production system of 2 parallel production lines X and Y. Defining the combination of lines X and Y as $(X + Y)$, the cost and cycle time of combined system can be calculated as follows:

$$\begin{cases} C(\langle X + Y \rangle) = C(X) + C(Y) \\ T(\langle X + Y \rangle) = \dfrac{T(X) \times T(Y)}{T(X) + T(Y)} \end{cases} \tag{4.18}$$

We can easily demonstrate that the cycle time of the combined solution is al- ways lower than those of each initial line, which means that the solution generated by combination neither dominates nor is dominated by any of the initial solutions.

4.2.4 Greedy Randomised Adaptive Search Procedure

We propose the following metaheuristic algorithm, which is based on random generation of set of initial allowed sequences and applying 3 mentioned above approaches to obtain and improve solutions for the overall studied multi-objective problem:

- Create a pool of potentially non-dominated solutions
- Until stopping criteria repeat
  - Generate a random sequence with respect to the precedences
  - Check the feasibility of the sequence for inclusion, exclusion and accessibility constraints
  - Solve the MIP with a solver (section 4.2.1)
  - Apply local search procedure (section 4.2.2)
  - Update the pool

  - While new solutions are added to the pool do
    - For each pair of solutions in the pool do
      - Combine them to generate a new solution (section 4.2.3)
      - Update the pool
    - End do
  - End do

- End repeat

4.3 Computational Experiments

We have tested proposed approach on an example of reconfigurable transfer line where 26 operations has to be performed to produce a product. Precedence graph for the operations has 14 direct relations. There are 2 exclusion sets, 3 inclusion sets and 3 part-fixing positions. All computer experiment were performed on 2,6 GHz Intel Core i7 mobile processor equipped with 8 GB 1600 MHz DDR3 RAM memory and SSD disk. IBM CPLEX solver were used for solving MIP problems. The final implementation of GRASP procedure including all sub approaches were implemented in C#

Studied as an example production line has the following key parameters and restrictions:

- Costs
  - One workstations - 50,000
  - One CNC machine - 200,000
- Upper bounds
  - Cycle time - 40
  - Cost - 10,000,000
  - Machines per workstation - 3
  - Operations per workstation - 10
  - Workstations per line - 10
  - Parallel lines - 3

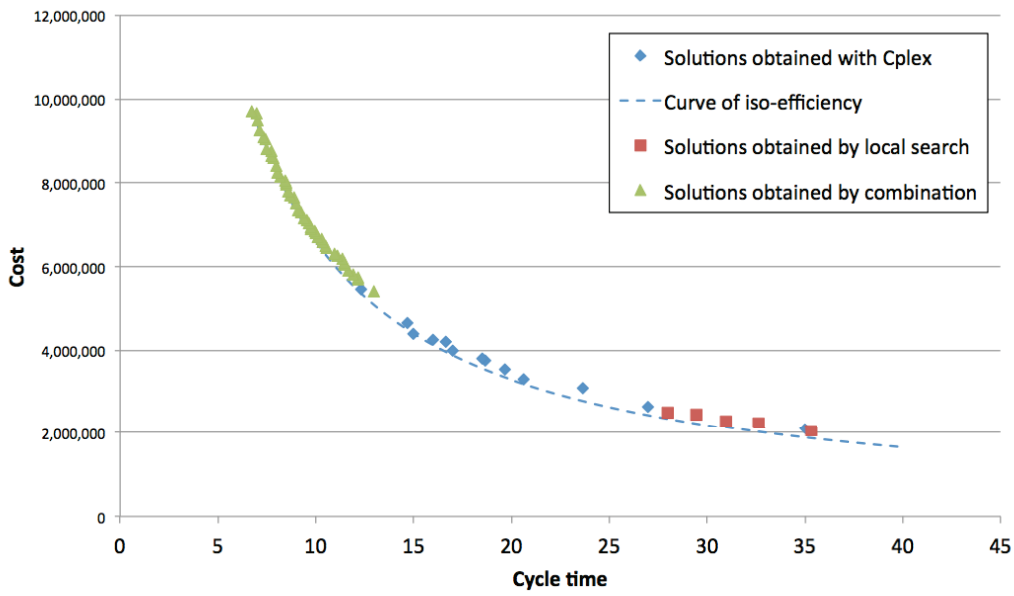


FIGURE 4.1: Analysis of the final Pareto front obtained

On the Figure 4.1 we provide the obtained non dominated solutions and final Pareto front based on those solutions.

If the solution generated with CPLEX are mostly at the center of the Pareto front, it is interesting to note that local search tends to obtain solutions with lower cost and higher cycle time, but very similar efficiency. Solution combination provides higher efficiency in the range of lower cycle time and higher total cost.

4.4 Conclusions

Preliminary results of this chapter were described in [32].

In this chapter we addressed several problems arising during single product transfer line design. By solving single-objective MIP model for a previously generated feasible random sequence of operation we solved balancing problem (assignment of operations to stations) and equipment problem (Choice of number of machines). During local search procedure we also solved a scheduling problem (choosing sequence of operation) for each station (but not globally for entire line). Combination of lines into parallel production lines reduce cycle time of generated production system. Integration of all described approaches in a meta heuristic allowed

us to solve line balancing problem for reconfigurable transfer line taking into account two main criteria (total cost and cycle time) at the same time.

Multi-objective optimisation methods for assembly lines are relevant, because they better addresses needs of manufactures, those are trying to minimise all spending resources to achieve maximum gain. Another important performance indicator for industry is a how fast line can be reconfigured to produce a new product, which can be characterised by level of reconfigurability, which was not considered in this chapter, but is a subject for future research.

Chapter 5

amePLM project

5.1 Introduction

Developed techniques and results obtained in Chapter 2 were used in european project amePLM [1]. amePLM (advanced platform for manufacturing engineering and PLM, FoF-ICT-285171) is a project funded by the European Commission in the frame of the Factories-of-the-Future Public-Private-Partnership. The project began at October 1st, 2011 and had a duration of almost 4 years and ended in May 2015. amePLM brings together an exciting project consortium comprising industrial partners - Intel, MB Technology, RTT, Aerogen and Shannon Coiled Springs - with research and development partners - Fraunhofer IAO, Ontoprise, Politecnico di Torino, Armines, University of Limerick, University of Nottingham, and Universita degli Studi di Trietse.

The development of products and productions in industrial companies can be characterized by large amounts of information from a variety of disciplines and backgrounds, created and processed by a multitude of methods and tools that have to be considered to realise new products in short time-to-market and time-to-production in a cost-effective and resource-optimized manner. Therefore, the focus of amePLM is to support knowledge-based cross-disciplinary collaborative engineering. Essential challenges of this focus include:

- engineering in development and manufacturing is often done in a multidisciplinary manner;
- decision making in engineering often involves specialists from different backgrounds, organisational departments and potentially locations;
- engineering is often done in company networks or clusters, leading to a variety of methods, tools and systems that are concerned to a

situation that is comparable to large companies, where a multitude of systems are used;

- the support of engineering activities by different tools and systems results in information flows that have to be seamlessly realised through the concerned systems by continuous engineering workflows;

- experiences and knowledge of all partners along the lifecycle of a product have to be considered already in the early phases of manufacturing and engineering to facilitate short time-to-market and -to-production by less iteration cycles and assuring cost-effective high quality products that are needed to support the competitiveness of European industrial companies, especially when looking to emerging and existing competitors from the BRIC countries.

5.2 Objectives of the project

amePLM targets the support of product and production engineers by a radically new and extensible approach to collaborative engineering of products and productions that leverages state-of-the art research on semantics, heuristics and visualization. The main work areas of amePLM are therefore:

- to support users with appropriate knowledge, experiences and information, relieve them from extensive manual retrieval of information,,

- to facilitate continuous information flows and workflows in engineering,

- to provide users with situation- and context-specific solution method support for complex tasks and decisions in design, analysis, virtual testing and optimization in engineering,

- to enable users to handle, investigate and manipulate complex and large sets of information by appropriate visualisation support, and

- to support complex collaborative tasks of decision making by appropriate methods and tools.

One of the key objectives for industrial partners in the amePLM project, which represents a subset of industry needs, is to reduce the time for information retrieval for engineers and accelerating the product and product engineering through capturing and re-use of knowledge and experiences along the product lifecycle.

5.3 Intelligent Information Layer

The amePLM ontologies are the basis of the amePLM platform. The first kind of ontology, i.e., the data model ontology, is accessed by the application modules through the semantic backend, which is the set of functions available to insert data as ontology instances and retrieve the information previously stored. The second kind of ontology, i.e., the taxonomy, is used to tag the documents and any other knowledge item involved in the product lifecycle of the company. This taxonomy is also used to perform semantic searches among them. The ontology-set is utilised by different parts of the amePLM intelligent information layer (IIL) as can be seen in Figure 5.1, which specifies as the amePLM knowledge structure.

There are several pilot cases studied during the amePLM project. Later in this chapter MB-T pilot that support the company during the optimisation of a virtual production facility will be discovered. There are models for each pilot, those represent the data structure for the information that have to be stored for each pilot case in order to allow the efficient management of its lifecycle, accordingly with the requirements expressed by the five pilots. The information stored according to these models can be accessed and managed by the other modules through the Intelligent Information Layer, and particularly the semantic backend. All the other modules and components of the amePLM platform will consider these models as a reference to store and retrieve data. One of the major benefits of the ontologies is that they allow the linking of different product lifecycle stages within the platform, while the domain ontologies serve as metadata repositories to give the individual knowledge items and documents a meaning within the platform.

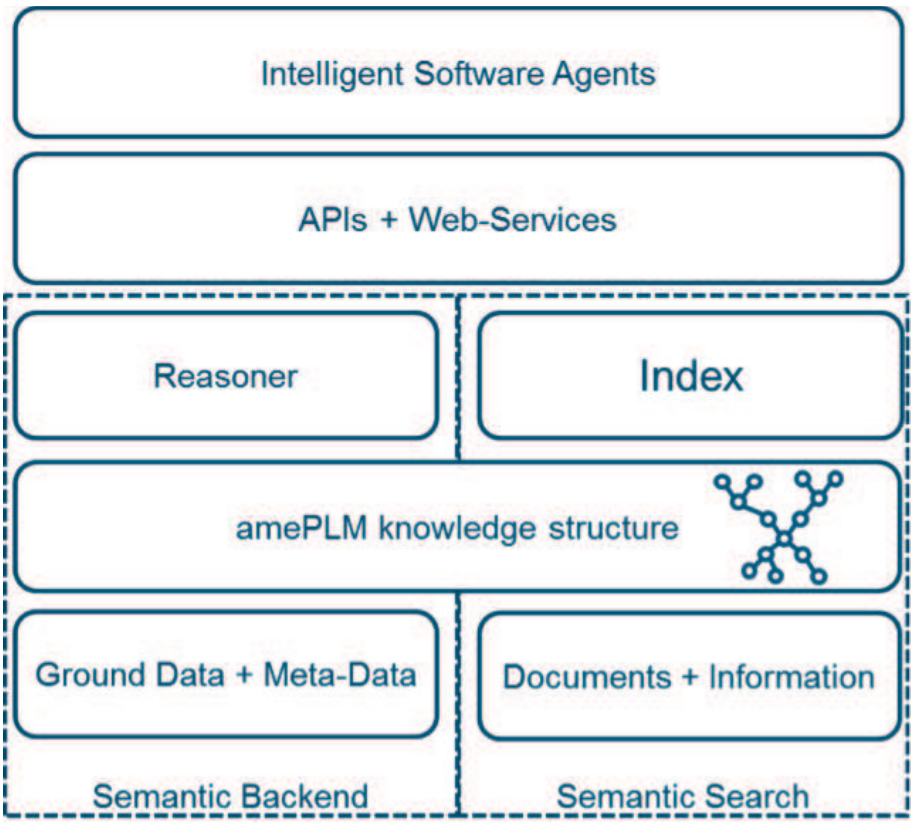


FIGURE 5.1: Structure of Intelligent Information Layer

5.4 3D workspace

The amePLM consortium proposed the concept of the 3D workspace as a means to create containers for a specific work context, into which artefacts supporting that context can be assembled, providing an environment where the computer becomes more like an intelligent assistant, allowing the user to explore, refine, annotate and synthesise their knowledge.

The main aim of the amePLM platform is not to substitute existing solutions used within companies but seamlessly integrate information and knowledge coming from different stages of the product life cycle (PLC) and support the creation, sharing, retrieval and preservation of knowledge along the PLC. The 3D workspace created to support such integrated 'Information Ecology' experience (Find it, Keep it, Build on it, Share it) in a unifying way across an engineer's environment of information, particularly when combined with an intelligent information layer (IIL) (see Figure 5.2).

FIGURE 5.2: Information Ecology within amePLM

The 3D workspace Miramar is a client side application. With the 3D workspace being a key user interface to access the various modules of the amePLM platform, it has been used across all trials to enable users to interact with the amePLM Engineering platform.

5.5 Developed optimisation module and its integration

Developed heuristic methods were implemented inside the optimisation module, which was integrated to amePLM platform. This module has the structure shown in Figure 5.3. The optimisation module was tested and demonstrated on MBTech pilot.

Due to its architecture, the optimisation module can easily communicate with the amePLM ontology and other engineering modules of the platform. The front-end integration of the optimisation module is implemented via Miramar (an example is shown in Figure 5.4), which is done by hosting the optimisation module on a web server and allowing miramar to access it via web url from the Miramar internal web browser. And the

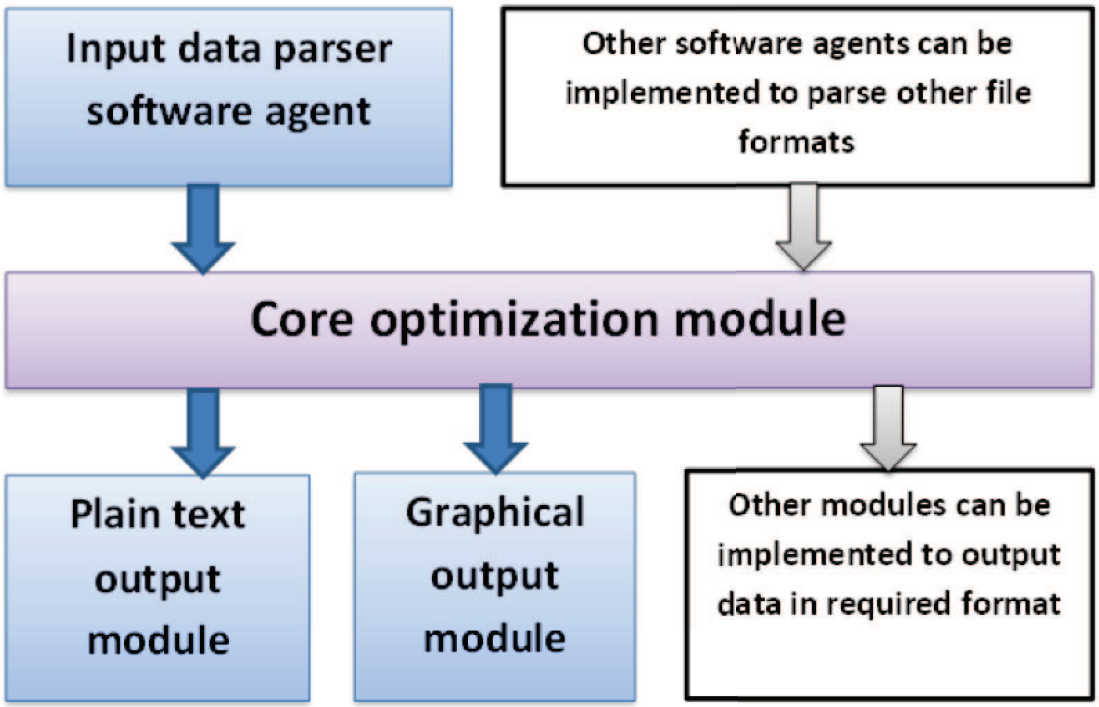


FIGURE 5.3: Structure of the optimisation module

backend Integration of the optimisation module is attained via an Intelligent Information Layer (IIL) by using the API of IIL to retrieve all the files already added and stored inside the platform related to optimization. All the files containing input data can be solved with optimisation module. Obtained results can be saved back to IIL.

The amePLM platform is used to store all the information relevant for the product lifecycle via Miramar into IIL. They can be retrieved through the Intelligent Information Layer search functionality and used to create input file for the optimisation tool. After the optimisation is finished, the results can be stored as information item within Intelligent Information Layer via Miramar and other meta-data can be added for future information retrieval. Therefore, users may use the optimisation module through the platform in various ways:

1. The data for the optimisation module can be extracted from amePLM ontology.

2. The input and output files used by the optimisation module can be

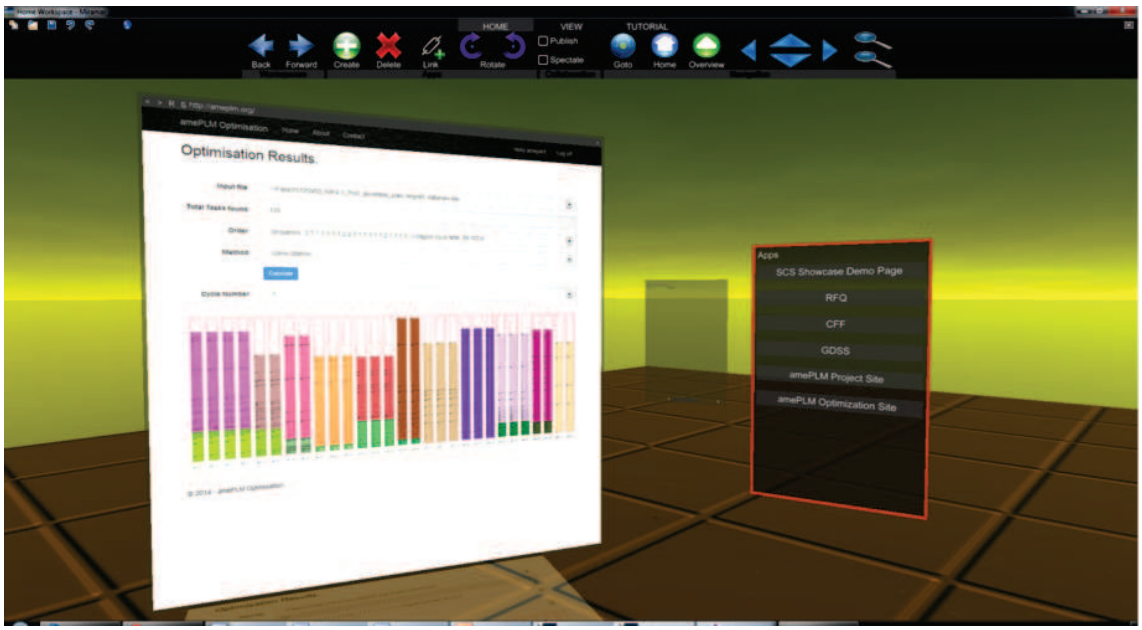


FIGURE 5.4: Optimisation module in Miramar environment

stored in the platform and enriched by meta-data to facilitate the information retrieval.

3. The optimisation results stored in the platform can be used for other platform modules.

For each product sequence, the module will screen the data describing worker's utilization in text (Figure 5.5) and visual (Figure 5.6) formats.

The aim of text format is to list all the workers and for each worker provide the list of task he needs to perform or to list all the tasks and provide the information to each of it about home many workers are needed and exact worker numbers. Such output (both workers list and tasks list) provides the full information about the obtained solution and can be used as an input file for any other tools. At the same time, the visual format helps an end user to analyse the obtained solution. This colorful visualization format has been conceived in the following way:

1. Each bar represent one worker. For example, 26 bars in Figure 11 represent 26 workers needed for the line designed.

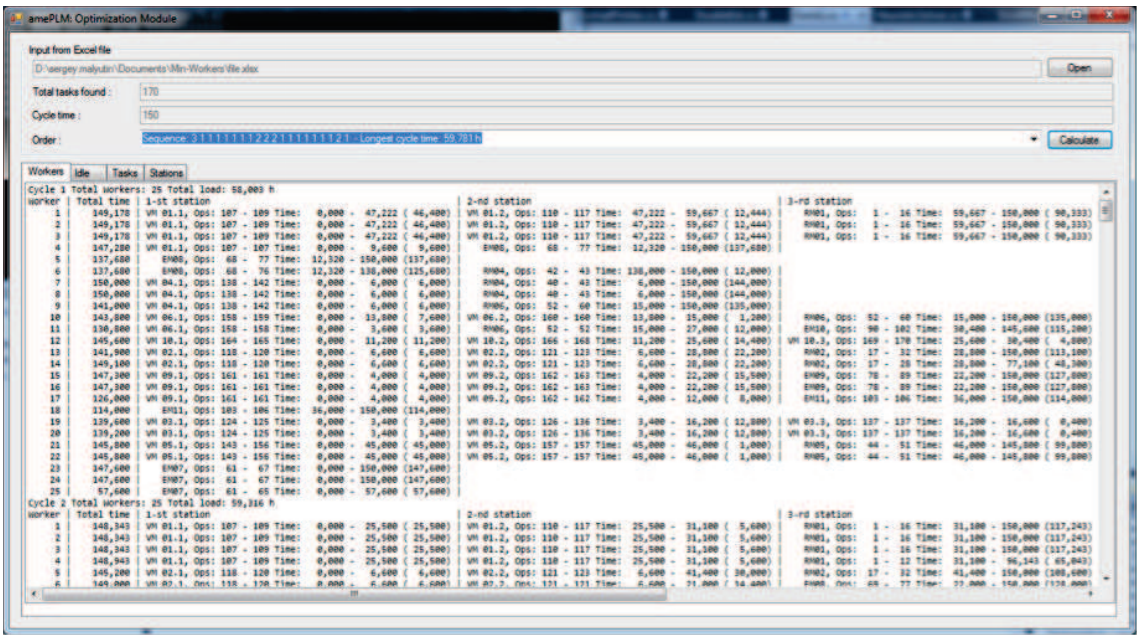


FIGURE 5.5: Results in text format

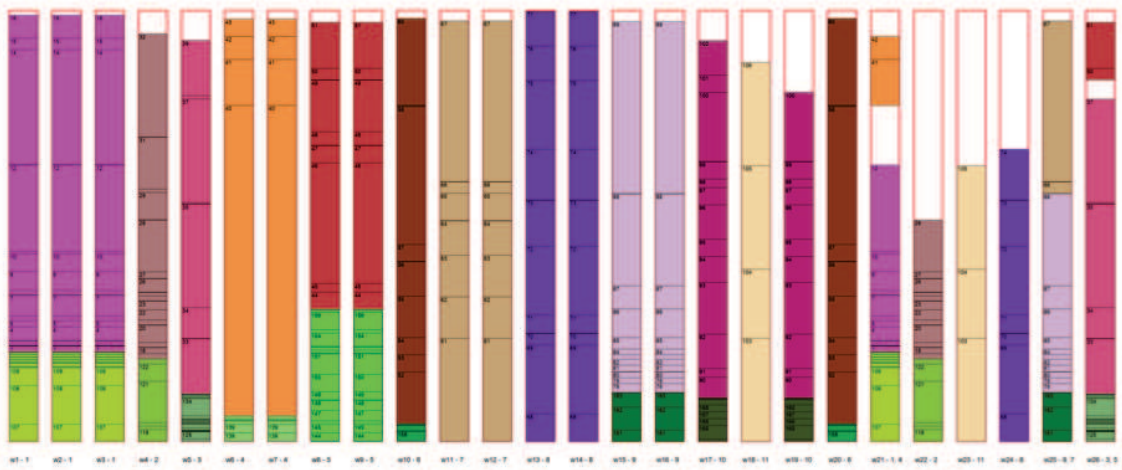


FIGURE 5.6: Results in visual format

2. Each workstation is represented by its own color. For example, purple color of the first 3 bars corresponds to RM01 main station and green color corresponds to VM 01.1 substation. Colors of the 4-th bars correspond to the stations RM02, VM 02.1 and VM 02.2 and etc.

3. All green tones correspond to sub stations and all non-green to main stations.

4. The vertical axis of the bar means time. Full height of the bar corresponds to the cycle time, so all blanc (white) spaces are idle times.

5. There are also sub bars with a label inside them. Each sub bar corresponds to a task performed by the worker and a label inside indicates the task number (see Figure 5.7).

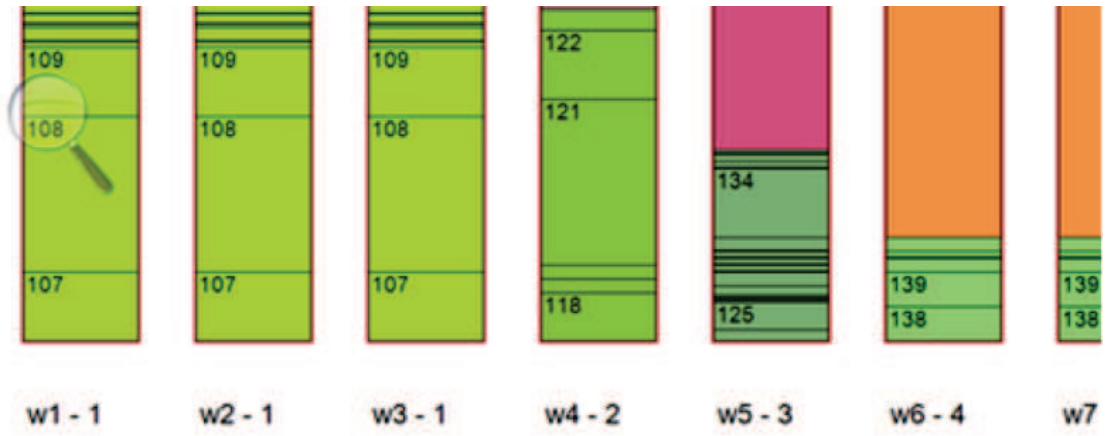


FIGURE 5.7: Visualisation of individual tasks

6. Each worker starts to perform his/her list of the task from the bottom. And he/she performs the tasks one by one in ascending order until the task at the top is completed.

7. Labels below the bars have the format like 'wN-M', where N indicates the number of the worker and M stands for the station number. For example, w4-2 means that the 4-th worker is performing tasks on the station 2 (both substation and main station). The label like w21-1, 4 means that the worker with number 21 works on both 1st and 4th stations.

8. Each picture illustrates only one cycle. There is a dropdown to select the cycle to show (see Figure 5.8).

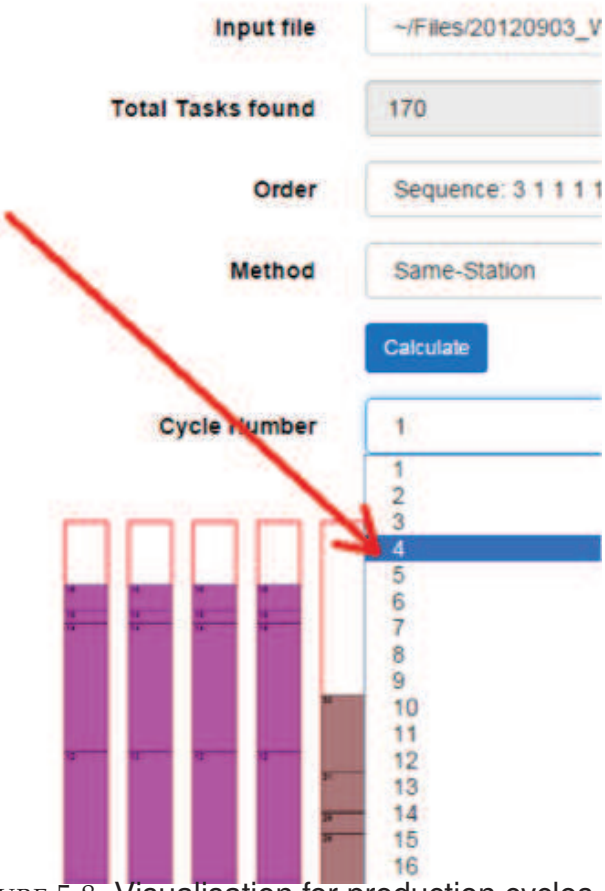


FIGURE 5.8: Visualisation for production cycles

5.6 Conclusions

Since the line configuration problem is highly combinatorial, the optimal solution is hard to find, in practice for our industrial partner this results in the approach 'Trial and error' validated by a simulation model. The use of an optimisation model before the simulation helps to reduce the number of trials and to save the time in finding the best solution. Therefore, the use of the optimisation module helps to improve both the quality of the solution obtained and the time spent on the project.

A component based approach chosen for the platform consisting of the IIL functioning as semantic middleware, the 3D workspace as central user interface to facilitate collaborative and distributed knowledge-capturing along the PLC and engineering models serving as interoperable models

for capturing and re-usage of product and manufacturing related engineering knowledge allowed to build a flexible and extensible platform. The optimisation module (as one of the engineering modules) successfully integrated in the platform amePLM and available as service for industrial partners. The example of its integration can serve for further integration of other modules available as service in the platform.

General Conclusions

From the perspective of the modern advanced technology and production, initial assembly lines were simple, small amount of non complicated operations were executed. Even if any kind of optimisation of production line was used, it was applied without any electronic devises. Computer revolution had changed a lot. It influenced on the assembly lines and allowed to create more complex production lines controlled by computers. And vice versa evolution of production lines leads to more advanced electronic systems. Complex mathematical models and methods were developed to reflect new generations of complicated production systems. Solving new problems by humans without the use of special tools is no longer possible. Many programming languages were created in order to implement developed methods and algorithms in a software tools. Those software tools consist of a machine code representing developed methods understandable by computer and output data with a solution in human readable format. That allowed to use computational power for solving optimisation problems for production systems. The final decision making for manufactures is based on results of solving optimisation problem and another factors such as financial, technological, economical, payback time and so on. This thesis is a synthesis of mathematical methods and their implementations inside software tools for decision support is design of assembly and transfer lines.

The results of the thesis are the following.

- New workforce minimisation problem for semi automatic multi product paced assembly line with chains precedence relations and condition that several identical workers can be assigned to one station was solved. Computer implementation of developed heuristic methods was integrated to a software tool with graphical user interface for personal computers equipped with Windows operation systems. That tool was supplied to our industrial partner in order to help in decision making during final procedure of production line design.

- New problem of minimisation of number of workers for multi product serial assembly line with arbitrary acyclic graph was solved by a bisection search reduction to a series of feasibility problems $Feasible(Q)$. Problems $Feasible(Q)$ solved optimally with a developed MIP approach. Computer experiments of MIP model and developed heuristic methods were performed.

- New multi objective approach for single product transfer line balancing problem was developed and tested on a computer. Method includes solving of scheduling for each station problem, balancing and equipment problems for entire production line.

- Another web based software tool with a core optimisation module based on workforce minimisation methods was created and integrated inside amePLM platform during european project.

Appendix A

Computer experiment results for problem $Feasible(Q)$ with 20 operations

#	Number of Stations	Sum of task times	LB	UB	q	Idle (%)	$Feasible(q)$	Time (s)
1	4	2882	3	6	3	3.93%	TRUE	1.58
2	3	2861	3	5	3	4.63%	TRUE	168.37
3	3	2785	3	5	3	7.17%	TRUE	6.97
4	3	2727	3	4	3	9.10%	TRUE	1.92
5	3	2837	3	4	3	5.43%	TRUE	2.17
6	2	2914	3	4	3	2.87%	TRUE	2.61
7	3	2786	3	5	3	7.13%	TRUE	1.45
8	1	2985	3	3	3	0.50%	FALSE	3598.58
9	2	2925	3	4	3	2.50%	TRUE	17.04
10	4	2831	3	6	3	5.63%	TRUE	2.76
11	3	2766	3	5	3	7.80%	TRUE	1.58
12	3	2930	3	4	3	2.33%	TRUE	34.06
13	3	2896	3	4	3	3.47%	TRUE	6.78
14	3	2979	3	4	3	0.70%	TRUE	30.77
15	4	2981	3	6	3	0.63%	TRUE	2943.98
16	3	10376	11	11	11	5.67%	FALSE	3599.58
17	4	9134	10	11	10	8.66%	TRUE	2390.16
18	2	9524	10	11	10	4.76%	FALSE	3599.91
					11	13.42%	FALSE	3599.98
19	2	11113	12	12	12	7.39%	FALSE	3599.98

20	3	9829	10	11	10	1.71%	FALSE	3599.98
					11	10.65%	TRUE	2258.16
21	3	11256	12	13	12	6.20%	FALSE	3599.88
					13	13.42%	FALSE	3599.27
22	2	10171	11	11	11	7.54%	FALSE	3598.58
23	4	10374	11	12	11	5.69%	FALSE	3601.54
					12	13.55%	FALSE	3600.08
24	4	9982	10	11	10	0.18%	FALSE	3600.50
					11	9.25%	FALSE	3600.70
25	2	9670	10	10	10	3.30%	FALSE	3600.64
26	3	10110	11	11	11	8.09%	FALSE	3600.61
27	3	10521	11	12	11	4.35%	FALSE	3601.83
					12	12.33%	FALSE	3600.64
28	2	10138	11	11	11	7.84%	FALSE	3602.74
29	5	9079	10	11	10	9.21%	TRUE	836.63
30	3	11578	12	13	12	3.52%	FALSE	3600.60
					13	10.94%	FALSE	3600.74
31	3	10270	11	12	11	6.64%	TRUE	3052.23
32	1	10489	11	11	11	4.65%	FALSE	3600.57
33	3	9944	10	11	10	0.56%	FALSE	3600.84
					11	9.60%	FALSE	3600.63
34	3	10203	11	12	11	7.25%	FALSE	3600.60
					12	14.98%	FALSE	3600.83
35	1	9906	10	10	10	0.94%	FALSE	3600.48
36	2	10314	11	12	11	6.24%	FALSE	3600.49
					12	14.05%	FALSE	3600.54
37	3	10328	11	12	11	6.11%	FALSE	3600.46
					12	13.93%	FALSE	3612.06
38	3	10083	11	11	11	8.34%	FALSE	3600.59
39	2	10662	11	11	11	3.07%	FALSE	3600.68
40	3	10193	11	11	11	7.34%	FALSE	3600.66
41	5	5156	6	8	6	14.07%	TRUE	36.06
42	3	4391	5	6	5	12.18%	TRUE	32.00
43	4	4688	5	6	5	6.24%	TRUE	106.77
44	4	4561	5	7	5	8.78%	TRUE	5.80
45	3	5320	6	7	6	11.33%	TRUE	75.05

46	2	3540	4	5	4	11.50%	TRUE	4.13
47	3	3690	4	5	4	7.75%	TRUE	14.70
48	3	4851	5	7	5	2.98%	FALSE	3600.23
					6	19.15%	TRUE	3.44
49	3	3888	4	5	4	2.80%	TRUE	35.63
50	4	3576	4	6	4	10.60%	TRUE	5.05
51	2	3701	4	5	4	7.48%	TRUE	334.39
52	4	3901	4	6	4	2.48%	TRUE	191.20
53	4	4501	5	7	5	9.98%	TRUE	99.02
54	3	4192	5	6	5	16.16%	TRUE	10.02
55	3	4492	5	6	5	10.16%	TRUE	5.45
56	2	3865	4	5	4	3.38%	TRUE	2.63
57	3	3714	4	5	4	7.15%	TRUE	4.34
58	3	4296	5	5	5	14.08%	TRUE	12.66
59	4	3985	4	7	4	0.38%	FALSE	3600.39
					5	20.30%	TRUE	3.34
60	4	5144	6	8	6	14.27%	TRUE	18.20
61	2	6202	7	7	7	11.40%	TRUE	526.35
62	4	4276	5	6	5	14.48%	TRUE	11.69
63	5	4650	5	8	5	7.00%	TRUE	7.41
64	3	4877	5	7	5	2.46%	TRUE	71.97
65	3	4564	5	6	5	8.72%	TRUE	11.95
66	1	2743	3	3	3	8.57%	TRUE	2.33
67	1	2942	3	3	3	1.93%	TRUE	23.25
68	1	2906	3	3	3	3.13%	TRUE	44.84
69	1	1972	2	2	2	1.40%	TRUE	0.37
70	1	2985	3	3	3	0.50%	TRUE	245.49
71	1	2957	3	3	3	1.43%	TRUE	64.00
72	1	2827	3	3	3	5.77%	TRUE	26.76
73	1	1933	2	2	2	3.35%	TRUE	0.42
74	1	2792	3	3	3	6.93%	TRUE	2.83
75	2	2886	3	4	3	3.80%	TRUE	9.95
76	1	2903	3	3	3	3.23%	TRUE	2.67
77	1	2939	3	3	3	2.03%	TRUE	19.77
78	1	2810	3	3	3	6.33%	TRUE	3.27
79	1	2945	3	3	3	1.83%	TRUE	75.00

80	1	2914	3	3	3	2.87%	TRUE	98.08
81	1	2937	3	3	3	2.10%	TRUE	64.72
82	1	3505	4	4	4	12.38%	TRUE	4.64
83	1	2847	3	3	3	5.10%	TRUE	165.96
84	1	2932	3	3	3	2.27%	TRUE	232.40
85	1	2817	3	3	3	6.10%	TRUE	16.81
86	1	2912	3	3	3	2.93%	TRUE	566.39
87	1	2824	3	3	3	5.87%	TRUE	15.94
88	1	2727	3	3	3	9.10%	TRUE	9.77
89	1	2877	3	3	3	4.10%	TRUE	12.17
90	1	2807	3	3	3	6.43%	TRUE	11.08
91	1	9478	10	10	10	5.22%	FALSE	3600.81
92	1	9786	10	10	10	2.14%	FALSE	3600.66
93	1	10465	11	11	11	4.86%	FALSE	3600.56
94	1	9224	10	10	10	7.76%	FALSE	3600.48
95	1	9527	10	10	10	4.73%	FALSE	3600.55
96	1	9537	10	10	10	4.63%	FALSE	3600.43
97	1	10460	11	11	11	4.91%	FALSE	3600.55
98	1	10293	11	11	11	6.43%	FALSE	3600.56
99	1	10219	11	11	11	7.10%	FALSE	3600.54
100	1	9448	10	10	10	5.52%	FALSE	3600.42
101	1	9979	10	10	10	0.21%	FALSE	3600.51
102	1	10067	11	11	11	8.48%	FALSE	3600.41
103	1	9823	10	10	10	1.77%	FALSE	3600.43
104	1	9561	10	10	10	4.39%	TRUE	788.76
105	1	10532	11	11	11	4.25%	FALSE	3600.36
106	2	9330	10	10	10	6.70%	FALSE	3600.49
107	1	10770	11	11	11	2.09%	FALSE	3600.54
108	1	10672	11	11	11	2.98%	FALSE	3600.43
109	1	10183	11	11	11	7.43%	FALSE	3600.75
110	2	9743	10	11	10	2.57%	TRUE	448.80
111	1	10295	11	11	11	6.41%	FALSE	3600.75
112	1	9815	10	10	10	1.85%	FALSE	3600.67
113	1	9741	10	10	10	2.59%	FALSE	3600.55
114	1	10203	11	11	11	7.25%	TRUE	406.22
115	1	9661	10	10	10	3.39%	FALSE	3600.51

116	1	4592	5	5	5	8.16%	TRUE	8.28
117	1	4641	5	5	5	7.18%	TRUE	126.53
118	1	4546	5	5	5	9.08%	TRUE	61.09
119	1	5779	6	6	6	3.68%	TRUE	322.28
120	1	5485	6	6	6	8.58%	TRUE	135.09
121	1	4888	5	5	5	2.24%	TRUE	797.33
122	1	5277	6	6	6	12.05%	TRUE	115.97
123	1	4514	5	5	5	9.72%	TRUE	25.27
124	1	4226	5	5	5	15.48%	TRUE	21.33
125	1	4231	5	5	5	15.38%	TRUE	6.08
126	1	4411	5	5	5	11.78%	TRUE	30.50
127	1	3634	4	4	4	9.15%	TRUE	4.66
128	1	4615	5	5	5	7.70%	TRUE	58.88
129	1	4639	5	5	5	7.22%	TRUE	2650.91
130	1	5381	6	6	6	10.32%	TRUE	995.47
131	1	6542	7	7	7	6.54%	FALSE	3612.84
132	1	3614	4	4	4	9.65%	TRUE	35.89
133	1	4580	5	5	5	8.40%	TRUE	86.50
134	1	5262	6	6	6	12.30%	TRUE	49.98
135	2	5041	6	6	6	15.98%	TRUE	82.45
136	1	4946	5	5	5	1.08%	TRUE	734.02
137	1	4677	5	5	5	6.46%	TRUE	207.26
138	1	4852	5	5	5	2.96%	TRUE	277.65
139	1	4818	5	5	5	3.64%	FALSE	3600.25
140	1	4763	5	5	5	4.74%	TRUE	294.20
141	4	2908	3	4	3	3.07%	TRUE	4.84
142	4	2808	3	6	3	6.40%	TRUE	1.50
143	4	2934	3	5	3	2.20%	TRUE	9.05
144	2	3507	4	4	4	12.33%	TRUE	3.33
145	3	2966	3	5	3	1.13%	TRUE	23.64
146	4	2672	3	5	3	10.93%	TRUE	1.70
147	6	2857	3	7	3	4.77%	TRUE	8.37
148	5	2852	3	5	3	4.93%	TRUE	3.78
149	5	2938	3	6	3	2.07%	TRUE	15.30
150	2	2962	3	4	3	1.27%	TRUE	67.31
151	5	2870	3	6	3	4.33%	TRUE	2.23

152	4	2781	3	5	3	7.30%	TRUE	1.52
153	4	2940	3	5	3	2.00%	TRUE	4.05
154	4	2859	3	5	3	4.70%	TRUE	2.81
155	3	2735	3	5	3	8.83%	TRUE	1.66
156	4	2785	3	5	3	7.17%	TRUE	1.39
157	6	2823	3	6	3	5.90%	TRUE	1.37
158	6	2924	3	7	3	2.53%	TRUE	31.91
159	3	2781	3	5	3	7.30%	TRUE	1.91
160	2	2969	3	4	3	1.03%	TRUE	19.81
161	5	2940	3	5	3	2.00%	TRUE	65.45
162	4	2766	3	4	3	7.80%	TRUE	2.02
163	2	2964	3	4	3	1.20%	TRUE	50.89
164	3	3456	4	5	4	13.60%	TRUE	4.09
165	5	2865	3	5	3	4.50%	TRUE	7.08
166	4	10257	11	12	11	6.75%	FALSE	3600.61
					12	14.53%	TRUE	1001.92
167	2	9497	10	10	10	5.03%	FALSE	3600.92
168	5	9512	10	12	10	4.88%	FALSE	3600.62
					11	13.53%	TRUE	911.76
169	4	9448	10	12	10	5.52%	TRUE	2762.55
170	5	10001	11	12	11	9.08%	FALSE	3600.72
					12	16.66%	FALSE	3600.49
171	6	10414	11	14	11	5.33%	FALSE	3601.57
					12	13.22%	FALSE	3600.71
					13	19.89%	TRUE	2741.05
172	4	9979	10	12	10	0.21%	FALSE	3600.73
					11	9.28%	FALSE	3603.55
					12	16.84%	TRUE	318.42
173	3	10059	11	11	11	8.55%	TRUE	1455.72
174	3	10408	11	12	11	5.38%	FALSE	3600.57
					12	13.27%	FALSE	3601.12
175	2	9509	10	10	10	4.91%	FALSE	3600.36
176	1	9525	10	10	10	4.75%	FALSE	3600.47
177	4	8674	9	11	9	3.62%	FALSE	3600.59
					10	13.26%	TRUE	1466.40
178	4	9919	10	12	10	0.81%	FALSE	3600.48

					11	9.83%	FALSE	3600.91
					12	17.34%	FALSE	3600.91
179	3	10019	11	12	11	8.92%	FALSE	3601.14
					12	16.51%	FALSE	3608.28
180	4	10238	11	13	11	6.93%	FALSE	3600.52
					12	14.68%	TRUE	1167.77
181	4	10053	11	13	11	8.61%	FALSE	3600.38
					12	16.23%	FALSE	3600.47
					13	22.67%	FALSE	3600.80
182	2	9611	10	11	10	3.89%	FALSE	3600.54
					11	12.63%	TRUE	1073.85
183	5	9855	10	12	10	1.45%	FALSE	3600.77
					11	10.41%	FALSE	3600.78
					12	17.88%	TRUE	1337.00
184	5	10304	11	13	11	6.33%	FALSE	3600.63
					12	14.13%	TRUE	3148.97
185	4	10853	11	13	11	1.34%	FALSE	3600.50
					12	9.56%	FALSE	3600.54
					13	16.52%	FALSE	3600.93
186	2	10943	11	12	11	0.52%	FALSE	3600.51
					12	8.81%	FALSE	3600.61
187	2	9287	10	10	10	7.13%	FALSE	3600.73
188	3	9646	10	11	10	3.54%	FALSE	3622.26
					11	12.31%	FALSE	3600.93
189	3	10312	11	12	11	6.25%	FALSE	3600.78
					12	14.07%	FALSE	3600.69
190	3	10762	11	12	11	2.16%	FALSE	3600.84
					12	10.32%	FALSE	3601.13
191	6	3709	4	7	4	7.28%	TRUE	4.97
192	4	4669	5	7	5	6.62%	TRUE	7.27
193	3	4577	5	7	5	8.46%	TRUE	16.70
194	5	5099	6	7	6	15.02%	TRUE	17.19
195	5	5023	6	9	6	16.28%	TRUE	69.25
196	6	4511	5	8	5	9.78%	TRUE	14.14
197	4	3951	4	6	4	1.23%	TRUE	19.05
198	4	5322	6	8	6	11.30%	TRUE	102.27

199	4	4669	5	6	5	6.62%	TRUE	26.30
200	3	5195	6	7	6	13.42%	TRUE	100.46
201	6	5401	6	8	6	9.98%	TRUE	9.81
202	4	3984	4	6	4	0.40%	TRUE	187.19
203	2	3920	4	5	4	2.00%	TRUE	128.97
204	4	4762	5	7	5	4.76%	TRUE	67.17
205	3	5044	6	7	6	15.93%	TRUE	87.17
206	3	4632	5	6	5	7.36%	TRUE	74.14
207	3	5163	6	7	6	13.95%	TRUE	9.98
208	5	4906	5	7	5	1.88%	TRUE	271.43
209	2	3859	4	5	4	3.53%	TRUE	7.09
210	5	4603	5	8	5	7.94%	TRUE	13.63
211	3	4178	5	5	5	16.44%	TRUE	13.58
212	5	4685	5	9	5	6.30%	TRUE	48.85
213	3	4124	5	6	5	17.52%	TRUE	6.89
214	3	4766	5	7	5	4.68%	TRUE	33.14
215	4	4783	5	6	5	4.34%	TRUE	83.08
216	1	2928	3	3	3	2.40%	TRUE	2.66
217	1	3532	4	4	4	11.70%	TRUE	14.53
218	1	2870	3	3	3	4.33%	TRUE	3.52
219	1	2816	3	3	3	6.13%	TRUE	242.02
220	1	2954	3	3	3	1.53%	TRUE	102.11
221	1	2857	3	3	3	4.77%	TRUE	34.86
222	1	2882	3	3	3	3.93%	TRUE	19.16
223	1	2835	3	3	3	5.50%	TRUE	3.11
224	1	2808	3	3	3	6.40%	TRUE	16.84
225	1	2791	3	3	3	6.97%	TRUE	21.34
226	1	2934	3	3	3	2.20%	TRUE	430.29
227	1	2872	3	3	3	4.27%	TRUE	25.34
228	1	1891	2	2	2	5.45%	TRUE	0.44
229	1	2969	3	3	3	1.03%	FALSE	3600.07
230	1	2782	3	3	3	7.27%	TRUE	8.66
231	1	2848	3	3	3	5.07%	TRUE	3.36
232	1	2933	3	3	3	2.23%	TRUE	576.10
233	1	2772	3	3	3	7.60%	TRUE	2.39
234	1	2937	3	3	3	2.10%	TRUE	3.67

235	1	2761	3	3	3	7.97%	TRUE	6.23
236	1	2906	3	3	3	3.13%	TRUE	24.28
237	1	2908	3	3	3	3.07%	TRUE	21.41
238	1	2854	3	3	3	4.87%	TRUE	17.95
239	1	2785	3	3	3	7.17%	TRUE	2.27
240	1	2910	3	3	3	3.00%	TRUE	12.05
241	1	9944	10	10	10	0.56%	FALSE	3600.28
242	1	9917	10	10	10	0.83%	FALSE	3600.78
243	1	8837	9	9	9	1.81%	FALSE	3600.64
244	1	9618	10	10	10	3.82%	FALSE	3600.84
245	1	10379	11	11	11	5.65%	FALSE	3601.32
246	1	10677	11	11	11	2.94%	FALSE	3600.66
247	1	9485	10	10	10	5.15%	FALSE	3600.75
248	1	9989	10	10	10	0.11%	FALSE	3600.49
249	1	10171	11	11	11	7.54%	TRUE	370.94
250	1	9192	10	10	10	8.08%	FALSE	3600.37

TABLE A.1: Feasible(Q) with 20 operations

Appendix B

Computer experiment results for workforce assignment problem with 170 operations

		Same-Stations		Sequential-Stations		Sequential-Stations-Random		Sequential-Stations-One-Traveling-Worker		Min-Idle-One-Traveling-Worker		Random-One-Traveling-Worker		Sequential-Additional-Workers		Random-Additional-Workers	
Takt	LB	Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)
1	24	29	21	36	50	32	33	27	13	27	13	25	4	27	13	25	4
2	24	29	21	36	50	32	33	26	8	27	8	25	4	28	17	26	8
3	23	28	22	36	57	28	22	26	13	26	13	25	9	26	13	25	9
4	24	30	25	40	67	32	33	26	8	26	8	26	8	27	13	26	8
5	24	29	21	36	50	32	33	26	8	26	8	25	4	27	13	26	8
6	23	29	26	36	57	32	39	26	13	26	13	25	9	26	13	24	4
7	24	29	21	36	50	32	33	25	4	25	4	25	4	27	13	26	8
8	24	29	21	36	50	32	33	29	21	29	21	26	8	27	13	26	8

9	23	29	26	36	57	28	22	27	17	27	17	25	9	27	17	25	9
10	24	29	21	36	50	32	33	26	8	26	8	25	4	27	13	26	8
11	23	28	22	36	57	28	22	27	17	27	17	25	9	27	17	24	4
12	24	29	21	36	50	32	33	26	8	26	8	25	4	27	13	25	4
13	24	29	21	36	50	28	17	26	8	26	8	25	4	26	8	25	4
14	24	30	25	36	50	32	33	26	8	26	8	25	4	27	13	25	4
15	24	29	21	36	50	32	33	26	8	26	8	25	4	27	13	26	8
16	24	29	21	36	50	32	33	26	8	26	8	26	8	27	13	25	4
17	24	29	21	36	50	32	33	26	8	26	8	25	4	27	13	25	4
18	24	30	25	36	50	32	33	27	13	27	13	26	8	27	13	26	8
19	24	30	25	36	50	32	33	26	8	26	8	25	4	28	17	25	4
20	24	29	21	36	50	32	33	26	8	26	8	25	4	28	17	26	8

TABLE B.1: Heuristics

Bibliography

[1] http://cordis.europa.eu/project/rcn/100702_en.html

[2] http://malyutins.github.io/Workforce_minimization/

[3] Akagi, F., Osaki, H., Kikuchi, S., 1983. A method for assembly line balancing with more than one worker in each station. International Journal of Production Research, 21(5), 755 - 770.

[4] Araújo, F.F.B., Costa, A.M., Miralles, C., 2012. Two extensions for the ALWABP: Parallel stations and collaborative approach. International Journal of Production Economics, 140(1), 483 - 495.

[5] Arcus A.L., 1966. A computer method of sequencing operations for assembly lines. International Journal of Production Research, 4, 259 - 277.

[6] Artigues, C., Demassey, S., Néron, E., 2013. Resource-constrained project scheduling: models, algorithms, extensions and applications. ISTE Ltd and John Wiley & Sons.

[7] Barketau, M., Kovalyov, M.Y., Weglarz, J., Machowiak, M., 2014. Scheduling arbitrary number of malleable tasks on multiprocessor systems. Bulletin of the Polish Academy of Sciences Technical Sciences. 62(2), 255 - 261.

[8] Battaïa, O., Delorme, X., Dolgui, A., Hagemann, J., Horlemann, A., Kovalev, S., Malyutin, S., 2015. Workforce minimization for a mixed-model assembly line in the automotive industry, International Journal of Production Economics, 170, 489 - 500.

[9] Battaïa, O., Delorme, X., Dolgui, A., Hagemann, J., Kovalev, S., Malyutin, S., 2014. Optimal Designof Assembly Lines with Flexible Workers, 20th Conference of the International Federation of Operational Research Societies (IFORS).

[10] Battaïa, O., Delorme, X., Dolgui, A., Malyutin, S., Horlemann, A., Kovalev, S., 2015. Workforce Minimization for a Mixed-Model Assembly

Line, Preprints of the Eighteenth International Working Seminar on Production Economics, 3, 51 - 63.

[11] Baybars I., 1986. A survey of exact algorithms for the simple assembly line balancing problem. Management Science, 32(8), 909-932.

[12] Blazewicz, J., Machowiak, M., Weglarz, J., Kovalyov, M.Y., Trystram, D., 2004. Scheduling malleable tasks on parallel processors to minimize the makespan. Annals of Operations Research, 129, 65 - 80.

[13] Blazewicz, J., Cheng, T.C.E., Machowiak, M., Oguz, C., 2011. Berth and quay crane allocation: a moldable task scheduling model. Journal of the Operational Research Society, 62(7), 1189 - 1197.

[14] Blazewicz, J., Drabowski, M., Weglarz, J., 1986. Scheduling multiprocessor tasks to minimize schedule length. IEEE Transactions on Computers. 35(5), 389 - 393.

[15] Blazewicz, J., Ecker, K., Plateau, B., Trystram, D., 2000. Handbook on parallel and distributed processing, Springer, Berlin.

[16] Belmokhtar S., Dolgui A., Guschinsky N., Levin G., 2006. An integer programming model for logical layout design of modular machining lines. Computers and Industrial Engineering, 51(3), 502 - 518.

[17] Besikci, U., Bilge, U., Ulusoy, G., 2013. Resource dedication problem in a multi-project environment, Flexible Services and Manufacturing Journal, 25 (1-2), 206 - 229.

[18] Blum, C., Miralles, C., 2011. On solving the assembly line worker assignment and balancing problem via beam search. Computers & Operations Research, 38(1), 328 - 329.

[19] Borba, L., Ritt, M., 2014. A heuristic and a branch-and-bound algorithm for the Assembly Line Worker Assignment and Balancing Problem. Computers & Operations Research, 45, 87 - 96.

[20] Borisovsky P., Delorme X., Dolgui A., 2013. Genetic algorithm for balancing reconfigurable machining lines. Computers and Industrial Engineering.

[21] Boysen, N., Emde, S., 2014. Scheduling the part supply of mixed-model assembly lines. European Journal of Operational Research, 239(3), 820 - 829.

[22] Boysen N., Fliedner M., and Scholl A., 2008. Assembly line balancing: which model to use when. International Journal of Production Economics, 111, 509 - 528.

[23] Boysen N., Fliedner M., Scholl A., 2007. A classification of assembly line balancing problems. European Journal of Operational Research, 183, 674 - 693.

[24] Burkard, R.E., Dell'Amico, M., Martello, S., 2009. Assignment problems. SIAM e-books, Philadelphia.

[25] Burke, E., Trick, M., 2005. Practice and theory of automated timetabling V: 5th International Conference. Springer.

[26] Camm, J.D., Magazine, M.J., Polak, G.G., Zaric, G.S., 2008. Scheduling parallel assembly workstations to minimize a shared pool of labor. IIE Transactions, 40(8), 749 - 758.

[27] J. C. Carver and T. Epperly, 2014. Software engineering for computational science and engineering, Computing in Science and Engineering, 16, 6 - 9.

[28] Choundhary, A.N., Narahari, B., Nicol, D.M., Simha, R., 1994. Optimal processor assignment for a class of pipelined computations. IEEE Transactions on Parallel and Distributed Systems, 5/4, 439 - 445.

[29] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2001. Introduction to Algorithms, 2nd edn. MIT Press, Cambridge, 10 - 12.

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

[31] De Bruecker, P., Van den Bergh, J., Beliën, J., Demeulemeester, E., 2015. Workforce planning incorporating skills: State of the art. European Journal of Operational Research, 243(1), 1-16.

[32] Delorme, X., Dolgui, A., Malyutin S., 2014. A multi-objective algorithm for balancing reconfigurable transfer lines, 20th Conference of the International Federation of Operational Research Societies (IFORS).

[33] Demeulemeester, E.L., Herroelen, W.S., Elmaghraby, S.E., 1996. Optimal procedures for the discrete time/cost tradeoff problem in project networks. European Journal of Operational Research, 88(1), 50-68.

[34] Dolgui, A., Proth, J.-M., 2006. Systèmes de production modernes. Hermès Science/ Lavoisier, Londres.

[35] Dolgui, A., Proth, J.-M., 2010. Supply Chains Engineering: Useful Methods and Techniques. Springer, London.

[36] Dolgui A., Ihnatsenka I., 2009. Branch and bound algorithm for a transfer line design problem: Workstations with sequentially activated multi-spindle heads. European Journal of Operational Research, 197(3), 1197 - 1232.

[37] Dolgui A., Finel B., Guschinsky N., Levin G., Vernadat F., 2006. MIP approach to balancing transfer lines with blocks of parallel operations. IIE Transactions, 38, 869 - 882.

[38] Dolgui A., Guschinsky N., Levin G.. 1999. On problem of optimal design of transfer lines with parallel and sequential operations. Proceedings of the 7-th IEEE international conference on emerging technologies and factory automation (ETFA-99), 1, 329 - 334.

[39] Dolgui, A., Kovalev, S., Kovalyov, M.Y., Malyutin, S., Soukhal, A., 2015. Minimizing the number of workers for one cycle of a paced production line. IFAC-PapersOnLine, 48 (3), 2281?2286.

[40] Dolgui, A., Guschinsky, Levin, G., 2000. Approaches to balancing of transfer lines with blocks of parallel operations; Preprints No 8, Institute of Engineering Cybernetics of Academy of Sciences of Belarus/University of Technology of Troyes, 42p.

[41] Dolgui A., Guschinskaya O., Eremeev A., 2008. MIP-based GRASP and genetic algorithm for balancing transfer lines. In Proceedings of the Matheuristics 2008: The second international workshop on model based metaheuristics, 22p.

[42] Drozdowski, M., 1996. Scheduling multiprocessor tasks - An overview. European Journal of Operational Research, 94(2), 215-230.

[43] Dutot, P.-F., Mounié, G., Trystram, D., 2004. Scheduling parallel tasks approximation algorithms. In Handbook of Scheduling: Algorithms, Models, and Performance Analysis, edited by Joseph Y.-T. Leung, CRC Press, LLC.

[44] Du, J., Leung, J.Y-T., 1989. Complexity of scheduling parallel task systems. SlAM Journal on Discrete Mathematics 2(4), 473 - 487.

[45] Du, J., Leung, J.Y-T., Young, G.H., 1991. Scheduling chain-structured tasks to minimize makespan and mean flow time. Information and Computation, 92(2), 219 - 236.

[46] Essafi M., Delorme X., Dolgui A., 2012. A reactive GRASP and Path Relinking for balancing reconfigurable transfer lines. International Journal of Production Research, 50(18), 5213 - 5238.

[47] Essafi M., Delorme X., Dolgui A., Guschinskaya O., 2010. A MIP approach for balancing transfer line with complex industrial constraints. Computer and Industrial Engineering, 58, 393 - 400.

[48] Essafi M., Delorme X., Dolgui A., 2010. Balancing lines with CNC machines: A multi-start and based heuristic. CIRP Journal of Manufacturing Science and Technology, 2, 176 - 182.

[49] Finel B., Dolgui A., Vernadat F., 2008. A random search and backtracking procedure for transfer line balancing. International Journal of Computer Integrated Manufacturing, 21(4), 376 - 387.

[50] Ford, H., Crowther, S., 1923. My Life and Work. Garden City, NY: Garden City Publishing.

[51] A.T. Ernst, H. Jiang, M. Krishnamoorthy, D. Sier, 2004. Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research, 153, 3 - 27.

[52] Helgeson W.P., Birnie D.P., 1961. Assembly line balancing using the ranked positional weight technic. Journal of Industrial Engineering, 12, 394-398.

[53] Garey, M.R., Johnson, D.S., 1979. Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco.

[54] Ghoddousi, P., Eshtehardian, E., Jooybanpour, S., Javanmardi, A., 2013. Multi-mode resource-constrained discrete time-cost-resource optimization in project scheduling using non-dominated sorting genetic algorithm. Automation in Construction, 30, 216 - 227.

[55] Giard, V., Jeunet, J., 2010. Optimal sequencing of mixed models with sequence-dependent setups and utility workers on an assembly line. International Journal of Production Economics, 123(2), 290-300.

[56] Gökçen, H., Ağpak, K., Benzer, R., 2006. Balancing of parallel assembly lines. International Journal of Production Economics, 103, 600 - 609.

[57] Gökçen, H., Ağpak, K., 2006. A goal programming approach to simple U-line balancing problem. European Journal of Operational Research, 171, 577 - 585.

[58] Guschinskaya O., Dolgui A., Guschinsky N., Levin G., 2008. A heuristic multi-start decomposition approach for optimal design of serial machining lines. European Journal of Operational Research, 189, 902 - 913.

[59] Hannay, J.E., MacLeod, C., Singer, J., Langtangen, H.P., Pfahl, D., Wilson, G., 2009. How do scientists develop and use scientific software? Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, 5069155, 1 - 8.

[60] Hunold, S., 2015. One step toward bridging the gap between theory and practice in moldable task scheduling with precedence constraints. Concurrency Computation, 27(4), 1010 - 1026.

[61] Kolisch, R., Sprecher, A., 1997. PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. European Journal of Operational Research, 96(1), 205 - 216.

[62] Kolisch, R., Sprecher, A., Drexl, A., 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science, 41(10), 1693 - 1703.

[63] Kovalyov, M.Y., Delorme, X., Dolgui, A., 2016. Workforce planning for cyclic production of multiple parts. The First International Workshop on Dynamic Scheduling Problems, Poznan.

[64] Koren, Y., Jovane, F., Heisel, U., Moriwaki,, T., Pritschow G., Ulsoy G., and VanBrussel H., 1999. Reconfigurable Manufacturing Systems. A Keynote paper. CIRP Annals, 48(2), 527 - 598.

[65] Kouvelis, P., Karabati, S., 1999. Cyclic scheduling in synchronous production lines, IIE Transactions, 31(8), 709 - 719.

[66] Ku, W.-Y., Beck, J.C. Mixed Integer Programming models for job shop scheduling: A computational analysis (2016) Computers and Operations Research, 73, 165 - 173.

[67] Lee, C.-Y., Vairaktarakis, G.L., 1997. Workforce planning in mixed model assembly systems. Operations Research, 45(4), 553 - 567.

[68] Li, H., Womer, K., 2012. Optimizing the supply chain configuration for make-to-order manufacturing. European Journal of Operational Research, 221(1), 118 - 128.

[69] Lloyd, E.L., 1981. Concurrent task systems. Operations Research, 29(1), 189 - 201.

[70] Lutz, C.M., Davis, K.R., 1994. Development of operator assignment schedules: a DSS approach. Omega 22(1), 57 - 67.

[71] Merelo-Guervós, J.J., Romero, G., García-Arenas, M., Castillo, P.A., Mora, A.M., Jiménez-Laredo, J.L., 2011. Implementation matters: programming best practices for evolutionary algorithms. Advances in Computational Intelligence, 333 - 340.

[72] Monma, C.L., Schrijver, A., Todd, M.J., Wei, V.K., 1990. Convex resource allocation problems on directed acyclic graphs: duality, complexity, apecial cases, and extensions. Mathematics of Operations Research, 15(4), 736 - 748.

[73] Moreira, M.C.O., Costa, A.M., 2013. Hybrid heuristics for planning job rotation schedules in assembly lines with heterogeneous workers. International Journal of Production Economics, 141(2), 552 - 560.

[74] Mutlu, O., Polat, O., Supciller, A.A., 2013. An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type II. Computers & Operations Research, 40(1), 418 - 426.

[75] Otto, A., Otto, C., Scholl, A., 2013. Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. European Journal of Operational Research, 228(1), 33 - 45.

[76] Ranjbar, M.R., Kianfar, F., 2007. Solving the discrete time/resource trade-off problem in project scheduling with genetic algorithms. Applied Mathematics and Computation, 191(2), 451 - 456.

[77] Ritt, M., Costa, A.M., Miralles, C., 2016. The assembly line worker assignment and balancing problem with stochastic worker availability. International Journal of Production Research, 54(3), 907 - 922.

[78] Rekiek B., Dolgui A., Delchambre A., and Bratcu A., 2002. State of art of assembly lines design optimization. Annual Reviews in Control, 26(2), 163 - 174.

[79] Salewski, F., Schirmer, A., Drexl, A., 1997. Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application. European Journal of Operational Research, 102(1), 88 - 110.

[80] Salveson M.E., 1955. The assembly line balancing problem. Journal of Industrial Engineering, 6 (3), 1825.

[81] Scholl A., 1999. Balancing and Sequencing of Assembly Lines. Physica-Verlag, Heidelberg.

[82] Scholl, A.; Boysen, N.; Fliedner, M., 2008. The sequence-dependent assembly line balancing problem. Operations Research Spectrum, 30, 579 - 609.

[83] Segal, J., Morris, C., 2008. Developing Scientific Software. IEEE Software, 25 (4), 18 - 20.

[84] Srinivasa Prasanna, G.N., Musicus, B.R., 1994. Generalized multiprocessor scheduling for directed acyclic graphs. In: Proceedings of Supercomputing 1994, IEEE Press, New York, 237 - 246.

[85] Sungur, B., Yavuz, Y., 2015. Assembly line balancing with hierarchical worker assignment. Journal of Manufacturing Systems, 37(1), 290 - 298.

[86] Tseng, L.Y., Chen, S.C., 2009. Two-phase genetic local search method for multimode resource-constrained project scheduling problem. IEEE Transactions on Evolutionary Computation, 13(4), 848-857.

[87] Vairaktarakis, G.L., Cai, X., 2003. Complexity of workforce scheduling in transfer lines. Journal Journal of Global Optimization, 27(2-3), 273 - 291.

[88] Vairaktarakis, G.L., Cai, X., Lee, C.-Y., 2002. Workforce planning in synchronous production systems. European Journal of Operational Research, 136(1), 551 - 572.

[89] Vairaktarakis, G.L., Winch, J.K., 1999. Worker cross-training in paced assembly lines, Manufacturing and Service Operations Management, 1(2), 112 - 131.

[90] N.S. Vidic, 2008. Developing methods to solve the workforce assignment problem considering worker heterogeneity and learning and forgetting. Doctoral Dissertation, University of Pittsburgh.

[91] Vilà, M., Pereira, J., 2014. A branch-and-bound algorithm for assembly line worker assignment and balancing problems. Computers & Operations Research, 44, 105-114.

[92] Wang, Q., Cheng, K.H., 1991. List scheduling of parallel tasks. Information Processing Letters, 37(5), 291 - 297.

[93] Wang, Q., Cheng, K.H., 1992. A heuristic of scheduling parallel tasks and its analysis. SIAM Journal on Computing, 21(2), 281 - 294.

[94] Wilson, J.M., 1986. Formulation of a problem involving assembly lines with multiple manning of work stations, International Journal of Production Research, 24(1), 59-63.

[95] Wilson, G., Aruliah, D.A., Brown, C.T., Chue Hong, N.P., Davis, M,, Guy., R.T., et al., 2014. Best Practices for Scientific Computing. PLoS Biol 12(1): e1001745. doi:10.1371/journal.pbio.1001745

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *24 October 2016*

Sergey MALYUTIN

Algorithms and Software for Decision Support in Design of Assembly and Transfer Lines

Speciality : Industrial Engineering

Keywords : assembly line, transfer line, line balancing, workforce assignment, algorithms, software, MILP, heuristics, computer experiments

Abstract :

An overview of existing problems and methods for the design of assembly and transfer lines is given. A new workforce assignment problem for a paced multi-product assembly line with a goal of minimising the number of workers is studied. Various precedence relations between operations and various functions of operation processing times, dependent on the number of workers, are considered. A new problem of multi-objective optimisation for a single product transfer line is solved. Several exact and heuristic methods and their computer implementations for both problems are developed by the author. An application of developed approaches to solving a real production problem relevant to the European project amePLM is demonstrated.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *24 Octobre 2016*

Sergey MALYUTIN

Algorithmes et logiciels pour aide à la décision dans la conception de lignes d'assemblage et des lignes de transfert

Spécialité: génie industriel

Mots clefs : ligne d'assemblage, ligne de transfert, équilibrage de ligne, affectation de ma main d'oeuvre, algorithmes d'optimisation, logiciel d'aide à la décision, études numériques

Résumé :

Une vue d'ensemble des problèmes et des méthodes pour la conception des lignes d'assemblage et d'usinage est donnée. Un nouveau problème d'affectation de la main-d'œuvre pour une ligne d'assemblage multi-produits cadancée avec un objectif de minimiser le nombre d'opérateurs est étudié. Diverses relations de priorité entre les opérations et les différentes fonctions définissant les temps d'opérations, en fonction du nombre d'opérateurs sont considérés. Un nouveau problème d'optimisation multiobjectif pour une ligne d'usinage mono-produit est formulé. Plusieurs méthodes exactes et heuristiques et leurs implémentations informatiques pour les deux problèmes sont développés par l'auteur. Un module logiciel d'aide à la décision pour résoudre ces problèmes est développé et implémenté dans un environnement d'un nouveau PLM d'IBM dans le cadre du projet europen amePLM. Ce module est testé sur un exemple réel de conception d'une ligne de montage des moteurs chez Mercedes Benz en Allemagne.